Interactive Simulation of Surgical Needle Insertion and Steering

Nuttapong Chentanez

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2010-129
http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-129.html

October 5, 2010

Acknowledgement

We thank members of the Berkeley Graphics Group and the Automation Lab for their helpful comments. We thank Allison Okamura, Noah Cowan, Robert Webster III, Landon Unninayar, and Vinutha Kallem at Johns Hopkins for the tissue phantom video, Judy Hoffman for marker tracking code, James Demmel and Samuel Webb Williams for discussions about sparse matrix solvers, Vincent Duindam of Intuitive Surgical and Jean Pouliot, I-Chow Joe Hsu, and Adam Cunha of UCSF for advice on prostate treatment. This work was supported in part by California MICRO 08-077, NSF Award CCF-0635381, NIH Award 1R01EB-006435-01A1, the UC Lab Fees Research Program, and by support from NVIDIA, Pixar, Autodesk, Intel, and Sony.

Interactive Simulation of Surgical Needle Insertion and Steering

by

Nuttapong Chentanez

B.S. (University of Michigan) 2005
M.S. (University of California, Berkeley) 2007

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:
Professor James F. O'Brien, Chair
Professor Ken Goldberg
Professor Jonathan Shewchuk
Professor Sara McMains

Fall 2010

The dissertation of Nuttapong Chentanez is approved:

Chair Date

Date

Date

Date

University of California, Berkeley

Fall 2010

Interactive Simulation of Surgical Needle Insertion and Steering

by

Nuttapong Chentanez

Abstract

Interactive Simulation of Surgical Needle Insertion and Steering

by

Nuttapong Chentanez

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor James F. O'Brien, Chair

We present algorithms for simulating and visualizing the insertion and steering of needles through deformable tissues for surgical training and planning. Needle insertion is an essential component of many clinical procedures such as biopsies, injections, neurosurgery, and brachytherapy cancer treatment. The success of these procedures depends on accurate guidance of the needle tip to a clinical target while avoiding vital tissues. Needle insertion deforms body tissues, making accurate placement difficult. Our interactive needle insertion simulator models the coupling between a flexible needle and deformable tissue. We introduce (1) a novel algorithm for local remeshing that quickly enforces the conformity of a tetrahedral mesh to a curvilinear needle path, enabling accurate computation of contact forces, (2) an efficient method for coupling a 3D finite element simulation with a 1D inextensible rod with stick-slip friction, and (3) optimizations that reduce the computation time for physically based simulations. We can realistically and interactively simulate needle insertion into a prostate mesh of 13,375 tetrahedra and 2,763 vertices at a 25 Hz frame rate on an 8-core 3.0 GHz Intel Xeon PC. The simulation models prostate brachytherapy with needles of varying stiffness, steering needles around obstacles, and supports motion planning for robotic needle insertion. We evaluate the accuracy of the simulation by comparing against real-world experiments in which flexible, steerable needles were inserted into gel tissue phantoms.

Professor James F. O'Brien
Dissertation Committee Chair

To my family and close friends,
for all your support throughout my years as a student.

Contents

Acknowledgments

I would like to first thanks my advisor Prof. James O'Brien for guiding me through my graduate study, giving constructive comments to my poor presentation early in my study, reading and improving my rough drafts of papers, teaching me good taste in graphics, encouraging me when things did not go as planned, pushing me when things needed to be done quickly and letting me explore freely during the initial phase of research. I would like to also thanks Prof. Jonathan Shewchuk for giving me excellent suggestions about research, teaching two of the best classes I've ever taken in computational geometry and mesh generation, helping me revise and sometimes rewrite several of my papers which made them shine an order of magnitude more. I would like to thanks Prof. Ken Goldberg for introducing me to the needle steering project and the follow–up works, giving me great suggestions about how to improve my presentations and papers, and being an impeccable example of how to manage a research group with lots of students working on many projects. I would like to also thanks Prof. Sara McMains for teaching me GPGPU programming, giving me great advice about searching the literature, asking me great questions during my talks and for reviewing drafts of my thesis.

I would like to also thanks all of my fellow students co-authors: Bryan Feldman, Tolga Goktekin, Bryan Klingner, Ron Alterovitz, Kris Hauser, Daniel Ritchie, Lita Cho and François Labelle for their ideas, help, support, suggestions, and criticisms throughout all the research I got involved in. I would like to also thank many graduate and undergraduate students in the Berkeley Graphics Group and the Automation Group for their help and their great friendships during my years at Berkeley. Notably, Bryan Feldman for giving me great advice during my first two years at Berkeley, Tolga Goktekin for his kind help many times during my internship at Pixar, Pushkar Joshi for welcoming me to Berkeley and for his help and suggestions many times throughout the years.

Finally, I would like to thank my parents, my fiancée, my brother, my grandmother and all my close friends for their love and support during my years of studying abroad. Their encouragement helps bring me up when things are rough in my life throughout my study.

Chapter 1

Introduction

Needle insertion is an essential component of many clinical procedures such as biopsies, injections, neurosurgery, and brachytherapy cancer treatment [1]. The success of these procedures depends on how closely the needle tip is maneuvered to the target. It is crucial that the needle avoid bone, critical structures, and organs [34]. Unfortunately, needle insertion deforms body tissues enough that poor accuracy is the norm in practice. For example, experienced physicians inserting radioactive seeds into the prostate gland for brachytherapy prostate cancer treatment experience average placement errors of 6.3 mm, about 15% of the prostate's diameter [63].

Computer simulations of needle insertion procedures enable physicians and other clinicians to train in a controlled environment that exposes them to both common and rare patient cases without risks to patient safety. Studies indicate that surgical skills learned using computational simulators directly improve operating room performance by significantly decreasing procedure time and reducing the frequency of medical errors by up to sixfold compared to traditional training [56, 55, 24]. Surgical simulations also have uses for pre-operative planning [4, 64].

We present a new simulator that models tissue deformation, needle elasticity, and their interaction. It allows us to realistically simulate the deflections that occur as thin needles travel through inhomogeneous tissues. A motivation for modeling needle elasticity is a new class of flexible, steerable needles recently developed in collaboration between Prof. Ken Goldberg's group at University of California at Berkeley and Prof. Allison Okamura's group at Johns Hopkins University [74, 72]. These bevel-tip steerable needles have a flexible shaft that curves as it penetrates soft tissue, due

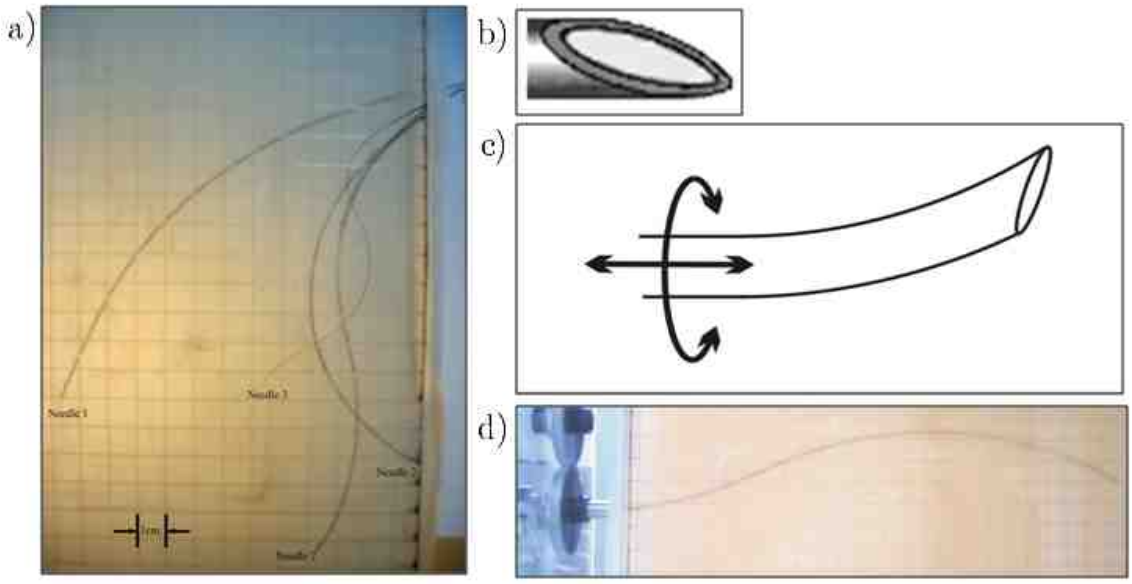

Figure 1.1: a) Trajectory of various bevel tip needles with different radii of curvature. b) A schematic view of the bevel tip. c) The needle can be steered by rotating its base. d) Needle trajectory with 180-degree base rotation in the middle. (Copyright Johns Hopkins University, 2005)

to asymmetric forces exerted at the needle's bevel tip. By twisting the needle as it is inserted, a physician can steer its tip around obstacles to reach clinical targets in soft tissues [5, 7], see Figure 1.1. It is not easy to learn how to control steerable needles, and realistic training simulations will accelerate their deployment in clinical practice.

Several impediments make it difficult to simulate the interaction between a needle and soft tissues: a static spatial discretization (*e.g.* a fixed finite element mesh) does not easily support the accurate computation of contact forces and needle steering; the mismatch between needle stiffness and tissue stiffness hinders numerical stability; and for training applications, the simulation must run at interactive rates. To address these challenges, I introduce (1) a novel algorithm for local remeshing, (2) an efficient algorithm for coupling a 3D finite element simulation and a 1D elastic rod simulation with stick-slip friction, and (3) several generally applicable optimizations for reducing computation time for physically based simulations. The algorithm is published in the proceedings of ACM SIGGRAPH 2009 [15].

Our remeshing algorithm efficiently relocates and creates nodes so that they lie along a curvilinear needle path in a volumetric mesh, enabling the simulation to apply cutting and frictional forces along

the needle shaft at mesh nodes while maintaining a high-quality tetrahedral mesh for computing tissue deformations. Our optimizations include accelerating the solution of the linear complementarity problem for node friction states, using a parallel conjugate gradient method on sparse matrices, efficiently updating the stiffness matrix structure in response to remeshing, and using a parallel lazy update of Jacobian matrices for tetrahedral elements.

Together, these algorithms and enhancements enable us to realistically simulate needle insertion in the prostate at interactive frame rates. We achieve frame rates of 25 Hz on an 8-core 3.0 GHz Intel Xeon PC for a prostate mesh of 13,375 tetrahedra and 2,763 vertices. We use realistic material properties for human tissue, making it more challenging than the more compliant materials for which real-time performance is usually reported. We present simulations of prostate brachytherapy with needles of different stiffness, simulations of needles steered around obstacles, and two applications to motion planning for robotic needle insertion. Throughout our trials, our remeshing procedure consistently maintained high mesh quality. We evaluate the accuracy of the simulator using data extracted from real-world experiments in which flexible, steerable needles were inserted into gel tissue phantoms. The simulated and real-world deformations are qualitatively and quantitatively similar.

a)

b)

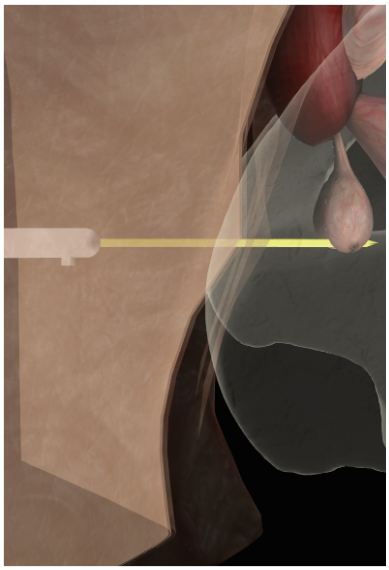

Figure 1.2: Screenshots from our prostate brachytherapy simulator. A needle is inserted from the left through the epidermis and dermis into the prostate gland. a) Bevel-tip flexible needle. b) Symmetric-tip stiff needle.

Chapter 2

Background

In this chapter, we first provide the background on tissue and needle continuum models as well as their discretizations. Then we introduce a time integration method for numerically evolving the discrete representations over time. Finally, we discuss how we obtain a volume discretization of the tissue from a given surface representation.

2.1 Tissue Constitutive Model and Discretization

Tissue can be modeled as an elastic solid, which is characterized by a time-dependent function $\phi : \Omega \subset \Re^3 \rightarrow \Re^3$, where Ω is a closed set representing the region in space occupied by the undeformed tissue. ϕ maps a point $\mathbf{u} \in \mathbf{\Re^3}$ representing undeformed material coordinates to a point $\mathbf{x} \in \mathbf{\Re^3}$ representing deformed world coordinates as illustrated in Figure 2.1. The internal force in an elastic solid is fully characterized by *stress*. Stress is a measure of force density on a surface, which is the force divided by the area of the surface. Cauchy stress σ is defined by a 3×3 tensor:

$$\sigma = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{pmatrix} \tag{2.1}$$

as illustrated in Figure 2.2a. We can then compute the traction, $\mathbf{t} = \sigma\mathbf{n}$, which is the force per unit area over an arbitrary oriented infinitesimal surface patch with the unit normal $\mathbf{n}$, shown in Figure 2.2b.

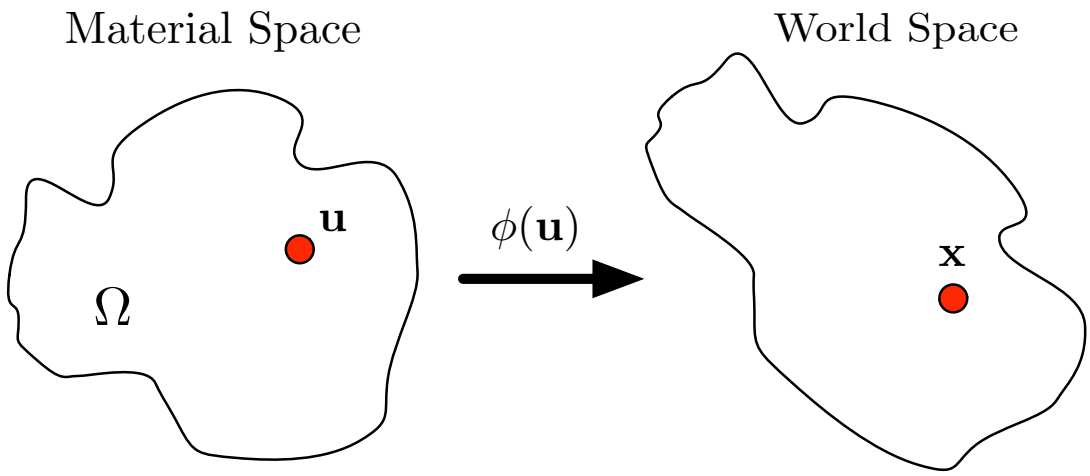

Figure 2.1: The undeformed elastic solid occupies the space $\Omega \subset \Re^3$. Any deformation of the solid can be described by a time-dependent mapping ϕ, which maps a point $\mathbf{u}$ in the material space to point $\mathbf{x}$ in the world space.

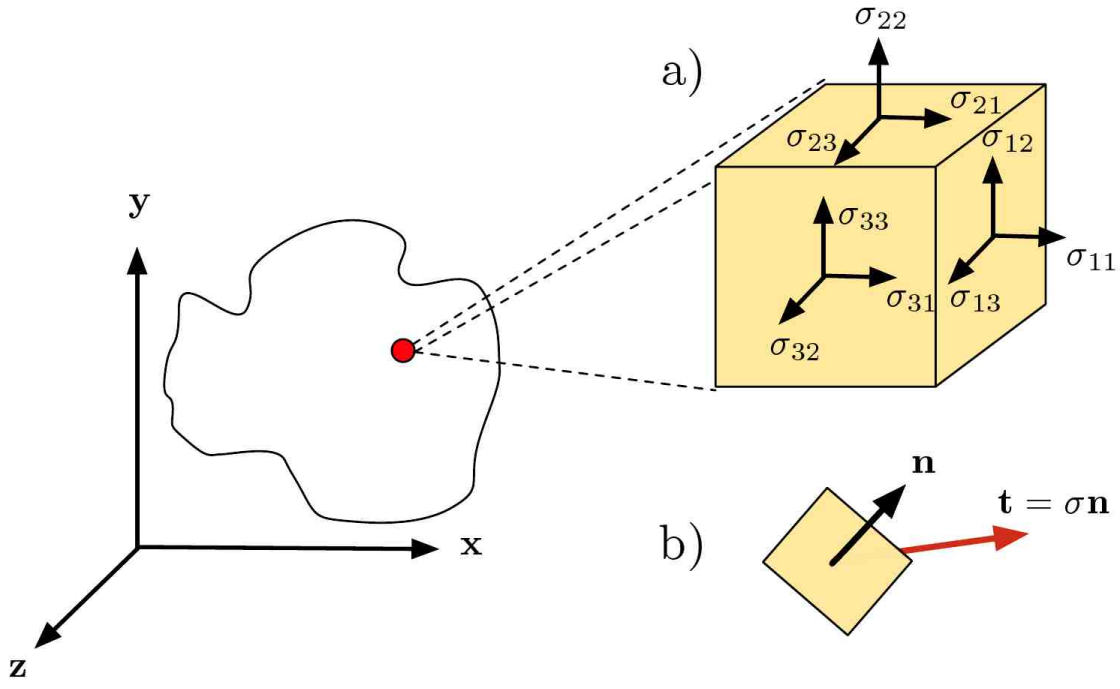

Figure 2.2: a) Components of the Cauchy stress tensor σ at a given point. b) Traction $\mathbf{t}$ at a given point depends on σ and the normal vector $\mathbf{n}$.

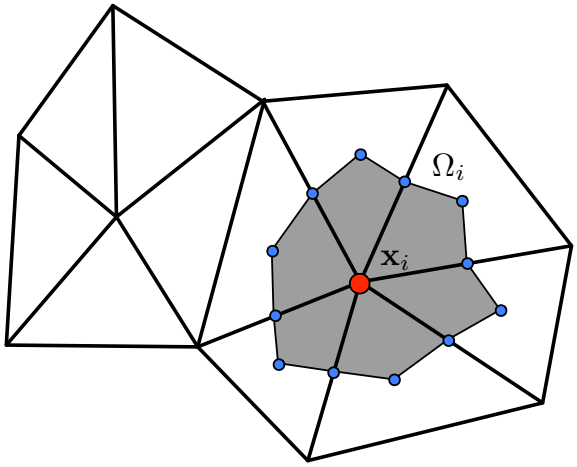

Figure 2.3: Discretization of a 2D tissue into a triangle mesh. The position of node i is $\mathbf{x}_i$. Its FVM integration region Ω_i, shown in grey, is bounded by the lines connecting the midpoints of the edges around node i to the midpoints of the triangles around node i.

The stress of an elastic solid at a given point is determined by three components: the deformation gradient $\frac{\partial \phi}{\partial \mathbf{u}}$, the tissue material properties and the constitutive model at that point. More commonly, stress is expressed in terms of strain ϵ, which can also easily be computed from the deformation gradient. As Chapter 5 discusses, we compute the stress directly from the deformation gradient, for efficiency and robustness.

For ease of exposition, we use 2D as an example of the derivation of the governing equations of the elastic solid discretized by a Finite Volume Method (FVM) [37], following Terran et al. [65]. Extension to 3D is straightforward and is discussed later on. The elastic solid occupies a domain $\Omega \subset \Re^2$. We discretize Ω into a triangle mesh consisting of nodes and triangles. Each node i is surrounded by a region Ω_i as shown in Figure 2.3. The mass of the node i is

$$m_i = \int_{\Omega_i} \rho \, d\mathbf{x} \tag{2.2}$$

where ρ is the density of the solid. The total force on this region is

$$\frac{d}{dt} \int_{\Omega_i} \rho \mathbf{v} \, d\mathbf{x} = \mathbf{F}_i = \oint_{\partial \Omega_i} \mathbf{t} \, dS = \oint_{\partial \Omega_i} \sigma \mathbf{n} \, dS \tag{2.3}$$

where dS is an infinitesimal line on $\partial \Omega_i$.

The first identity states that the time derivative of momentum equals force. The second identity follows from Gauss's theorem. The last identity follows from $\mathbf{t} = \sigma \mathbf{n}$.

The integral can be evaluated by summing up the contributions from the surrounding triangles. Figure 2.4a shows one of the triangles, its two edges ∂T_1 and ∂T_2 and the two edges of $\partial\Omega_i$ inside the triangle, labeled $\partial\Omega_1$ and $\partial\Omega_2$. As will be clear in Chapter 5, using the first-order FVM to discretize the domain Ω assigns a constant stress, σ, inside each triangle. Because σ is constant and the integral of the unit normal over any closed boundary is zero (following from the divergence theorem), we have

$$\oint_{\partial\Omega_1} \sigma\mathbf{n}\, dS + \oint_{\partial\Omega_2} \sigma\mathbf{n}\, dS + \oint_{\partial T_1} \sigma\mathbf{n}\, dS + \oint_{\partial T_2} \sigma\mathbf{n}\, dS = 0, \tag{2.4}$$

$$\oint_{\partial\Omega_1} \sigma\mathbf{n}\, dS + \oint_{\partial\Omega_2} \sigma\mathbf{n}\, dS = -\oint_{\partial T_1} \sigma\mathbf{n}\, dS - \oint_{\partial T_2} \sigma\mathbf{n}\, dS. \tag{2.5}$$

Therefore, we evaluate the integral of $\sigma\mathbf{n}$ over $\partial\Omega_1$ and $\partial\Omega_2$ by instead evaluating along ∂T_1 and ∂T_2. The equations also show that the integral over any smooth path inside the triangle such as the one shown in Figure 2.4b would yield an identical result, given that the path connects the same pair of points on the two triangle edges, e_1 and e_2, incident on $\mathbf{x}_i$. Choosing an arbitrary path that connects the mid-points of e_1 and e_2, the integral becomes simply $-\sigma\mathbf{n}\frac{e_1}{2} + -\sigma\mathbf{n}\frac{e_2}{2}$, where $\mathbf{n}_1$ and $\mathbf{n}_2$ are the normal vectors of the edges e_1 and e_2 respectively. Hence, the force contribution on node i from this triangle can be updated as

$$\mathbf{F}_i \mathrel{+}= \mathbf{F}_i - \frac{1}{2}\sigma(|e_1|\mathbf{n}_1 + |e_2|\mathbf{n}_2), \tag{2.6}$$

where $|e_1|$ and $|e_2|$ are the lengths of the edges, e_1 and e_2 respectively.

We extend the result to 3D space by replacing the line integral with the surface integral and replacing the area integral with the volume integral. For each tetrahedron, three faces contribute one-third of their force to a given node, hence we update the force on node x_i with

$$\mathbf{F}_i \mathrel{+}= \mathbf{F}_i - \frac{1}{3}\sigma(a_1\mathbf{n}_1 + a_2\mathbf{n}_2 + a_3\mathbf{n}_3) \tag{2.7}$$

where the a_i and the $\mathbf{n}_i$ are the area and the normal vector of the faces of a given tetrahedron incident to node i.

One could use Equation 2.7 in computing the tissue force on each node by first computing σ inside each tetrahedron. However, in practice we can rewrite Equation 2.7 in terms of a different stress measure, which permits a significant amount of precomputation that can improve the performance greatly. The precomputation is discussed in more detail in Chapter 5. The damping of the material's vibration over time will also be discussed there. External force can be added to each node to model the tissue interaction with the outside world.

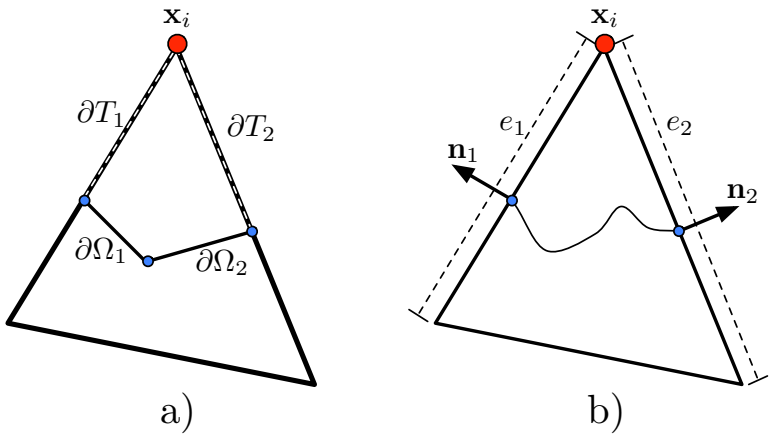

Figure 2.4: a) A subset of region Ω_i inside a triangle. b) The integral of the normal along any arbitrary path connecting the midpoints of the two edges is the same regardless of the path.

2.2 Needle Constitutive Model and Discretization

We model the needle as a one-dimensional elastic rod, described by a curve Γ in 3D space. Let $\mathbf{x}(s)$, $\mathbf{v}(s)$, and $\mathbf{a}(s) \in \Re^3$ be the position, velocity and acceleration along the curve Γ parameterized by arclength. We will describe the dynamic of an elastic rod using the energy formulation presented by Spillmann and Teschner [61]. There are several types of energies associated with a given elastic rod: $T(\Gamma)$ is the *kinetic energy*, which increases as the whole or part of the elastic rod moves faster. It is given by

$$T = \int_\Gamma \rho\pi r^2 \mathbf{v} \cdot \mathbf{v}\, ds \tag{2.8}$$

where ρ and r are the density and the radius of the rod respectively. $E(\Gamma)$ is the *potential energy*, which consists of bending, stretching and twisting energy. Another important energy is the *dissipation energy* $D(\Gamma)$, which models the internal friction and viscoelastic behavior of the rod. The expressions for $E(\Gamma)$ and $D(\Gamma)$ are more complicated than $T(\Gamma)$ and are discussed in more details in Chapter 6.

We can view the dynamic of the elastic rod as the conversion between these energies over time. Mathematically, this process is characterized by a critical point of the Lagrangian $L = T - E + D$. Using the calculus of variations, we obtain the equation of motion of an elastic rod,

$$\frac{d}{dt}\frac{\partial T}{\partial \mathbf{v}} - \frac{\partial T}{\partial \mathbf{x}} + \frac{\partial E}{\partial \mathbf{x}} + \frac{\partial D}{\partial \mathbf{v}} = 0. \tag{2.9}$$

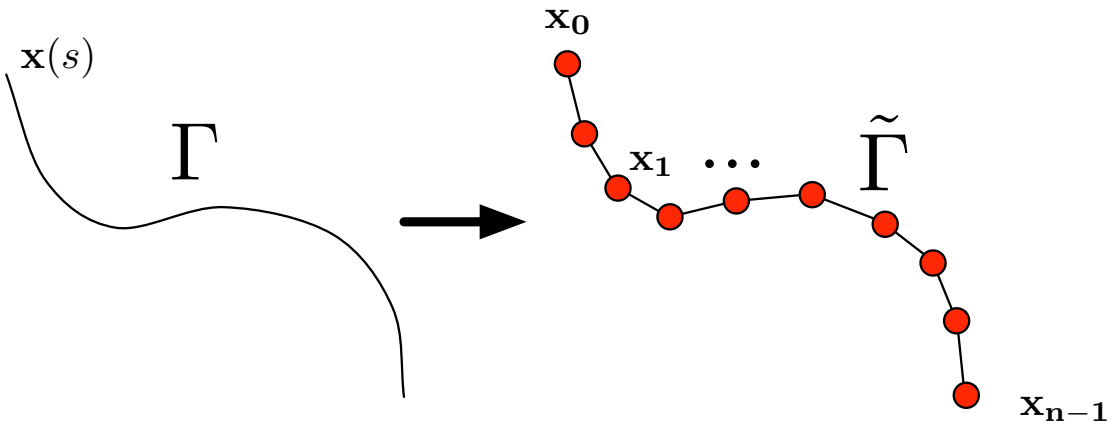

Figure 2.5: An elastic rod represented by the curve $\Gamma(s)$ can be discretized ito a piecewise linear curve $\tilde{\Gamma}$, which is fully described by $\mathbf{x}_0 \ldots \mathbf{x}_{n-1}$.

To numerically solve for the dynamic equilibrium, we discretize the elastic rod into piecewise linear segments as shown in Figure 2.5. Let $\mathbf{x}_i$, $\mathbf{v}_i$, $\mathbf{a}_i \in \Re^3$ be the position, the velocity, and the acceleration of node i of the elastic rod, respectively, where $0 \leq i < n$. The curve Γ is approximated as

$$\tilde{\Gamma} = \bigcup_{i=0}^{n-2} \tilde{\Gamma_i}, \tag{2.10}$$

$$\tilde{\Gamma_i} = \{\mathbf{x} : \exists t \in [0,1], \mathbf{x} = t\mathbf{x}_i + (1-t)\mathbf{x}_{i+1}\}. \tag{2.11}$$

We can express the energies in term of these discrete variables through symbolic integration and compute the appropriate partial derivatives to give

$$\mathbf{F} = \mathbf{M}\mathbf{a}, \tag{2.12}$$

$$\mathbf{F} = -\frac{dE(\Gamma)}{d\mathbf{x}} - \frac{dD(\Gamma)}{d\mathbf{v}}, \tag{2.13}$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{F}$ is the vector of the internal forces at all the nodes, $\mathbf{x}$ and $\mathbf{v}$ are the vectors resulting from concatenating $\mathbf{x}_i$ and $\mathbf{v}_i$ respectively. Note that $\mathbf{a}$ arises in the expression for the time derivative of $\mathbf{v}$. To simulate needle interaction with the outside world, we add external forces such as gravity and friction to the equation.

2.3 Time Integration

Once the acceleration of each node is determined, we numerically integrate it forward in time to obtain the velocity and the position. One of the simplest time integration methods is explicit Euler integration, which consist of the two equations

$$\mathbf{v}_i^{k+1} = \mathbf{v}_i^k + \Delta t \mathbf{a}_i^k, \tag{2.14}$$

$$\mathbf{x}_i^{k+1} = \mathbf{x}_i^k + \Delta t \mathbf{v}_i^k, \tag{2.15}$$

where k is the current time index and Δt is the size of the time step. For a sufficiently small Δt, and sufficiently large damping, one could use explicit Euler integration to advance the simulation forward from time index k to $k+1$. The position and the velocity at time index $k+1$ result in a different internal force and hence a different nodal acceleration. We could then continue the simulation to time index $k+2$, and so on. Unfortunately, for practical values of tissue and needle material properties and Δt, the explicit Euler integrator tends to be unstable. Therefore, our simulator uses a more sophisticated time integration technique called implicit Newmark integration, which is discussed in Section 4.3.

In this formulation, the needle and the tissue are not yet coupled. In general, they exert friction forces and normal forces on each other. We discuss the role of these forces and how we model them in Chapter 7.

2.4 Mesh Generation

As discussed in Section 2.1, we discretize the region in 3D space occupied by the tissue into a tetrahedral mesh. The most common representation of a surface in computer graphics is a polygonal mesh. Figure 2.6a shows two polygonal meshes that represent a prostate gland and the tissue around it. Our goal is to generate a tetrahedral mesh whose triangular faces conform to the prostate surface embedded inside. We first convert each polygonal mesh into a signed distance field using the method presented by Bærentzen and Aanæs [9]. We pick a convention that the signed distance field is positive outside the surface, negative inside the surface and zero on the surface. Suppose the signed distance functions of the exterior surface and the prostate are $d_E(\mathbf{x})$ and $d_P(\mathbf{x})$ respectively. We

can combine the two functions to get

$$d(\mathbf{x}) = \max(d_E(\mathbf{x}), -d_P(\mathbf{x})), \tag{2.16}$$

which is negative inside the region between the exterior and the prostate and positive everywhere else. Then we generate a tetrahedral mesh using the isosurface stuffing algorithm developed by Labelle and Shewchuk [36]. Isosurface stuffing takes as input a bounding box and a signed distance function $d(\mathbf{x})$. It outputs a tetrahedral mesh that represents the region inside the bounding box with an internal boundary separating the negative and positive regions of $d(\mathbf{x})$, as shown in Figure 2.6b. The algorithm first overlays a tetrahedral background grid over the bounding box. Then it snaps some of the nodes that are close to the zero-contour onto the contour. Finally, it splits all tetrahedra that span both the positive and the negative regions. We further modify the output tetrahedral mesh by throwing away the tetrahedra whose centers of mass lie outside the exterior surface. The resulting mesh is shown in Figure 2.6c. The mesh is now ready to be used by our simulator.

We could extend this approach to handle multiple internal organs in a straightforward manner. In some cases, generating a conforming tetrahedral mesh is not feasible, such as when organs are too small. We simply then alter the tissue properties of the tetrahedra intersecting the organs based on the amount of the overlapped volume. A more sophisticated approach based on numerical coarsening of elastic stress tensor presented by Lily et al.[32] can be employed for this purpose as well.

Other tetrahedral mesh generation techniques such as the advancing front method developed by Ito et al. [30] could be used to generate the tetrahedral mesh as well. However, they are more difficult to implement than isosurface stuffing. Another benefit of isosurface stuffing is has a theoretical guarantee on the quality of the mesh it generates, which is crucial for the stability and accuracy of the simulation, while other mesh generation algorithms do not.

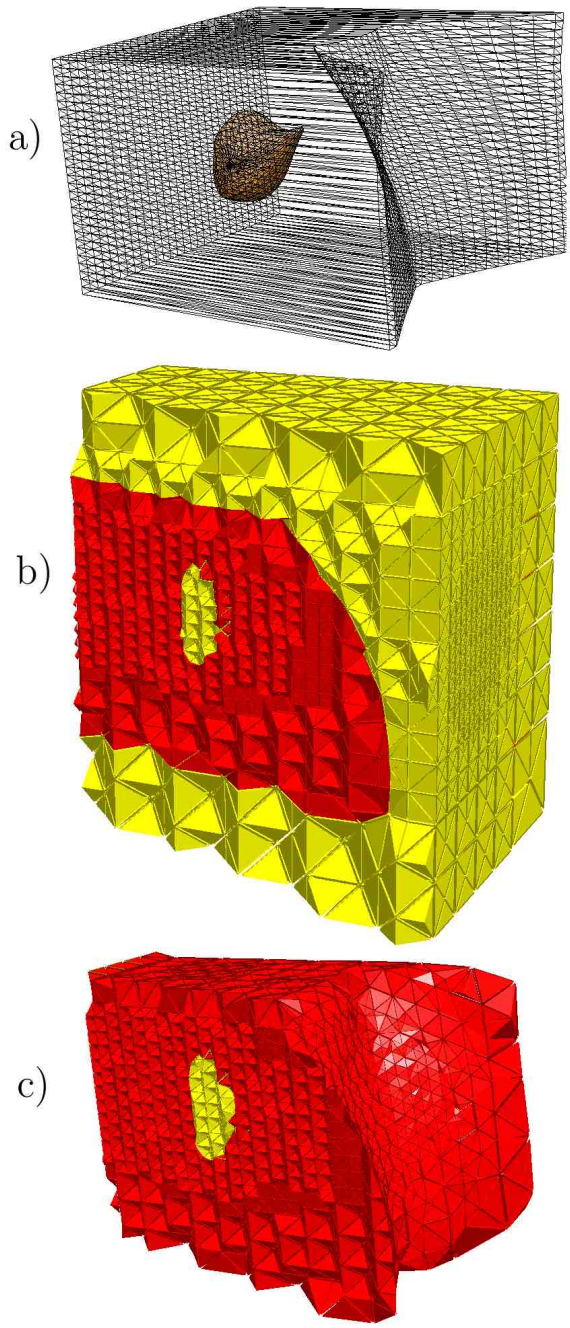

Figure 2.6: Mesh generation. a) Surface triangular meshes representing a male prostate and the tissue around it. b) Isosurface stuffing generates a two-sided tetrahedral mesh that encloses the bounding box of the mesh and conforms to the surfaces. Cutaway view is shown to expose the internal tetrahedra. c) Tetrahedra that are outside the exterior surface are removed. The remaining tetrahedra are used in our simulator.

Chapter 3

Previous Work

This dissertation draws material from several fields to develop the algorithm for simulating the coupled tissue and needle interaction. In this chapter, we first discuss previous work on elastic solid and elastic rod simulations, which are the basis of our simulator. Next, we will discuss other needle insertion simulations. Then we discuss the related work on measuring the tip and the friction forces during needle penetration and retraction, which are modeled in our simulator. Finally, we discuss previous work in remeshing, which motivates the development of our needle tip remeshing algorithm.

3.1 Elastic Solid Simulation

Elasticity theory is used to derive the constitutive model that represents the relationship between stress and strain in a deformable solid. An excellent introduction to this field can be found in a textbook by Fung [23]. Terzopoulos's seminal paper [67] in 1987 introduced deformable body simulation based on elasticity theory to the computer graphics community. It is based on a Finite Difference Method (FDM) where the partial differential equation that governs the behavior of an elastic body is discretized on a regular cubical grid. The Finite Element Method (FEM) [16] was introduced to computer graphics by Terzopoulos and Fleischer [66] in 1988 and was later extended to simulate brittle fracture by O'Brien and Hodgins [45]. They discretize the elastic body with a tetrahedral mesh, which can conform better to an arbitrary domain shape than a cubical grid and

can vary in size. They derive the governing equations using a quadratic Green strain, which is rotation-invariant. However, the simulated material becomes relatively less stiff when subject to compression compared to when it is undeformed. Therefore, the elements become inverted quite easily, which can cause the simulation to become unstable, namely the nodes positions and velocities grow without bound. Muller and Gross [42] use a linear Cauchy strain to derive the governing equations, which would not become weaker under compression, but is not rotation-invariant and thus will result in a bulging artifact when the object is rotated. They circumvent the artifact by using a co-rotational approach [22] where each element is rotated to a reference frame before the elastic force is computed and the resulting force is rotated back to the original space. Nonetheless, the method still becomes unstable when an element is inverted during the simulation. Irving et al. [29] use the linear Finite Volume Method (FVM) to simulate the deformation of an elastic body, which is shown by Teran et al. [65] to be equivalent to the linear FEM. However, the FVM has a more intuitive geometric interpretation than the FEM. It can also handle element inversion robustly by first decomposing the deformation gradient of each tetrahedron with the singular value decomposition (SVD). If any of the eigenvalues is negative, which implies that the tetrahedron is inverted, the algorithm makes it positive and negates the corresponding eigenvector. The modified deformation gradient is then used to compute the elastic force. The modification introduces an artificial force that acts to uninvert the element. The method can robustly handle large deformations and can easily be extended to handle anisotropic and nonlinear materials. Therefore, we use it as the basis of our tissue simulator.

Several open-source surgical simulation toolkits have been developed for soft tissue deformations. GiPSi [14] can simulate deformable bodies using either FEM or mass-spring models. A more recent framework is the Simulation Open Framework Architecture (SOFA) [3], which can simulate deformable bodies, rigid bodies, and particle-based fluids. In addition to simulating the tissue deformations, a surgery simulator must also model the tool-tissue interactions. FEM with tetrahedral remeshing was used to simulate soft-tissue cutting by Nienhuys and Stappen [43]. They split the tetrahedra along the cutting plane and remove the resulting tetrahedra that are badly shaped. Picinbono et al. [51] simulates deformable bodies in realtime by using linear elasticity when the tissue deformation is small and nonlinear elasticity when the deformation is large. Lindlad and Turkiyyah [38] use the discontinuous basis FEM to handle the cutting of elements, by aligning the discontinuities with the cut plane.

3.2 Elastic Rod Simulation

The needle can be modeled as a 1D elastic rod that bends and twists in 3D space. The elastic rod is governed by Cosserat theory, which is well investigated in the field of nonlinear elasticity. An excellent introduction to the theory is in the book by Antman [8]. The governing equations of the elastic rod can be discretized onto line elements with FEM as discussed in detail in the books by Simo [59] and Cao [52].

In the computer graphics community, Cosserat theory was first introduced by Pai [47], who solves a Boundary Value Problem (BVP) in each time step. However, the method was very computationally demanding, and had no guarantee on the running time complexity. A significant improvement has been made by Bertails et al. [13] to make the simulation run in $O(n^2)$ per time step, where n is the number of nodes, by making several simplifying assumptions. Bertails [12] later sped up the computation to $O(n)$. Loock and Schömer [39] use length springs, angular springs, and torsional springs to simulate an elastic rod that resists stretching, bending, and twisting. The static equilibrium of a rod modeled by joined elements, where each element has position and orientation represented by a quaternion, is solved by Gregoire et al. [26]. It was later augmented to include dynamics and solved in $O(n)$ by Spillmann and Teschner [61]. The quarternions and the positions are not independent and must be constrained in certain degrees of freedom to agree with one another. This significantly complicates the implicit integration implementation. Bergou et al. [11] compute the Bishop frame of the curve representing the rod, which is twist-free and represent the torsion of the rod by storing the angle of the rod twist from the Bishop frame, thus eliminating the use of quaternion. They assume that the twist wave propagates at an infinite speed through the rod. This allows the twist to be treated as a static problem and greatly simplifies the computation. Our needle simulator builds upon both the work of Spillmann and Teschner [61] and Bergou et al. [11].

3.3 Needle Insertion Simulation

Abolhassani et al. [1] survey recent work on needle insertion modeling and simulation. A physically-based simulation for the insertion of rigid needles into 2D tissue is developed by Alterovitz et al. [6]. The work uses the FEM to simulate tissue and employs a local mesh modification algorithm to ensure that the tissue mesh has nodes along the needle shaft. Flexible symmetric-tip needle inser-

tion into 2D tissue is considered by DiMaio and Salcudean [20]. They model the tissue as a linear elastic material and use Cauchy strain to measure the tissue deformation. This generates a constant stiffness matrix, which allows them to pre-compute the inverse of the matrix and use it to directly solve for the tissue deformation. They employ a local mesh modification to ensure that the tissue and the needle share nodes, then modify the inverse of the stiffness matrix accordingly with a simple rank-one update. Alterovitz et al. [5] simulates the insertion of a non-stiff bevel tip needle into 2D tissue, where the needle follows a curved path and not resist the tissue deformation.

For 3D tissues, simulations of the insertion of rigid needles have been developed using the mass-spring model [40] and the non-physically-based chain-mail model [70]. They do not employ mesh modification; they simply distribute the needle force to the nearby tissue nodes with non-physical distributions such as a Gaussian distribution. Nienhuys and Stappen [44] use FEM and recursive refinement of the tissue mesh until the mesh nodes are very close to the needle shaft. The recursive refinement creates many small tetrahedra, which significantly increase the simulation time. Goksel et al. [25] simulate 3D rigid needle insertion in linear elastic tissues with local remeshing by node snapping and face splitting. Their mesh modification algorithm occasionally generates badly shaped tetrahedra, which adversely affect the robustness of the simulation. Dehghan and Salcudean [19] extend this method to support nonlinear materials. We know of no prior work that can accurately simulate the coupled tissue and needle deformation in 3D interactively.

3.4 Forces During Needle Insertion

Several researchers have made measurements of the forces between the tissue and the needle during a needle insertion into real material. Simone and Okamura [60] propose that three forces play key roles during a needle insertion, namely, stiffness, friction and cutting forces. They measure these forces during a needle insertion into a bovine liver and provide model equations for these forces based on a least-squares fit with the experimental data. They also observe that the cutting force increases sharply as the needle approaches the membrane of the liver. O'Leary et al. [46] make further measurements with a similar experimental setup and conclude that the friction force consists of a Coulomb friction component, which does not depend on velocity, and a viscous friction component, which increases with the relative velocity of the tissue and the needle. Crouch et al. [17] optimize several simulation parameters to match the simulation result with their experimental data

to determine the coefficient of the viscous friction force. We include these forces in our simulator.

3.5 Remeshing

During simulation, the tissue node that corresponds to the needle tip moves in the material space when the needle penetrates or retracts. Eventually, the tetrahedra surrounding the tip node become badly shaped or inverted. To ensure good mesh quality, which is crucial for the robustness of the simulator, one needs to remesh the tissue. Paloc et al. [49] use Delaunay refinement [57] and a vertex decimation algorithm for hole filling [54] to handle surgical cutting. However, a node repositioning operation, which is necessary to make the needle share the tip node with the tissue, is not considered in their work. Klinger and Shewchuk [33] present an effective local mesh improvement method, which is successfully used for a dynamic simulation of elastoplastic solids by Wicke et al. [75]. The method could be adapted for remeshing, but it is too expensive for real-time applications. A local remeshing algorithm that considers node repositions and node additions is presented by Goksel et al. [25], but preserving the quality of the mesh is not considered in their work. Our new remeshing algorithm builds on this body of work by using additional remeshing operations (edge and tetrahedron splitting) and by choosing operations based on mesh quality, not on geometric heuristics.

Alternatively, different meshes can be tied together by binding constraints without remeshing, as Sifakis et al. [58] do. However, in real tissues, the deformation gradient around the needle shaft is discontinuous, with a near-singularity at the needle tip. Therefore, it is essential for accuracy that our volume mesh conform to the needle, and especially that it have a node at the needle tip. Non-conforming approaches distribute the needle forces onto nearby nodes, so the largest deformation does not coincide with the needle and the piecewise constant strains are inaccurate for elements that intersect the needle. While the bulk tissue behavior would be similar away from the needle, the needle path would be quite different.

Chapter 4

Simulation Overview

We present an overview of our simulator in this chapter. We first describe the requirements of the tissue and the needle simulation meshes. Then we describe how the simulator preceeds in each time step. Finally I describe implicit Newmark integration and derive the linear systems that relate forces to nodal accelerations, which must be computed in each time step.

4.1 Simulation Mesh

We model the tissue elasticity with constitutive equations discretized over a tetrahedral mesh by the Finite Element Method (FEM). The needle has a small diameter, so we model it as a 1D elastic rod, following DiMaio and Salcudean [21]. We denote the tetrahedral tissue mesh by $\hat{T}$. The needle is represented by a mesh $\widetilde{T}$ comprising a subset of the edges of $\hat{T}$, plus additional edges that represent the portion of the needle outside the tissue, as Figure 4.1 shows. We dynamically update the tetrahedral mesh so that it always conforms to the needle, as described in Section 9.1.

Throughout the simulation, we maintain for each node of $\hat{T}$ both a material position (recording the geometry of the undeformed mesh) and a world position (recording the deformed mesh). Let $\mathbf{u}_i^k$, $\mathbf{x}_i^k$, $\mathbf{v}_i^k$, $\mathbf{a}_i^k \in \mathbb{R}^3$ denote the material position, world position, velocity, and acceleration of the i^{th} node at time index k. We omit the node index to refer to a vector of properties for all the nodes. We omit the time index to refer to the current time. We use carets ($\hat{}$) to denote tissue properties, and tildes ($\tilde{}$) for needle properties.

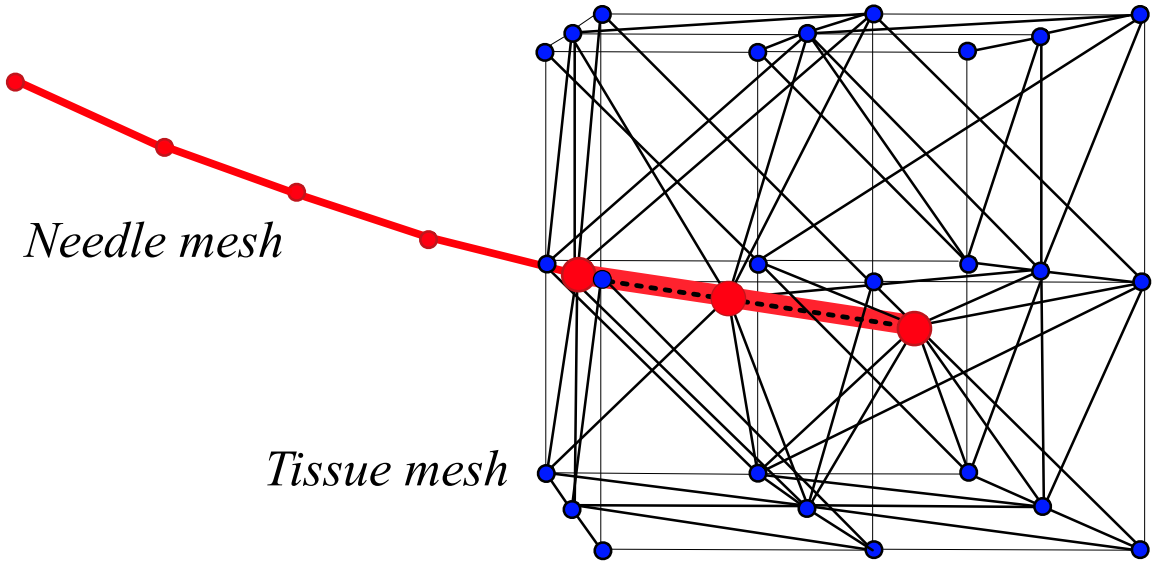

Figure 4.1: The needle mesh (red) and tissue mesh (black). The large red nodes and bold red edges belong to both meshes and couple the two objects.

4.2 Simulation Step

Algorithm 1 summarizes the execution of one simulation step. The simulation step begins by evaluating the internal forces and their Jacobians at the mesh nodes of the tissue and the needle. The Jacobians are used later in the solution step. Our choices of tissue and needle forces will be discussed in Chapter 5 and 6 respectively.

Algorithm 1 Needle simulation (one time step)

1: Compute tissue forces $\hat{\mathbf{F}}$, needle forces $\tilde{\mathbf{F}}$, Jacobians $\partial\mathbf{F}/\partial\mathbf{x}$ and $\partial\mathbf{F}/\partial\mathbf{v}$ for both tissue and needle (Chapter 5 and 6) and forces between tissue and needle (Chapter 7)
2: Solve the friction states and nodal accelerations that simultaneously satisfy the coupled system 8.1 and the inequalities 8.8, 8.9, and 8.10.
3: Update the positions and velocities of tissue and needle (Section 4.3)
4: **if** the needle tip is cutting or retracting **then**
5: Remesh around the tip (Section 9.1)
6: **end if**
7: Reparameterize the needle (Section 9.2)

The next step is to solve for the acceleration of each node in both meshes. Tissue and needle exert normal forces and various friction forces on each other, as discussed in Chapter 7. We model the friction along the needle shaft with a stick-slip friction model. Each node shared by the two meshes is in either a static or dynamic friction state. Static friction implies that the needle and tissue are moving in lockstep at the node; dynamic friction implies that they are sliding against each other. These friction states are not known in advance and must be solved for. We discuss how to solve for them in Chapter 8.

The needle tip also needs special care as it can either retract from the tissue or cut through the tissue. In these cases, the material coordinates of the needle tip and the corresponding tissue node move. This can cause tetrahedra near the tip to distort and become badly shaped and unsuitable for simulation. Hence, we perform local remeshing near the tip node to ensure that we have a good quality mesh, as discussed in Section 9.1. Finally, the needle dynamic friction nodes can slide away from the corresponding tissue nodes. To restore the needle and tissue nodes' conformity, we reposition the needle nodes to match the tissue nodes and transfer all the physical quantitities by interpolation, as discussed in Section 9.2.

4.3 Implicit Newmark Time Integration

Our simulator uses Newmark's method to specify how the velocity and position of a given tissue or needle node should be updated given the acceleration and Newton's law of motion, which relates the internal and the external forces to the nodal accelerations.

Let n be the number of nodes in the (tissue or needle) mesh. We integrate the node positions $\mathbf{x} \in \mathbb{R}^{3n}$ and velocities $\mathbf{v} \in \mathbb{R}^{3n}$ over time with Newmark's method,

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \triangle t \mathbf{v}^k + \triangle t^2 \left(\left(\frac{1}{2} - \beta \right) \mathbf{a}^k + \beta \mathbf{a}^{k+1} \right), \tag{4.1}$$

$$\mathbf{v}^{k+1} = \mathbf{v}^k + \triangle t \left((1-\gamma)\, \mathbf{a}^k + \gamma \mathbf{a}^{k+1} \right), \tag{4.2}$$

where $\triangle t$ is the time step, $0 \leq \beta \leq 0.5$, and $0 \leq \gamma \leq 1$. (All our simulation results use $\beta = 0.25$ and $\gamma = 0.5$.) We obtain the accelerations $\mathbf{a}^{k+1} \in \mathbb{R}^{3n}$ by solving Newton's equation of motion

$$\mathbf{F}(\mathbf{x}^{k+1}, \mathbf{v}^{k+1}) = \mathbf{M}\mathbf{a}^{k+1}, \tag{4.3}$$

where $\mathbf{M} \in \mathbb{R}^{3n\times 3n}$ is the mass matrix and $\mathbf{F}(\cdot) \in \mathbb{R}^{3n}$ is the sum of all internal forces such as stiffness and damping forces and external forces such as gravity. Because Equation (4.3) is nonlinear, we linearize it with one Newton–Raphson iteration; *i.e.* by solving

$$\mathbf{F}(\mathbf{x}^k, \mathbf{v}^k) + \frac{\partial \mathbf{F}}{\partial \mathbf{x}}(\mathbf{x}^{k+1} - \mathbf{x}^k) + \frac{\partial \mathbf{F}}{\partial \mathbf{v}}(\mathbf{v}^{k+1} - \mathbf{v}^k) \approx \mathbf{M}\mathbf{a}^{k+1}, \tag{4.4}$$

where $\partial\mathbf{F}/\partial\mathbf{x}$, $\partial\mathbf{F}/\partial\mathbf{v} \in \mathbb{R}^{3n\times 3n}$ are the Jacobian matrices of force with respect to position and velocity, evaluated at $(\mathbf{x}^k, \mathbf{v}^k)$.

Ignoring for now the coupling between the needle and the tissue, we substitute (4.1) and (4.2) into (4.4):

$$\mathbf{F}(\mathbf{x}^k, \mathbf{v}^k) + \frac{\partial \mathbf{F}}{\partial \mathbf{x}}\left(\triangle t \mathbf{v}^k + \triangle t^2\left(\left(\frac{1}{2} - \beta\right)\mathbf{a}^k + \beta \mathbf{a}^{k+1}\right)\right) + \frac{\partial \mathbf{F}}{\partial \mathbf{v}}\left(\triangle t\left((1-\gamma)\,\mathbf{a}^k + \gamma \mathbf{a}^{k+1}\right)\right) = \mathbf{M}\mathbf{a}^{k+1}, \tag{4.5}$$

$$\left(\mathbf{M} - \beta\triangle t^2 \frac{\partial \mathbf{F}}{\partial \mathbf{x}} - \gamma \triangle t \frac{\partial \mathbf{F}}{\partial \mathbf{v}}\right)\mathbf{a}^{k+1} = \mathbf{F}(\mathbf{x}^k, \mathbf{v}^k) + \frac{\partial \mathbf{F}}{\partial \mathbf{x}}\left(\triangle t \mathbf{v}^k + \triangle t^2\left(\left(\frac{1}{2} - \beta\right)\mathbf{a}^k\right)\right) + \frac{\partial \mathbf{F}}{\partial \mathbf{v}}\left(\triangle t\left((1-\gamma)\,\mathbf{a}^k\right)\right), \tag{4.6}$$

$$\mathbf{A}\mathbf{a}^{k+1} = \mathbf{b}, \tag{4.7}$$

where $\mathbf{A} = \mathbf{M} - \beta\triangle t^2 \frac{\partial \mathbf{F}}{\partial \mathbf{x}} - \gamma \triangle t \frac{\partial \mathbf{F}}{\partial \mathbf{v}}$ and $\mathbf{b} = \mathbf{F}(\mathbf{x}^k, \mathbf{v}^k) + \frac{\partial \mathbf{F}}{\partial \mathbf{x}}\left(\triangle t \mathbf{v}^k + \triangle t^2\left(\left(\frac{1}{2} - \beta\right)\mathbf{a}^k\right)\right) + \frac{\partial \mathbf{F}}{\partial \mathbf{v}}\left(\triangle t\left((1-\gamma)\,\mathbf{a}^k\right)\right)$.
We can obtain the linear system for the tissue and needle nodes' accelerations, respectively, as

$$\hat{\mathbf{A}}\hat{\mathbf{a}}^{k+1} = \hat{\mathbf{b}}, \tag{4.8}$$

$$\tilde{\mathbf{A}}\tilde{\mathbf{a}}^{*k+1} = \tilde{\mathbf{b}}. \tag{4.9}$$

The asterisk indicates that $\tilde{\mathbf{a}}^{*k+1}$ is a temporary quantity, because we still need to perform remeshing as explained in Section 9.2. Having solved for $\hat{\mathbf{a}}^{k+1}$ and $\tilde{\mathbf{a}}^{*k+1}$, we obtain $\hat{\mathbf{x}}^{k+1}$, $\hat{\mathbf{v}}^{k+1}$, $\tilde{\mathbf{x}}^{*k+1}$, and $\tilde{\mathbf{v}}^{*k+1}$ from Equations (4.1) and (4.2).

The sparsity of $\hat{\mathbf{A}}$ is unstructured. Each non-zero entry corresponds to a pair of nodes in the tissue mesh that are connected by an edge. $\tilde{\mathbf{A}}$ has bandwidth 5 (measured in 3×3 blocks) because each needle node has nonzero entries for the two nodes before and after it on the needle. We assemble both matrices and store $\hat{\mathbf{A}}$ with a block compressed sparse row format and $\tilde{\mathbf{A}}$ with a block banded format.

In the next two chapters, we will discuss how our simulator computes the tissue forces ($\hat{\mathbf{F}}$) and their Jacobians ($\frac{\partial \hat{\mathbf{F}}}{\partial \hat{\mathbf{x}}}$ and $\frac{\partial \hat{\mathbf{F}}}{\partial \hat{\mathbf{v}}}$) and the needle forces ($\tilde{\mathbf{F}}$) and their Jacobians ($\frac{\partial \tilde{\mathbf{F}}}{\partial \tilde{\mathbf{x}}}$ and $\frac{\partial \tilde{\mathbf{F}}}{\partial \tilde{\mathbf{v}}}$). We then use them to compute $\hat{\mathbf{b}}$ and $\tilde{\mathbf{b}}$ and assemble $\hat{\mathbf{A}}$ and $\tilde{\mathbf{A}}$.

Chapter 5

Tissue Force Computation

This chapter describes how the tissue forces $\hat{\mathbf{F}}$ are computed in the simulation. We first define the deformation gradient and then explain how it can be used to compute stress. We then describe a constitutive model for computing elastic forces and damping forces. Finally, we discuss how to compute Jacobians of the forces.

5.1 Deformation Gradient

The material position and world position of a point inside a tissue are related by a mapping $\phi(\mathbf{u}) = \mathbf{x}$. The deformation gradient is denoted by $\mathbf{D}(\mathbf{u}) = \frac{\partial \phi}{\partial \mathbf{u}}$. The tissue mesh $\hat{T}$ induces a piecewise linear approximation of ϕ; ϕ is approximated as a linear function over each tetrahedron, as shown in Figure 5.1. This approximation has a constant deformation gradient in each tetrahedron, which can be computed as $\mathbf{D} = [\mathbf{x}_1 - \mathbf{x}_0, \mathbf{x}_2 - \mathbf{x}_0, \mathbf{x}_3 - \mathbf{x}_0][\mathbf{u}_1 - \mathbf{u}_0, \mathbf{u}_2 - \mathbf{u}_0, \mathbf{u}_3 - \mathbf{u}_0]^{-1}$, where $\mathbf{u}_i$ and $\mathbf{x}_i$ denote the material coordinate and world coodinate of node i of the tetrahedron respectively.

5.2 First Piola–Kirchhoff Stress

The stress at a given point of an elastic material is determined by the deformation gradient. Our simulator uses the first Piola–Kirchhoff stress $\mathbf{P}$, which maps the area- weighted normal in material

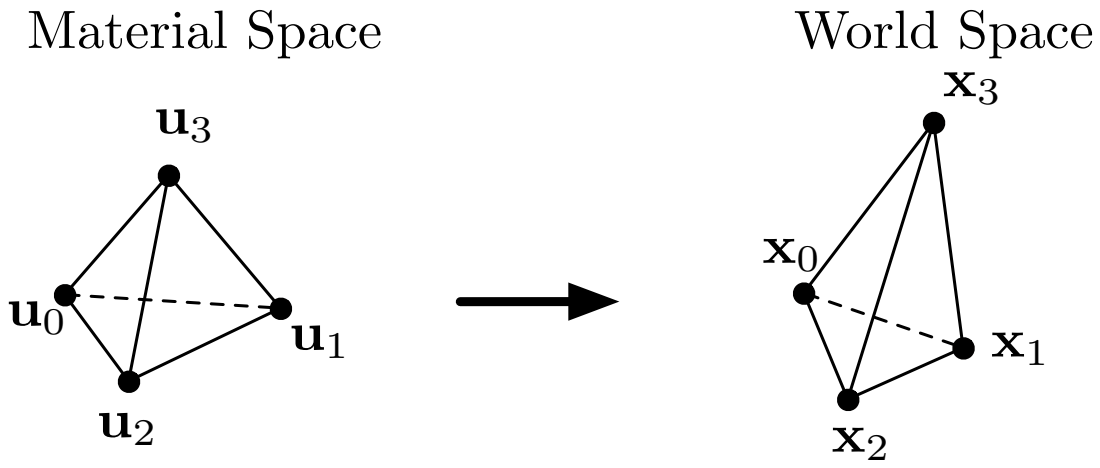

Figure 5.1: A locally linear mapping ϕ. A material space position $\mathbf{u}$ is mapped to the world space position$\mathbf{x}$ with $\mathbf{x} = \mathbf{x}_0 + \mathbf{D}(\mathbf{u} - \mathbf{u}_0)$, where $\mathbf{D} = [\mathbf{x}_1 - \mathbf{x}_0, \mathbf{x}_2 - \mathbf{x}_0, \mathbf{x}_3 - \mathbf{x}_0][\mathbf{u}_1 - \mathbf{u}_0, \mathbf{u}_2 - \mathbf{u}_0, \mathbf{u}_3 - \mathbf{u}_0]^{-1}$.

space to the traction in world space. Once $\mathbf{P}$ is determined inside a tetrahedron, it is used to compute the force $\mathbf{F}_i$ on a node i incident to the tetrahedron as $\mathbf{F}_i = -\mathbf{P}(a_1\mathbf{n}_1 + a_2\mathbf{n}_2 + a_3\mathbf{n}_3)$, where $a_j\mathbf{n}_j$ is the area-weighted normal in material space of face j incident to the node. Due to the law of conservation of momentum, $\mathbf{F}_0 = -(\mathbf{F}_1 + \mathbf{F}_2 + \mathbf{F}_3)$. We loop through each tetrahedron, compute its force contribution to the adjacent four nodes, and sum the forces at each of the nodes.

5.3 Rotation-Invariant and Isotropic Constitutive Model

Rigid transformations do not change the behavior of an elastic rod. This fact can be stated mathematically as $\mathbf{P}(\mathbf{UD}) = \mathbf{UP}(\mathbf{D})$ where $\mathbf{P}$ is the function that maps $\mathbf{D}$ to the Piola–Kirchhoff stress and $\mathbf{U}$ is an arbitrary rotation matrix. Moreover, we use an isotropic constitutive model for the tissue, which implies that the tissue is equally stretchable in all directions. Mathematically, it can be stated as $\mathbf{P}(\mathbf{DV}^\mathsf{T}) = \mathbf{P}(\mathbf{D})\mathbf{V}^\mathsf{T}$, where $\mathbf{V}$ is a rotation matrix. Therefore, for an anisotropic material, we can compute a singular value decomposition (SVD) of $\mathbf{D} = \mathbf{U}\hat{\mathbf{D}}\mathbf{V}^\mathsf{T}$ and express $\mathbf{P}$ as $\mathbf{P}(\mathbf{D}) = \mathbf{U}\hat{\mathbf{P}}(\hat{\mathbf{D}})\mathbf{V}^\mathsf{T}$, where $\hat{\mathbf{P}}$ is a diagonal matrix function of the diagonal deformation gradient $\hat{\mathbf{D}}$.

The $\mathbf{U}$ and $\mathbf{V}$ must be chosen so that $\det \mathbf{U} = \det \mathbf{V} = \mathbf{1}$ so that $\mathbf{U}$ and $\mathbf{V}$ are rotation matrices.

When the tetrahedron is flat ($\det \mathbf{D} \sim 0$) or inverted ($\det \mathbf{D} < 0$) in world space, special care is needed in choosing $\mathbf{U}$, $\mathbf{V}$ and the singular values. Irving et al. [29] use the rule of thumb that the resulting force must act to un-invert the tetrahedron. The linear constitutive model relates $\hat{\mathbf{P}}$ to $\hat{\mathbf{D}}$ as

$$\hat{\mathbf{P}} = 2\mu(\hat{\mathbf{D}} - \mathbf{I}) + \lambda \text{tr}(\hat{\mathbf{D}} - \mathbf{I}), \tag{5.1}$$

where μ and λ are the Lamé Constants. We use Equation 5.1 to compute $\hat{\mathbf{P}}$, then compute $\mathbf{P}$ and finally obtain $\mathbf{F}$.

5.4 Damping Force

The damping force is computed with the velocity gradient, $\dot{\mathbf{D}} = \mathbf{U}\frac{\partial \mathbf{v}}{\partial \mathbf{u}}\mathbf{V}^{\mathsf{T}}$, where $\frac{\partial \mathbf{v}}{\partial \mathbf{u}} = [\mathbf{v}_1 - \mathbf{v}_0, \mathbf{v}_2 - \mathbf{v}_0, \mathbf{v}_3 - \mathbf{v}_0][\mathbf{u}_1 - \mathbf{u}_0, \mathbf{u}_2 - \mathbf{u}_0, \mathbf{u}_3 - \mathbf{u}_0]^{-1}$ and $\mathbf{v}_i$ is the velocity of the node i of the tetrahedron. We first form the stress rate $\dot{\mathbf{P}} = \phi(\dot{\mathbf{D}} + \dot{\mathbf{D}}^{\mathsf{T}}) + \psi \text{tr}(\dot{\mathbf{D}})\mathbf{I}$. $\dot{\mathbf{P}}$ can then be multiplied with the area weighted normal to obtain the damping force.

5.5 Jacobian Computation

To evaluate the Jacobians of the tissue forces, we differentiate the force expressions with respect to $\mathbf{x}$ and $\mathbf{v}$. We hold the SVD rotation matrices constant during the differentiation, following Irving et al. [29], so we can obtain the closed-form expressions for the Jacobians.

As an alternative to the method discussed in this chapter, one could use the co-rotational approach presented by Muller and Gross [42] to compute the tissue force and jacobian. Other components of the simulator described in this thesis can be used without any change.

Chapter 6

Needle Force Computation

In this chapter, we describe how to compute the needle forces, $\tilde{\mathbf{F}}$. Our bending and twisting force computations follow Bergou et al. [11]. The needle is inextensible, but instead of enforcing inextensibility by projection, we use the method of Spillmann and Teschner [61] that applies compensatory stretching forces. We summarize our combined approach in this chapter.

6.1 Energies of an Elastic Rod

We model the needle as a one-dimensional elastic rod, which can be described by a curve Γ in 3D space (see Figure 6.1 left). Let $\mathbf{x}(s)$, $\mathbf{v}(s) \in \Re^3$ be the position and velocity along the curve Γ parameterized by the arclength from the base of the undeformed rod. In addition to the position and velocity, we define an orthogonal material frame $\{\mathbf{t}(s), \mathbf{m}^1(s), \mathbf{m}^2(s)\}$ along the curve, where $\mathbf{t}(s) = \frac{\mathbf{x}(s)}{||\mathbf{x}(s)||}$, and $\mathbf{m}^1(s)$ and $\mathbf{m}^2(s)$ store the local orientation of the elastic rod's material. We denote the derivative with respect to s as prime ($'$). Throughout this chapter, we refer to $\kappa = \mathbf{t}'$ as the curvature vector.

6.1.1 Potential Energy

The Kirchhoff theory of elastic rods can be used to compute the potential energy stored in the curve Γ, which consists of three terms: stretching energy, bending energy and twisting energy.

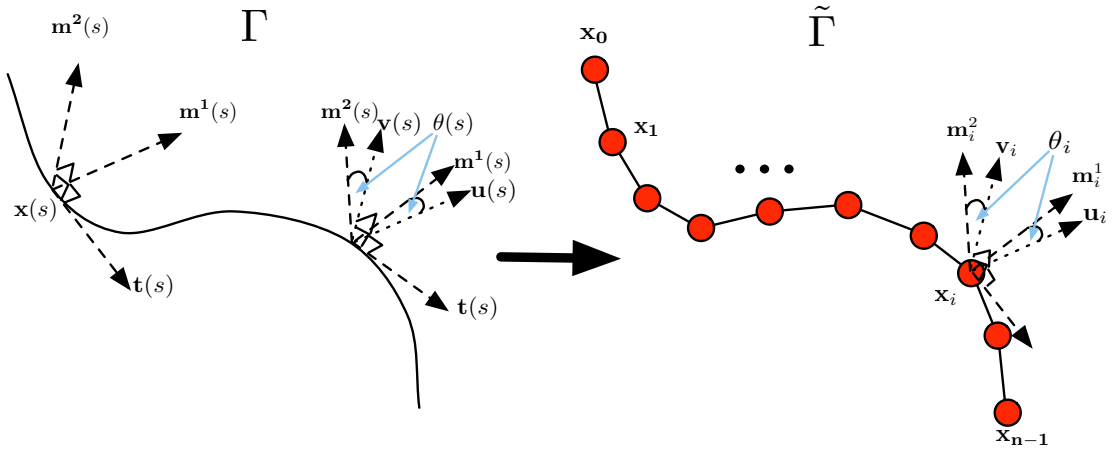

Figure 6.1: An elastic rod represented by a curve Γ can be discretized into a piecewise linear curve $\tilde{\Gamma}$. Γ can be represented by position $\mathbf{x}(s)$ and material frame $\{\mathbf{t}(s), \mathbf{m}^1(s), \mathbf{m}^2(s)\}$. The curve has a natural twist-free Bishop frame $\{\mathbf{t}(s), \mathbf{u}(s), \mathbf{v}(s)\}$. The material frame can alternatively be represented simply as $\theta(s)$, the angle to rotate the material frame onto the Bishop frame about the axis $\mathbf{t}(s)$. $\tilde{\Gamma}$ can be represented by $\mathbf{x}_i$ and material frame $\{\mathbf{t}_i, \mathbf{m}_i^1, \mathbf{m}_i^2\}$. Its Bishop frame is $\{\mathbf{t}_i, \mathbf{u}_i, \mathbf{v}_i\}$. Similar to the continuous counterpart, the material frame forms an angle of θ_i radians with the Bishop frame.

Stretching Energy

The stretching energy E_{stretch} is

$$E_{\text{stretch}} = \frac{1}{2}\int_{\Gamma} K_s \cdot (||\mathbf{x}'|| - 1)\, ds, \tag{6.1}$$

where K_s is the stretching stiffness, which can be computed from Young's modulus, E_s, with $K_s = E_s \pi r^2$, where r is the radius of the rod. This energy measures how much the rod stretches or compresses from the rest length.

Bending Energy

The bending energy E_{bend} is

$$E_{\text{bend}} = \frac{1}{2}\int_{\Gamma} K_b ||\kappa||^2\, ds, \tag{6.2}$$

where K_b is the bending stiffness. This energy measures how much the rod bends, i.e. the more the rod bends the larger its curvature becomes.

Twisting Energy

The twisting energy is

$$E_{\text{twist}} = \frac{1}{2}\int_{\Gamma} K_t m^2\, ds, \tag{6.3}$$

where K_t is the twisting stiffness and $m = \mathbf{m}^{1'} \cdot \mathbf{m}^2$, which measures how much the orthogonal frame twists along the curve. We can see that E_{twist} only depends on the dot product. This motivates Bergou et al. [11] to seek an equivalent expression in term of reduced coordinates.

Given a curve in 3D, one can find a twist-free frame called a *Bishop frame* $\{\mathbf{t}(s), \mathbf{u}(s), \mathbf{v}(s)\}$, such that the frame has zero twist at all points, i.e. $\mathbf{u}' \cdot \mathbf{v} = -\mathbf{v}' \cdot \mathbf{u} = 0$. Assigning a frame to one point on the curve will uniquely determine the Bishop frame throughout the curve. In our simulator, we assign the frame at the needle base. The evolution of the orthogonal frame while traversing the

curve can be described as

$$\mathbf{t}' = \Omega \times \mathbf{t}, \tag{6.4}$$

$$\mathbf{u}' = \Omega \times \mathbf{u}, \tag{6.5}$$

$$\mathbf{v}' = \Omega \times \mathbf{v}, \tag{6.6}$$

where $\Omega(s)$ is the *Darboux vector* of the frame, which signifies the axis and the speed of rotation of the orthogonal frame along the curve. For the Bishop frame, since $\mathbf{u}' \cdot \mathbf{v} = 0$, the first and the second equations imply that $\Omega = \mathbf{t} \times \mathbf{t}'$ which is the binormal of the curve.

Given a vector $\mathbf{w}$ at a given point on the curve, one can transport it to another point on the curve by integrating the ODE $\mathbf{x}' = \kappa \mathbf{b} \times \mathbf{x}$. This is referred to by Bergou et al. [11] as *parallel transport*. Infinitesimally, it can be interpreted as a rotation about the binormal ($\mathbf{t} \times \mathbf{t}'$) of the curve, which we will make use of in the discrete case.

The twist-free Bishop frame allows one to represent the twist of the curve using a simple parameterization. Let $\theta(s)$ be the scalar function that measures the rotation about the tangential axis of the material frame relative to the Bishop frame (see Figure 6.1, left). We have

$$\mathbf{m}^1 = \cos\theta \cdot \mathbf{u} + \sin\theta \cdot \mathbf{v}, \tag{6.7}$$

$$\mathbf{m}^2 = -\sin\theta \cdot \mathbf{u} + \cos\theta \cdot \mathbf{v}. \tag{6.8}$$

Since $m = \mathbf{m}^{1'} \cdot \mathbf{m}^2$ and $\mathbf{u}' \cdot \mathbf{v} = 0$, we have $m(s) = \theta'(s)$. Therefore, we express the twisting energy as

$$E_{\text{twist}} = \frac{1}{2} \int_\Gamma K_t(\theta')^2 \, ds. \tag{6.9}$$

6.1.2 Dissipation Energy

We also model the dissipation of energy due to internal friction and the viscoelastic effect of the rod following Spillmann and Teschner [61]. The internal friction damps away the relative translational velocity of the rod, $\mathbf{v}^{\text{rel}} = \frac{1}{||\mathbf{x}'||^2}(\mathbf{v}' \cdot \mathbf{x}')\mathbf{x}'$. The dissipation energy is

$$D = \frac{1}{2} \int_\Gamma K_d ||\mathbf{v}^{\text{rel}}||^2 \, ds, \tag{6.10}$$

where K_d is the damping coefficient.

6.2 Discrete Energies

We discretize the curve Γ into a series of vertices $\mathbf{x}_i$, $0 \leq i < n$, where n is the number of nodes (see Figure 6.1, right), with velocity $\mathbf{v}_i$. Define a sequence of edges $\mathbf{e}_i = \mathbf{x}_{i+1} - \mathbf{x}_i$ for $0 \leq i < n-1$ and discrete material frames for each edge, $\{\mathbf{t}_i, \mathbf{m}_i^1, \mathbf{m}_i^2\}$, where $\mathbf{t}_i = \frac{\mathbf{e}_i}{||\mathbf{e}_i||}$. We express the energies using these discrete variables as shown below. After that, we can take appropriate derivatives to obtain the expressions for the forces on the nodes.

Let the length of edge i be $l_i = ||\mathbf{e}_i||$, and the average length of the two edges adjacent to a node i be $h_i = \frac{l_{i-1}+l_i}{2}$. We denotes the quantities associated with the elastic rod in its rest configuration with a bar ($\bar{}$). For example, $\bar{l}_i$ denotes the rest length of edge i.

6.2.1 Discrete Potential Energy

We derive the expressions for the discrete potential energies of the Kirchoff rod by assuming that the integrand is constant in each edge. The integral then becomes the sum over all edges.

Discrete Stretching Energy

The discrete stretching energy is

$$E_{\text{stretch}}^{\text{dis}} = \frac{1}{2} \sum_{i=0}^{n-2} K_s \bar{l}_i \left(\frac{l_i}{\bar{l}_i} - 1 \right)^2. \tag{6.11}$$

Discrete Bending Energy

To compute the discrete bending energy, we need to obtain an estimate for the curvature. Following Bergou et al. [11], we define $(\kappa \mathbf{b})_i = \frac{2\mathbf{e}_{i-1} \times \mathbf{e}_i}{l_{i-1} l_i + \mathbf{e}_{i-1} \cdot \mathbf{e}_i}$. We can then express the discrete bending energy as

$$E_{\text{bend}}^{\text{dis}} = \frac{1}{2} \sum_{i=0}^{n-2} K_b \left\| \frac{(\kappa \mathbf{b})_i}{\bar{h}_i} \right\|^2 \bar{h}_i = \sum_{i=0}^{n-2} 2K_b \frac{\|(\kappa \mathbf{b})_i\|^2}{\bar{h}_i}. \tag{6.12}$$

Discrete Twisting Energy

To formulate the discrete twisting energy, we first define the concept of discrete parallel transport and the Bishop frame, so that we can express the twist in term of the angle of deviation from the twist-free frame, similar to the continuous case.

We can define the parallel transport operator that transports the vector residing at a node $i-1$ to node i as a rotation $\mathbf{P}_i$ about $(\kappa \mathbf{b})_i$ where $\mathbf{P}_i$ satisfies

$$\mathbf{P}_i(\mathbf{t}_{i-1}) = \mathbf{t}_i, \tag{6.13}$$

$$\mathbf{P}_i(\mathbf{t}_{i-1} \times \mathbf{t}_i) = \mathbf{t}_{i-1} \times \mathbf{t}_i. \tag{6.14}$$

The Bishop frame of the discrete curve can then be defined by transporting a unit vector $\mathbf{u}^0$ orthogonal to $\mathbf{t}^0$ by using $\mathbf{P}_i$'s operators, namely,

$$\mathbf{u}_i = \mathbf{P}_i(\mathbf{u}_{i-1}), \tag{6.15}$$

$$\mathbf{v}_i = \mathbf{t}_i \times \mathbf{u}_i. \tag{6.16}$$

We need to maintain that $\mathbf{u}_0 \perp \mathbf{t_0}$ throughout the simulation. However, after a simulation step, $\mathbf{t}_0$ will change (unless the first needle segment simply translates) and $\mathbf{u}_0$ will no longer be perpendicular to $\mathbf{t}_0$. We can update $\mathbf{u}_0$ by parallel transporting it in time with the rotation operator that will rotate the old $\mathbf{t}_0$ to the new $\mathbf{t}_0$.

Let θ_i denote the angle needed to rotate the Bishop frame to the discrete material frame at node i. Similar to the continuous case, we have

$$\mathbf{m}_i^1 = \cos\theta_i \cdot \mathbf{u}_i + \sin\theta_i \cdot \mathbf{v}_i, \tag{6.17}$$

$$\mathbf{m}_i^2 = -\sin\theta_i \cdot \mathbf{u}_i + \cos\theta_i \cdot \mathbf{v}_i. \tag{6.18}$$

The discrete twisting energy is

$$E_{\text{twist}}^{\text{dis}} = \frac{1}{2} \sum_{i=1}^{n-2} K_t \left(\frac{\theta_i - \theta_{i-1}}{\bar{l}_i} \right)^2 \bar{l}_i. \tag{6.19}$$

6.2.2 Discrete Dissipation Energy

The discrete dissipation energy is

$$D^{\text{dis}} = \frac{1}{2}\sum_{i=0}^{n-2} K_d ||\mathbf{v}_i^{\text{rel}}||^2 \bar{l}_i, \tag{6.20}$$

where $\mathbf{v}_i^{\text{rel}}$ is the relative tangential velocity

$$\begin{aligned} \mathbf{v}_i^{\text{rel}} &= \frac{1}{||\mathbf{x}_i'||^2}(\mathbf{v}_i' \cdot \mathbf{x}'_i)\mathbf{x}_\mathbf{i}' & (6.21)\\ &= \frac{1}{l_i^3}(\mathbf{x}_{i+1} - \mathbf{x}_i)((\mathbf{v}_{i+1} - \mathbf{v}_i) \cdot (\mathbf{x}_{i+1} - \mathbf{x}_i)) & (6.22) \end{aligned}$$

assuming that $||\mathbf{x}_i'||^2 \approx 1$, which is valid because the needle length only barely changes during simulation.

6.3 Force and Jacobian Computation

We obtain the force $\tilde{\mathbf{F}}$ on the needle nodes by computing the derivatives of the energies as

$$\tilde{\mathbf{F}}_i = -\frac{dE^{\text{dis}}}{d\mathbf{x}_i} - \frac{dD^{\text{dis}}}{d\mathbf{x}_i} \tag{6.23}$$

where $E^{\text{dis}} = E^{\text{dis}}_{\text{stretch}} + E^{\text{dis}}_{\text{bend}} + E^{\text{dis}}_{\text{twist}}$. The total derivatives can be computed in a straightforward manner, except for the twisting term, as θ implicitly depends on $\mathbf{x}$. Moreover, assuming that the twist wave propagates at infinite speed in the rod, one can compute θ_i at each time step by minimizing E^{dis} subject to the boundary conditions of the rod. We refer the reader to Bergou et al. [11] for the details about these computations.

From the expression 6.23 for $\tilde{\mathbf{F}}$, we can compute the Jacobians $\frac{\partial \tilde{\mathbf{F}}}{\partial \mathbf{x}}$ and $\frac{\partial \tilde{\mathbf{F}}}{\partial \mathbf{v}}$ directly. One caveat is that the Jacobians of the twisting force is asymmetric. To avoid solving an asymmetric linear system, we set the contribution of the twisting force to Jacobians to zero. During the Newmark time integration, where we linearized the forces about the current state, setting the Jacobian to zero is equivalent to treating the twisting force explicitly because it will not depend on the next state.

Chapter 7

Forces Between Tissue and Needle

Various forces including the normal force, friction forces and the cutting force play key roles during needle insertion. These forces are applied to the needle and the tissue in equal magnitude but opposite directions. In this chapter, we will describe these forces in details and show examples of situations where they are prominent. Figure 7.1 shows the cutaway view of a simulated tissue. The checkerboard texture is used to show the internal deformation. The red region represents a stiffer material than the gray region. The needle is blue. Other figures throughout this chapter demonstrate various scenarios to demonstrate the role of the forces. They show the screenshots of the tissue and the needle during simulations.

7.1 Static and Dynamic Friction

When the needle is pushed or pulled, the relative motion of the tissue and the needle is initially resisted by the static friction force. The force grows larger as needed to resist the relative movement. However, this force can only reach a certain threshold before the needle and tissue start to slide relative to each other. Crouch et al. [17] observe this behavior and determine that the threshold is proportional to the length of the needle inside the tissue.

Once the needle and the tissue start to slide, the force is equal to the dynamic friction. O'Leary et al. [46] observe that this is roughly equal to the static friction threshold. This friction force remains at the threshold until the relative velocity of the tissue and the needle is zero. The static friction

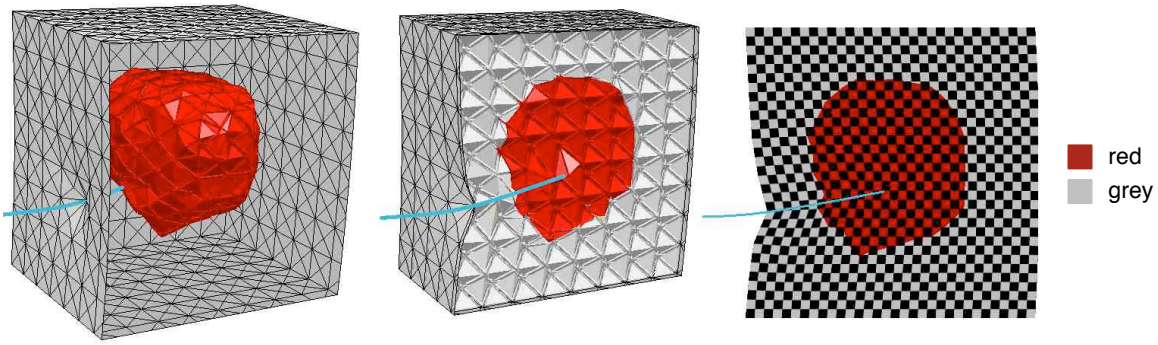

Figure 7.1: Left: The surface of two types of tissues are rendered during the middle of a simulation. The needle is shown as a thick line. Middle: Cutaway view of the tetrahedral mesh. Right: A thin slice of the mesh is shown with a checkerboard texture for visualizing the deformation.

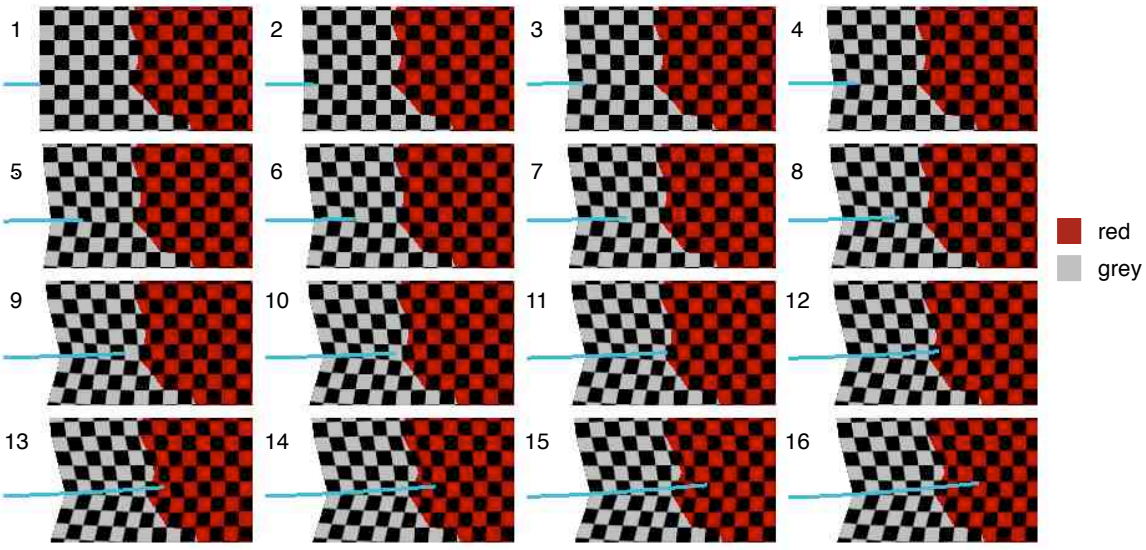

Figure 7.2: This figure demonstates a situation when the static friction threshold is high. The viscous force and the cutting force are zero. The needle is inserted at a constant speed. Because the needle tends to stick to the tissue, the tissue deformation around the needle shaft is large.

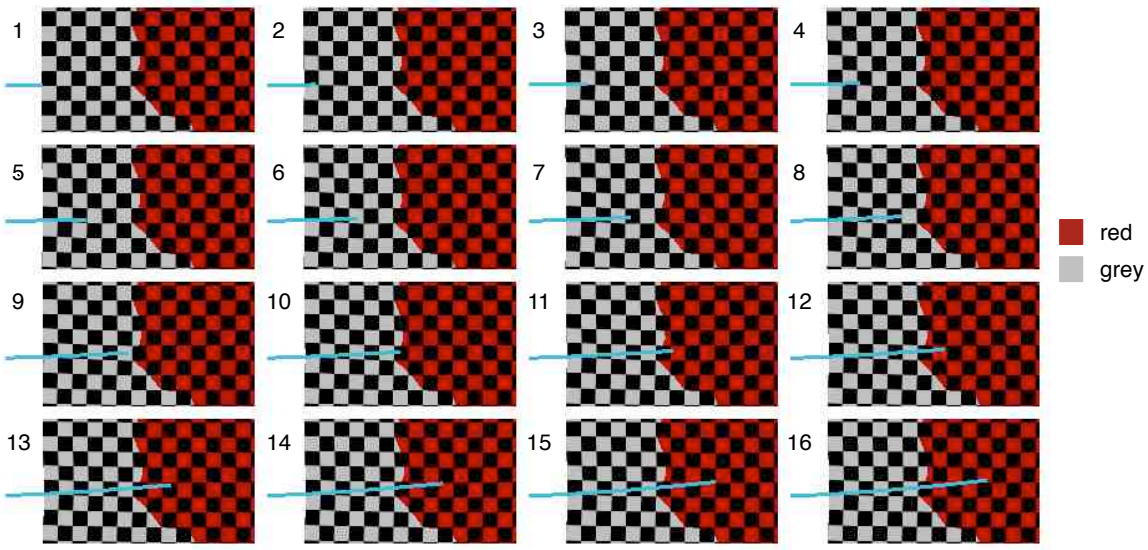

Figure 7.3: This figure demonstates a situation when the static friction threshold is low. The viscous force and the cutting force are zero. The needle is inserted at a constant speed. Because the needle can slide through tissue easily, the tissue deformation around the needle shaft is relatively small.

threshold f_i at node i is modeled in the simulator as $f_i = f_{\text{perlen}} h_i$, where f_{perlen} is the needle-tissue friction per unit length and h_i is the average length of the two edges adjoining the node i in the needle mesh. This threshold is used to check whether the coupled tissue and needle nodes should slip away from each other or not in Section 8.4. Figure 7.2 and 7.3 demonstrate situations when the static friction threshold is large and small, respectively.

7.2 Viscous Friction

Simone et al. [60] observed that the friction force increases linearly with the relative velocity of the tissue and the needle. This is the so-called *viscous effect*, which occurs because of the increase in the surface area of the tissue through which the needle passes over a unit time step as the relative velocity increases. A situation when the viscous friction plays a key role is shown in Figure 7.4. We model this friction by $f_i^{\text{vis}} = \eta h_i (\tilde{\mathbf{v}}_i - \hat{\mathbf{v}}_i) \cdot \mathbf{t}_i$, where $\mathbf{t}_i$ is the tangent vector at node i and η is the viscous friction coefficient. Recall that carets ($\hat{}$) denote tissue properties, and tildes ($\tilde{}$) denote needle properties. This force is added as an external force to the coupled needle and tissue nodes, with equal magnitude but opposite direction.

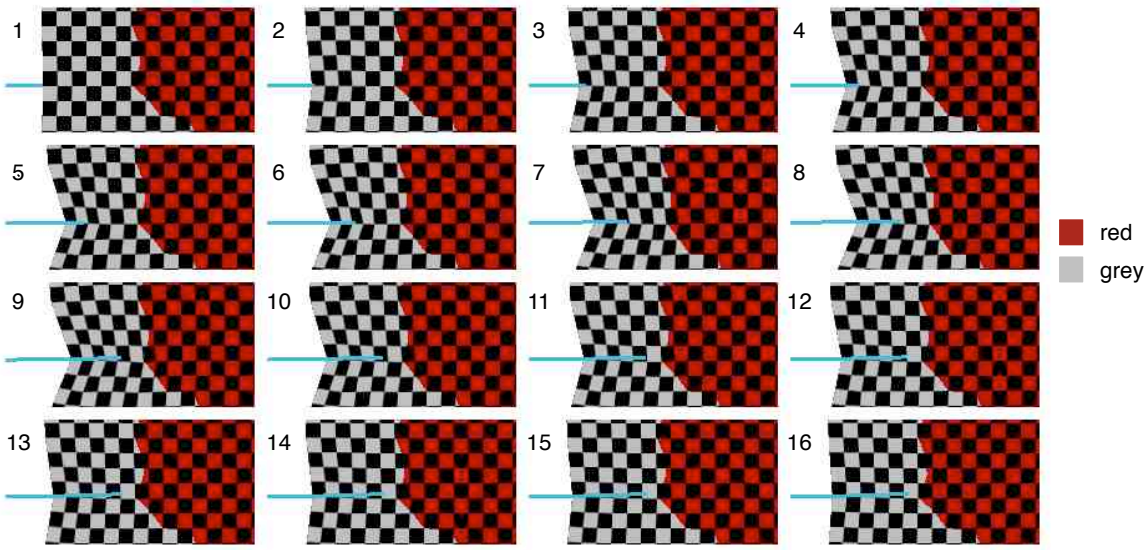

Figure 7.4: This figure demonstates the effect of the viscous friction. The static friction threshold is small in this case while the viscous friction coefficient is high. The cutting force is zero. The needle is inserted at a constant speed halfway and then held still. Large tissue deformation occurs during the first half of the simulation, when the relative velocity between tissue and needle is large. Once the relative velocity decreases, the needle slides through tissue easily due to the decreasing viscous friction.

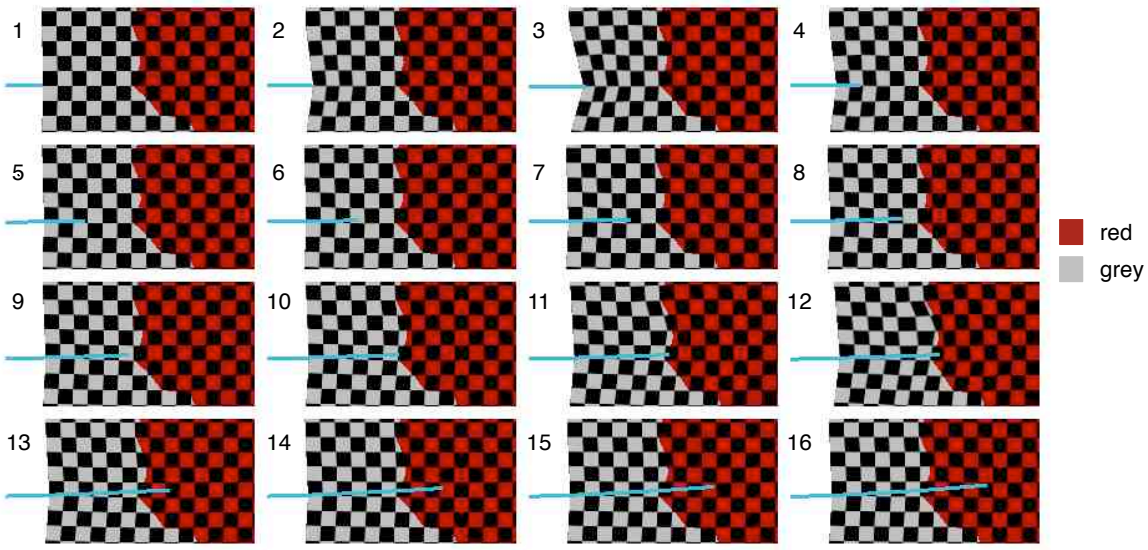

Figure 7.5: This figure demonstrates the capsule puncture effect. The static friction threshold is small. The viscous friction is zero. The cutting force is high when the needle attempts to penetrate through the membrane between air and the tissue and the membrane between two types of tissue. The needle is inserted with constant velocity. Notice how the tissue deformation is large before the initial penetration. Once the needle cuts through the capsule, the cutting force drops significantly, which allow the tissue to slide back along the needle. A similar effect appears during the second membrane penetration, albeit less noticable, because it is deeper inside the tissue.

7.3 Cutting Force and Capsule Puncture Effect

To penetrate through the tissue, the needle must overcome the resistance of the tissue. This resistance can be modeled by the *cutting force*. The force is applied at the tip of the needle. It is especially large when the needle attempts to penetrate through the membrane surrounding an organ. Simone et al. [60] observed this sharp force increase during the experiment of inserting a needle into a porcine liver and called it the *capsule puncture* effect.

Our simulator models the cutting force f_{cut} as a function that maps the material position of the needle tip to the magnitude of the force. To capture the capsule puncture effect, f_{cut} is at its peak inside the thin membrane separating two organs. Figure 7.5 shows the capsule puncture effect during a needle insertion. The cutting force is taken into account at the needle tip when the needle is penetrating, but not when the needle is retracting (see Section 8.4).

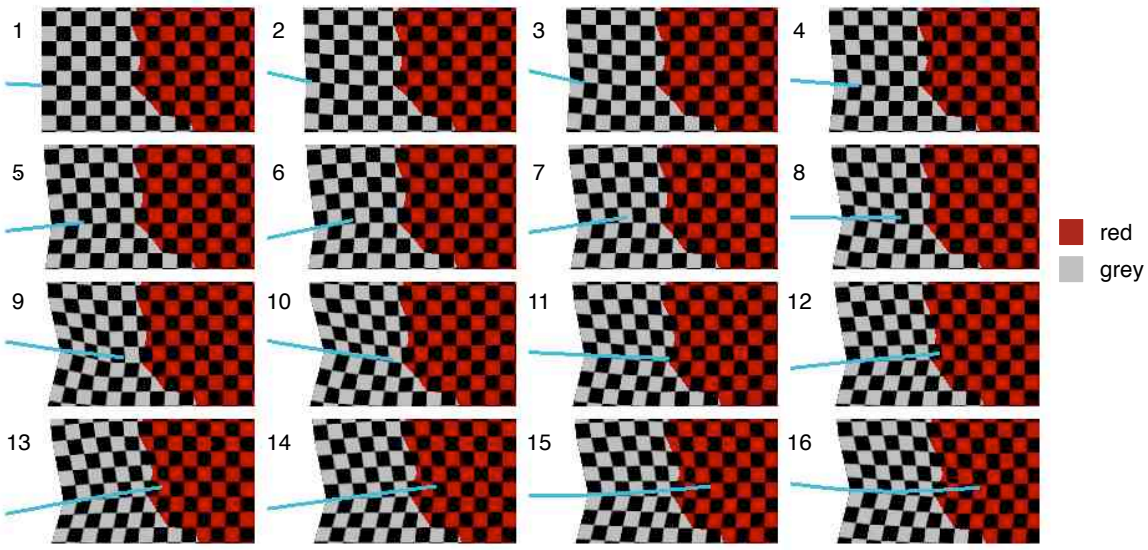

Figure 7.6: This figure demonstrates the lateral motion of the needle, which causes the tissue to deform in the same lateral direction. The needle is inserted at a constant velocity, while the needle base moves up and down in a periodic motion.

7.4 Normal Force

During the needle insertion into tissue, if one moves the needle base laterally, the needle bends and the tissue deforms in the same lateral direction due to the normal force that they exert on each other. In practice, the tissue is tough enough to prevent the needle from slicing through laterally. We made this non-slicing assumption in our simulator to simplify the remeshing process considerably. In the situation demonstrated in Figure 7.6, the needle is both pushed forward and moved laterally. The normal forces act as Lagrange multipliers to the coupled tissue and needle nodes and are solved for in each time step, as explained in Section 8.2.

Chapter 8

Tissue-Needle Coupling and Cutting

In this chapter, we formulate the coupling of the tissue and the needle through stick-slip friction and normal forces as a Linear Complementarity Problem (LCP) that must be solved in each simulation step. We first describe the friction states that each needle node can be in. We then formulate the coupled linear system of tissue and needle' accelerations by introducing the coupling forces as Lagrange multipliers, with an assumption that the friction states are known. Then we discuss how the linear system can be solved effectively with the Conjugate Gradient method (CG). We next discuss the additional inequality constraints that the stick-slip friction model impose on the coupled system. These inequalities lead to an LCP formulation whose solution determines the correct friction states. Finally, we discuss how we solve the LCP efficiently.

8.1 Friction States

Each node i of $\widetilde{T}$ has a friction state s_i which is one of FREE, STATIC, DYNAMIC$-$, or DYNAMIC$+$. The sign of dynamic friction indicates the direction in which the needle is sliding along the tissue at that node. The FREE nodes are not in the tissue. The other nodes, which are shared by the tissue mesh T and the needle mesh $\widetilde{T}$, are called *coupling nodes*. Assume that each coupling node has the same index in both meshes. Recall that $\mathbf{u}_i^k$, $\mathbf{x}_i^k$, $\mathbf{v}_i^k$, $\mathbf{a}_i^k \in \mathbb{R}^3$ denote the material position, world position, velocity, and acceleration of the i^{th} node at time index k. Carets ($\hat{}$) denote tissue properties, and tildes ($\tilde{}$) denote needle properties. Also, an asterisk indicates a temporary

quantity which requires reparameterization before we can continue to simulate the next time step, as explained in Section 9.2.

8.2 Coupled Linear System

If we know every state s_i, we can solve the coupled equations with Lagrange multipliers, introducing for each coupling node i an additional variable $\mathbf{c}_i \in \mathbb{R}^3$, the constraint force required to satisfy the stick-slip constraint. This force acts upon tissue and needle nodes i in equal magnitude but opposite directions. To accommodate dynamic friction and sliding of the needle, we express $\mathbf{c}_i$ in a local coordinate system of the needle, in which the first axis is the unit vector $\mathbf{t}_i$ that is tangential to the needle at node i. We approximate the tangent as $\mathbf{t}_i = \text{NORMALIZE}(\text{NORMALIZE}(\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_{i-1}) + \text{NORMALIZE}(\tilde{\mathbf{x}}_{i+1} - \tilde{\mathbf{x}}_i))$, where $\text{NORMALIZE}(\mathbf{u}) = \frac{\mathbf{u}}{\|\mathbf{u}\|}$. Let $\mathbf{R}_i \in \mathbb{R}^{3\times 3}$ be the rotation matrix that transforms from local coordinates to world coordinates, so that $\mathbf{R}_i\mathbf{c}_i$ is the constraint force at the coupling node i in world coordinates. Therefore, the first component of $\mathbf{c}_i$ is the tangential force. For a DYNAMIC node, we set the first column of $\mathbf{R}_i$ to zero, thus ignoring the tangential constraint force in direction $\mathbf{t}_i$. The second and third components of $\mathbf{c}_i$ represent the normal force that the needle and the tissue exert on each other.

The coupled system is

$$\begin{bmatrix} \hat{\mathbf{A}} & 0 & \hat{\mathbf{W}}\mathbf{R} \\ 0 & \tilde{\mathbf{A}} & -\tilde{\mathbf{W}}\mathbf{R} \\ (\hat{\mathbf{W}}\mathbf{R})^\mathsf{T} & -(\tilde{\mathbf{W}}\mathbf{R})^\mathsf{T} & \mathbf{Z} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{a}}^{k+1} \\ \tilde{\mathbf{a}}^{*k+1} \\ \mathbf{c} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{b}} + \hat{\mathbf{W}}\mathbf{R}\mathbf{d} \\ \tilde{\mathbf{b}} - \tilde{\mathbf{W}}\mathbf{R}\mathbf{d} \\ \mathbf{e} \end{bmatrix}. \tag{8.1}$$

See Section 8.5 for the derivation. The first two rows are Equations (4.8) and (4.9) augmented with the constraint forces. Here, $\mathbf{d}_i$ is $[0,0,0]^\mathsf{T}$ for a STATIC node and $[s_i f_i, 0, 0]^\mathsf{T}$ for a DYNAMIC node, s_i is 1 for DYNAMIC+ or -1 for DYNAMIC$-$, f_i is the magnitude of dynamic friction (possibly including a cutting force f_{cut} at the needle tip), and $\hat{\mathbf{W}}$ and $\tilde{\mathbf{W}}$ are 0–1 matrices that map coupling nodes to the tissue nodes and needle nodes, respectively. Thus, for a dynamic node i, the unknown tangential component of the constraint force $\mathbf{c}_i$ on the left-hand side is supplanted by the known friction $\pm f_i$ on the right, because the first column of $\mathbf{R}_i$ is zero.

The third row of Equation (8.1) constrains the coupling nodes to have the same positions in the tissue and needle meshes, except that a node in a dynamic friction state permits the needle to slide

tangentially relative to the tissue. If s_i is STATIC, we constrain $\hat{\mathbf{x}}_i$ and $\tilde{\mathbf{x}}_i$ to be identical. If it is DYNAMIC, we constrain them to agree in the directions orthogonal to $\mathbf{t}_i$. Thus, $\mathbf{Z}$ is a diagonal matrix in which a diagonal entry is 1 for the tangential component of a DYNAMIC node and 0 otherwise, and $\mathbf{e} = -(\hat{\mathbf{W}}\mathbf{R})^{\mathsf{T}}\left(\frac{1}{\triangle t\beta}\hat{\mathbf{v}}^k + \frac{\left(\frac{1}{2}-\beta\right)}{\beta}\hat{\mathbf{a}}^k\right) + (\tilde{\mathbf{W}}\mathbf{R})^{\mathsf{T}}\left(\frac{1}{\triangle t\beta}\tilde{\mathbf{v}}^{*k} + \frac{\left(\frac{1}{2}-\beta\right)}{\beta}\tilde{\mathbf{a}}^{*k}\right)$ is the right-hand side of the equation found by substituting Equation (4.1) into

$$(\hat{\mathbf{W}}\mathbf{R})^{\mathsf{T}}(\hat{\mathbf{x}}^{k+1} - \hat{\mathbf{x}}^k) - (\tilde{\mathbf{W}}\mathbf{R})^{\mathsf{T}}(\tilde{\mathbf{x}}^{*k+1} - \tilde{\mathbf{x}}^{*k}) = \mathbf{0} \tag{8.2}$$

and moving the terms that include $\hat{\mathbf{a}}^{k+1}$ and $\tilde{\mathbf{a}}^{*k+1}$ to the left-hand side as follows

$$(\hat{\mathbf{W}}\mathbf{R})^{\mathsf{T}}\left(\triangle t\hat{\mathbf{v}}^k + \triangle t^2\left(\left(\frac{1}{2}-\beta\right)\hat{\mathbf{a}}^k + \beta\hat{\mathbf{a}}^{k+1}\right)\right) - (\tilde{\mathbf{W}}\mathbf{R})^{\mathsf{T}}\left(\triangle t\tilde{\mathbf{v}}^{*k} + \triangle t^2\left(\left(\frac{1}{2}-\beta\right)\tilde{\mathbf{a}}^{*k} + \beta\tilde{\mathbf{a}}^{*k+1}\right)\right) = \mathbf{0}, \tag{8.3}$$

$$(\hat{\mathbf{W}}\mathbf{R})^{\mathsf{T}}\triangle t\hat{\mathbf{v}}^k + (\hat{\mathbf{W}}\mathbf{R})^{\mathsf{T}}\triangle t^2\left(\frac{1}{2}-\beta\right)\hat{\mathbf{a}}^k + (\hat{\mathbf{W}}\mathbf{R})^{\mathsf{T}}\triangle t^2\beta\hat{\mathbf{a}}^{k+1} - (\tilde{\mathbf{W}}\mathbf{R})^{\mathsf{T}}\triangle t\tilde{\mathbf{v}}^{*k} - (\tilde{\mathbf{W}}\mathbf{R})^{\mathsf{T}}\triangle t^2\left(\frac{1}{2}-\beta\right)\tilde{\mathbf{a}}^{*k} - (\tilde{\mathbf{W}}\mathbf{R})^{\mathsf{T}}\triangle t^2\beta\tilde{\mathbf{a}}^{*k+1} = \mathbf{0}, \tag{8.4}$$

$$(\hat{\mathbf{W}}\mathbf{R})^{\mathsf{T}}\triangle t^2\beta\hat{\mathbf{a}}^{k+1} - (\tilde{\mathbf{W}}\mathbf{R})^{\mathsf{T}}\triangle t^2\beta\tilde{\mathbf{a}}^{*k+1} = -(\hat{\mathbf{W}}\mathbf{R})^{\mathsf{T}}\triangle t\hat{\mathbf{v}}^k - (\hat{\mathbf{W}}\mathbf{R})^{\mathsf{T}}\triangle t^2\left(\frac{1}{2}-\beta\right)\hat{\mathbf{a}}^k + (\tilde{\mathbf{W}}\mathbf{R})^{\mathsf{T}}\triangle t\tilde{\mathbf{v}}^{*k} + (\tilde{\mathbf{W}}\mathbf{R})^{\mathsf{T}}\triangle t^2\left(\frac{1}{2}-\beta\right)\tilde{\mathbf{a}}^{*k}, \tag{8.5}$$

$$(\hat{\mathbf{W}}\mathbf{R})^{\mathsf{T}}\hat{\mathbf{a}}^{k+1} - (\tilde{\mathbf{W}}\mathbf{R})^{\mathsf{T}}\tilde{\mathbf{a}}^{*k+1} = -(\hat{\mathbf{W}}\mathbf{R})^{\mathsf{T}}\frac{1}{\triangle t\beta}\hat{\mathbf{v}}^k - (\hat{\mathbf{W}}\mathbf{R})^{\mathsf{T}}\frac{\left(\frac{1}{2}-\beta\right)}{\beta}\hat{\mathbf{a}}^k + (\tilde{\mathbf{W}}\mathbf{R})^{\mathsf{T}}\frac{1}{\triangle t\beta}\tilde{\mathbf{v}}^{*k} + (\tilde{\mathbf{W}}\mathbf{R})^{\mathsf{T}}\frac{\left(\frac{1}{2}-\beta\right)}{\beta}\tilde{\mathbf{a}}^{*k}, \tag{8.6}$$

$$(\hat{\mathbf{W}}\mathbf{R})^{\mathsf{T}}\hat{\mathbf{a}}^{k+1} - (\tilde{\mathbf{W}}\mathbf{R})^{\mathsf{T}}\tilde{\mathbf{a}}^{*k+1} = -(\hat{\mathbf{W}}\mathbf{R})^{\mathsf{T}}\left(\frac{1}{\triangle t\beta}\hat{\mathbf{v}}^k + \frac{\left(\frac{1}{2}-\beta\right)}{\beta}\hat{\mathbf{a}}^k\right) + (\tilde{\mathbf{W}}\mathbf{R})^{\mathsf{T}}\left(\frac{1}{\triangle t\beta}\tilde{\mathbf{v}}^{*k} + \frac{\left(\frac{1}{2}-\beta\right)}{\beta}\tilde{\mathbf{a}}^{*k}\right). \tag{8.7}$$

Note that the columns having a 1 on $\mathbf{Z}$'s diagonal are the same columns of $\mathbf{R}$ that we set to zero.

8.3 Solving the Coupled Linear System

The coupled system (8.1) is symmetric but indefinite. Indefinite systems usually cannot be solved by the conjugate gradient method (CG), and require algorithms like MINRES or SYMMLQ [48], which are about twice as expensive per iteration. Nevertheless, we find that CG effectively solves our system (8.1) to the desired tolerance in practice. Marcia [41] makes the same observation for linear systems similar to ours, and provides a partial explanation. He experiments with solving an indefinite but symmetric linear system of the form $\begin{bmatrix} \mathbf{A} & \mathbf{C} \\ \mathbf{C}^{\mathsf{T}} & \mathbf{0} \end{bmatrix}$ where $\mathbf{A}$ is a positive definite matrix and found that CG converges faster than MINRES and SYMMLQ, but with non-smooth convergence rate. He also proposes a fix to CG, by using a novel pivoting strategy in the matrix factorization, implicitly done in each iteration of CG. It requires only a small constant more flops per iteration than CG but result in much smoother convergence rate for a symmetric indefinite matrix. The method reduces to CG for a positive definite matrix. His approach can be applied to solve our linear system as well.

8.4 Linear Complementarity Problem (LCP) Formulation

Up to this point, we assume that all the states s_i are known. In practice, however, s_i are not known in advance (except the FREE ones). We must guess them, then guess again if we are wrong. We have guessed right if they satisfy the following constraints. For a STATIC node i other than the needle tip,

$$-f_i \leq [1\ 0\ 0]^{\mathsf{T}} \cdot \mathbf{c}_i \leq f_i, \tag{8.8}$$

where f_i is the static friction threshold, which experimentally is the same as the dynamic friction magnitude (see Section 7.1). For a STATIC needle tip i,

$$-(f_i + f_{\text{cut}}) \leq [1\ 0\ 0]^{\mathsf{T}} \cdot \mathbf{c}_i \leq f_i. \tag{8.9}$$

For a DYNAMIC node i (needle tip or not),

$$s_i \mathbf{t}_i \cdot \left((\hat{\mathbf{x}}_i^{k+1} - \hat{\mathbf{x}}_i^{k}) - (\tilde{\mathbf{x}}_i^{*k+1} - \tilde{\mathbf{x}}_i^{*k}) \right) \geq 0; \tag{8.10}$$

that is, the relative tangential movement between tissue and needle DYNAMIC coupling nodes must not change direction.

The equations and constraints together form a linear complementarity problem (LCP). Given q coupling nodes, there are 3^q possible settings of the friction states. The LCP has the potential to take exponential running time. Each wrong guess requires us to solve the linear system again, so even a moderate number of wrong guesses can kill real-time performance. Fortunately, the system has temporal coherence, and a good initial guess is to take the friction states from the previous time step. If these are wrong, we make local changes (driven by the constraints that are not satisfied) and usually find the correct states within a few trials. An advantage of our formulation (8.1) is that we can update it quickly when our guess of the friction states s_i change. No structural change of the matrix and no memory allocation are needed for each wrong guess.

Pseudocode for our LCP solver appears in Section 11.1, which discusses our method for updating the friction states and several optimizations that accelerate this computation.

8.5 Coupled system derivations

8.5.1 All Static Friction Case

To derived Equation 8.1, we start with the non-coupled system of equations of the tissue and the needle,

$$\hat{\mathbf{A}}\hat{\mathbf{a}}^{k+1} = \hat{\mathbf{b}}, \tag{8.11}$$

$$\tilde{\mathbf{A}}\tilde{\mathbf{a}}^{*k+1} = \tilde{\mathbf{b}}. \tag{8.12}$$

We assume for now that all the coupling nodes are STATIC to simplify the derivation, and extend it to the general case later on. We first introduce the coupling force $\mathbf{c_i} \in \Re^{\mathbf{3}}$ as a new unknown. It is in the local coordinate of each coupling node, where the first coordinate is along the tangent direction. Let $\mathbf{R}_i \in \Re^{3\times 3}$ be the rotation matrix that transforms space from local coordinates to world coordinates, so that $\mathbf{R}_i\mathbf{c}_i$ is the constraint force at the coupling node i in world coordinates. We also let $\hat{\mathbf{W}}$ and $\tilde{\mathbf{W}}$ be the 0-1 matrices that map coupling nodes to the tissue nodes and needle nodes, respectively. We can add the constraint forces with equal magnitude but opposite directions to the tissue and the needle system as

$$\hat{\mathbf{A}}\hat{\mathbf{a}}^{k+1} + \hat{\mathbf{W}}\mathbf{R}\mathbf{c} = \hat{\mathbf{b}}, \tag{8.13}$$

$$\tilde{\mathbf{A}}\tilde{\mathbf{a}}^{*k+1} - \tilde{\mathbf{W}}\mathbf{R}\mathbf{c} = \tilde{\mathbf{b}}. \tag{8.14}$$

We also need to add more equations to the system to constrain the position of each pair of corresponding needle and tissue coupling nodes i at the end of the time step to match (we assume they have the same index in both meshes):

$$\hat{\mathbf{x}}_\mathbf{i}^{\mathbf{k+1}} - \tilde{\mathbf{x}}_\mathbf{i}^{*\mathbf{k+1}} = \mathbf{0}. \tag{8.15}$$

From Newmark's Equation 4.1, we have

$$\hat{\mathbf{x}}_\mathbf{i}^{\mathbf{k+1}} = \hat{\mathbf{x}}_i^k + \triangle t\hat{\mathbf{v}}_i^k + \triangle t^2((\frac{1}{2}-\beta)\hat{\mathbf{a}}_i^k + \beta\hat{\mathbf{a}}_i^{k+1}), \tag{8.16}$$

$$\tilde{\mathbf{x}}_\mathbf{i}^{*\mathbf{k+1}} = \tilde{\mathbf{x}}_i^k + \triangle t\tilde{\mathbf{v}}_i^k + \triangle t^2((\frac{1}{2}-\beta)\tilde{\mathbf{a}}_i^k + \beta\tilde{\mathbf{a}}_i^{*k+1}). \tag{8.17}$$

Substituting Equations 8.16 and 8.17 into Equation 8.15, we have

$$\hat{\mathbf{x}}_i^k + \triangle t\hat{\mathbf{v}}_i^k + \triangle t^2((\frac{1}{2}-\beta)\hat{\mathbf{a}}_i^k + \beta\hat{\mathbf{a}}_i^{k+1}) - \tilde{\mathbf{x}}_i^k + \triangle t\tilde{\mathbf{v}}_i^k + \triangle t^2((\frac{1}{2}-\beta)\tilde{\mathbf{a}}_i^k + \beta\tilde{\mathbf{a}}_i^{*k+1}) = 0, \tag{8.18}$$

which simplifies to

$$\hat{\mathbf{a}}_i^{k+1} - \tilde{\mathbf{a}}_i^{*k+1} = \mathbf{q}_i, \tag{8.19}$$

which we can expressed as

$$\hat{\mathbf{W}}^\mathsf{T}\hat{\mathbf{a}}^{k+1} - \tilde{\mathbf{W}}^\mathsf{T}\tilde{\mathbf{a}}^{*k+1} = \mathbf{q}. \tag{8.20}$$

We multiply both sides of this equation with $\mathbf{R}^\mathsf{T}$ to give

$$\mathbf{R}^\mathsf{T}\hat{\mathbf{W}}^\mathsf{T}\hat{\mathbf{a}}^{k+1} - \mathbf{R}^\mathsf{T}\tilde{\mathbf{W}}^\mathsf{T}\tilde{\mathbf{a}}^{*k+1} = \mathbf{R}^\mathsf{T}\mathbf{q}. \tag{8.21}$$

Notice that the Equations 8.13, 8.14, and 8.21 form a symmetric linear system

$$\begin{bmatrix} \hat{\mathbf{A}} & 0 & \hat{\mathbf{W}}\mathbf{R} \\ 0 & \tilde{\mathbf{A}} & -\tilde{\mathbf{W}}\mathbf{R} \\ (\hat{\mathbf{W}}\mathbf{R})^\mathsf{T} & -(\tilde{\mathbf{W}}\mathbf{R})^\mathsf{T} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{a}}^{k+1} \\ \tilde{\mathbf{a}}^{*k+1} \\ \mathbf{c} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{b}} \\ \tilde{\mathbf{b}} \\ \mathbf{R}^\mathsf{T}\mathbf{q} \end{bmatrix}. \tag{8.22}$$

We solve this system to obtain $\hat{\mathbf{a}}^{k+1}$, $\tilde{\mathbf{a}}^{*k+1}$ and $\mathbf{c}$.

8.5.2 General Case

So far, we assumed that all coupling nodes are STATIC. For the general case that some nodes are DYNAMIC+ or DYNAMIC−, we make a few modifications to Equation 8.22. If a node i is a dynamic friction node, we already know the value of c_{i1} (f_i for DYNAMIC+ and $-f_i$ for DYNAMIC−). Therefore, we can remove c_{i1} from the system as a degree of freedom. We could simply perform substitution and remove the corresponding rows and columns of the matrix. However, this would require resizing the matrix, which would be inefficient for our interactive application. We hence instead do the following. First, we zero out the first column of $\mathbf{R}_i$, so that c_{i1} is ignored, if i is a dynamic friction node. We then let $\mathbf{d}_i$ be $[0,0,0]^\mathsf{T}$ for a STATIC node or $[c_{i1},0,0]^\mathsf{T}$ for a DYNAMIC node. Then we modify Equation 8.22 to

$$\begin{bmatrix} \hat{\mathbf{A}} & 0 & \hat{\mathbf{W}}\mathbf{R} \\ 0 & \tilde{\mathbf{A}} & -\tilde{\mathbf{W}}\mathbf{R} \\ (\hat{\mathbf{W}}\mathbf{R})^\mathsf{T} & -(\tilde{\mathbf{W}}\mathbf{R})^\mathsf{T} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{a}}^{k+1} \\ \tilde{\mathbf{a}}^{*k+1} \\ \mathbf{c} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{b}} + \hat{\mathbf{W}}\mathbf{R}\mathbf{d} \\ \tilde{\mathbf{b}} - \tilde{\mathbf{W}}\mathbf{R}\mathbf{d} \\ \mathbf{R}^\mathsf{T}\mathbf{q} \end{bmatrix}. \tag{8.23}$$

This system is singular because the rows and columns corresponding to the first coordinate of a dynamic node are 0. To make it non-singular, we let $\mathbf{Z}$ be a diagonal matrix in which a diagonal

entry corresponding to the first coordinate of a dynamic node is 1 and is 0 otherwise and modify the system to:

$$\begin{bmatrix} \hat{\mathbf{A}} & 0 & \hat{\mathbf{W}}\mathbf{R} \\ 0 & \tilde{\mathbf{A}} & -\tilde{\mathbf{W}}\mathbf{R} \\ (\hat{\mathbf{W}}\mathbf{R})^{\mathsf{T}} & -(\tilde{\mathbf{W}}\mathbf{R})^{\mathsf{T}} & \mathbf{Z} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{a}}^{k+1} \\ \tilde{\mathbf{a}}^{*k+1} \\ \mathbf{c} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{b}} + \hat{\mathbf{W}}\mathbf{R}\mathbf{d} \\ \tilde{\mathbf{b}} - \tilde{\mathbf{W}}\mathbf{R}\mathbf{d} \\ \mathbf{R}^{\mathsf{T}}\mathbf{q} \end{bmatrix}. \tag{8.24}$$

Letting $\mathbf{e} = \mathbf{R}^{\mathsf{T}}\mathbf{q}$, we have Equation 8.1.

Chapter 9

Remeshing and Reparameterization

After a simulated time step, some of the DYNAMIC coupling nodes of T and $\widetilde{T}$ are no longer coincident. However, our coupling method requires coupling nodes to have the same positions in both meshes. Thus, we dynamically adapt the meshes after each time step, which is the subject of this chapter. Our mesh adaptation consists of two steps: needle tip remeshing and needle reparameterization.

9.1 Needle Tip Remeshing

When the needle tip node is DYNAMIC, we change the tissue mesh T, sometimes topologically, so that it has a node at the new location of the needle tip. Mesh changes occur only near the needle tip, and are quite inexpensive. We skip this step if the tip node is STATIC.

The goals of needle tip remeshing are to make T conform to the needle, to have tetrahedra of as high quality as possible, and to do so quickly. We remesh in material space by applying one of the candidate operations depicted in Figure 9.1: the node snap, the edge split, the face split, and the tetrahedron split.

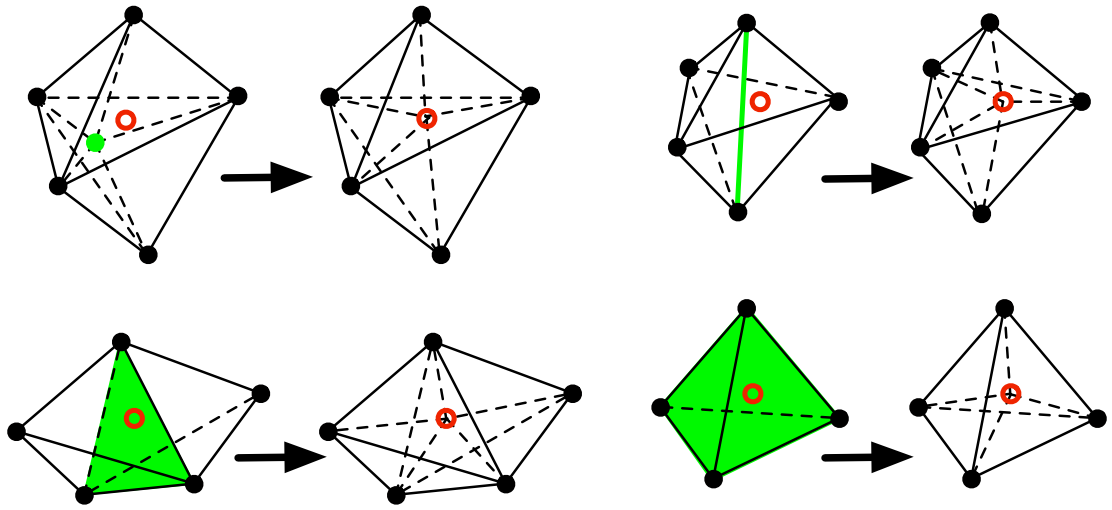

Figure 9.1: Mesh operations: node snap, edge split (upper right), face split (lower left), tetrahedron split. The needle tip $\mathbf{u}_{\text{new}}$ is at the red circle. The simplex the operation acts on is green.

9.1.1 Tip Remeshing Algorithm

Two ideas govern our remeshing algorithm. First, we choose among candidate operations by directly measuring the quality of the tetrahedra that would be created by each operation, and selecting the operation that maximizes the quality of the worst tetrahedron. Second, we maintain a stack of all the operations that have changed the mesh topology (*i.e.* all operations except node snaps), and we consider undoing the most recent operation before applying a new one. The stack is particularly important when the needle is retracted; our procedure is designed so that once the needle is fully withdrawn from the body, the tissue mesh will have returned to its original topology. We thereby prevent the accumulation of mesh quality degradation when the needle is inserted and withdrawn multiple times. Even when the needle is being inserted, the ability to undo the previous operation and replace it with a new one often offers better mesh quality.

Evaluating Remeshing Operations

For each candidate operation, we evaluate each new tetrahedron with a quality measure equal to its signed volume divided by the cube of its root-mean-squared edge length [50]. This measure is zero for a degenerate tetrahedron, and maximized by an equilateral tetrahedron. Various tetrahedra

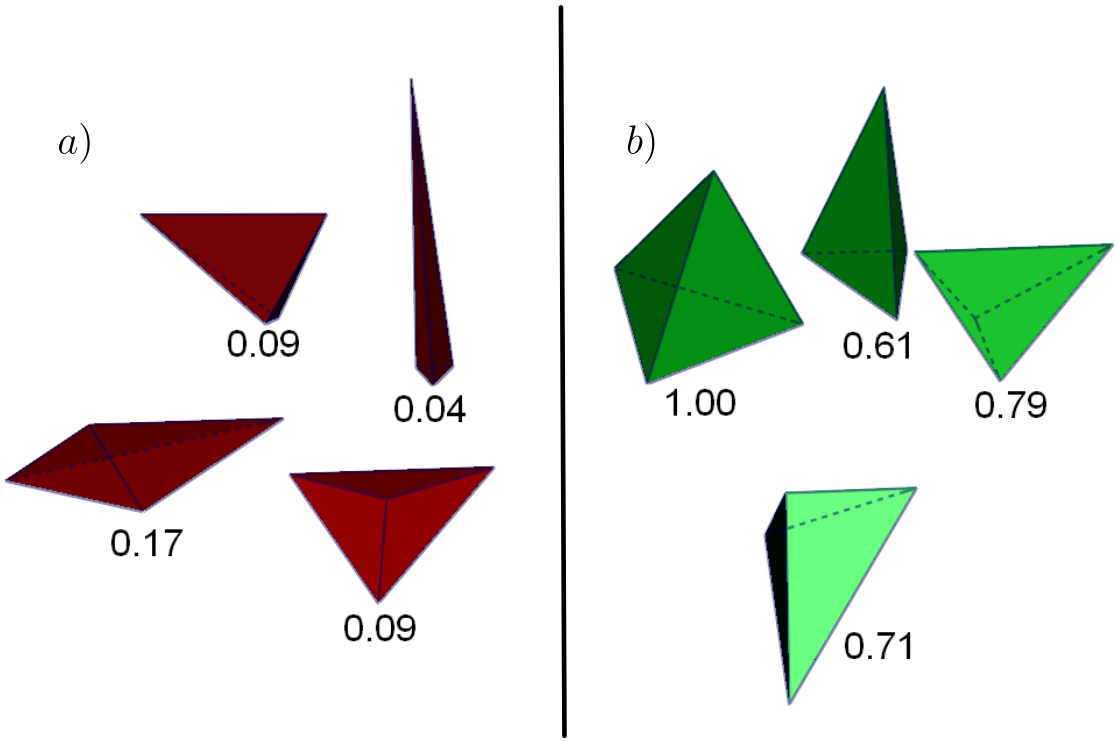

Figure 9.2: Various tetrahedra and their corresponding normalized quality, where the quality of equilateral tetrahedron is 1. a) Bad quality tetrahedra, clockwise from top left: wedge, needle, cap, and sliver. b) Good quality tetrahedra.

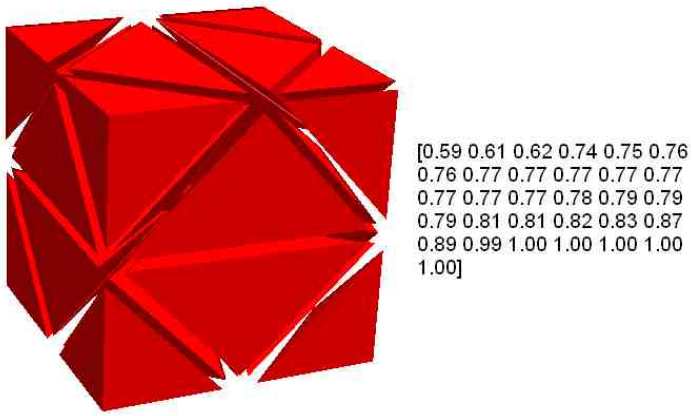

Figure 9.3: A small tetrahedral mesh with 31 tetrahedra and its corresponding normalized quality vector.

and their qualities are shown in Figure 9.2. We have found this measure to be both a good reflection of a tetrahedron's fitness for finite element simulation and amenable to numerical optimization [33]. Tetrahedron quality is always computed from material (not world) coordinates. Let R be the mesh region comprising the union of all tetrahedra that can be deleted or changed by the candidate operations. The *quality vector* of the tetrahedra in R is a list of the tetrahedron qualities, sorted from worst to best. The quality vector of a tetrahedral mesh is shown in Figure 9.3. Our algorithm chooses the operation that lexicographically maximizes that quality vector (that is, it maximizes the worst tetrahedron, breaking ties by maximizing the second-worst, then the third-worst, etc.).

Candidate Remeshing Operations For Needle Penetration

If the needle is penetrating tissue, we generate a set of candidate operations as follows. Let $\mathbf{u}_{\text{new}}$ be the new position of the needle tip in material space. (We obtain $\mathbf{u}_{\text{new}}$ by barycentric interpolation from the needle tip position in world space, and we obtain a tissue velocity and acceleration for it the same way.) We consider fifteen *standard operations* that transform the tetrahedron that contains $\mathbf{u}_{\text{new}}$: four node snaps, six edge splits, four face splits, and one tetrahedron split. Each operation places the new node or snapped node at $\mathbf{u}_{\text{new}}$. We also consider composite operations that first undo the operation atop the stack, then apply a new standard operation.

Legal Candidate Operations

Candidate operations that would change the mesh boundary are discarded, except when the needle first penetrates the skin. Operations that fail to properly connect the needle nodes are also discarded. Let $\mathbf{u}_1$ and $\mathbf{u}_2$ be the old positions in material space of the needle tip and the needle node next to the tip, respectively; see Figure 9.4. The needle nodes remain properly connected by the operations that snap $\mathbf{u}_1$ to $\mathbf{u}_{\text{new}}$, that create an edge connecting $\mathbf{u}_1$ to $\mathbf{u}_{\text{new}}$, or that delete $\mathbf{u}_1$ and create an edge connecting $\mathbf{u}_2$ to $\mathbf{u}_{\text{new}}$. (Undoing the top stack operation entails deleting $\mathbf{u}_1$ from both T and $\widetilde{T}$.)

Tip's Neighbor Node Optimization

Because the needle moves only a small distance during a time step, $\mathbf{u}_{\text{new}}$ tends to be close to $\mathbf{u}_1$ or $\mathbf{u}_2$, often producing a short edge that compromises the mesh quality. We avoid this pitfall by moving

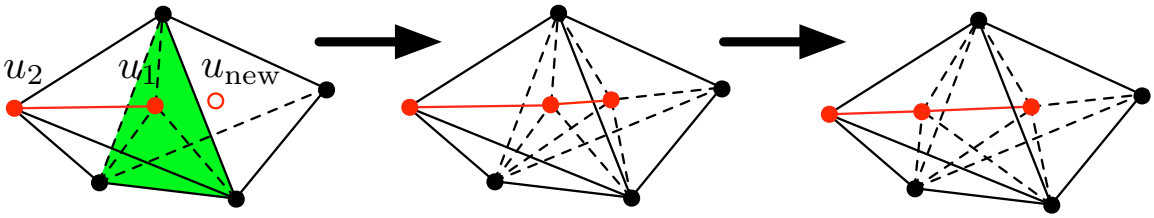

Figure 9.4: The red edges and nodes are part of the needle shaft. The old positions in material space of the needle tip and the node next to the tip are denoted as u_1 and u_2 respectively. The new position of the needle tip is denoted by u_{new}. The green face is split, creating a new needle tip node. We move the old needle tip node back along the needle shaft to maximize the mesh quality.

$\mathbf{u}_1$ to the optimal position on the segment $\mathbf{u}_{\text{new}}\mathbf{u}_2$, or (if the operation deletes $\mathbf{u}_1$) by moving $\mathbf{u}_2$ to the optimal position on the segment $\mathbf{u}_{\text{new}}\mathbf{u}_3$; see Figure 9.4. (To respect the needle curvature, we could have searched along the energy-minimizing curve that Spillmann and Teschner [62] use, but we find that placing the node on the segment is good enough.) The "optimal" position is the one that maximizes the minimum quality among the tetrahedra that adjoin the moved node. We approximate it by sampling a number of points (10 in our implementation) along the segment and compute the quality of the resulting mesh if we were to reposition $\mathbf{u}_2$ there. We then choose the position that yields the best mesh quality. This repositioning is part of the candidate operation, and is taken into account when the best operation is chosen.

Candidate Remeshing Operations For Needle Retraction

Needle retraction uses somewhat different candidate operations. The *only* operation we consider that does not delete the needle tip $\mathbf{u}_1$ from the needle mesh is a node snap that moves $\mathbf{u}_1$ to $\mathbf{u}_{\text{new}}$. The other candidate operations delete $\mathbf{u}_1$ as follows. If $\mathbf{u}_1$ was created by the operation on top of the stack, then the stack is popped and that operation is undone, deleting $\mathbf{u}_1$ from both T and $\widetilde{T}$; otherwise, $\mathbf{u}_1$ was placed by a node snap, in which case we delete it from the needle mesh $\widetilde{T}$ only. In either case, one of the standard operations subsequently creates a node at $\mathbf{u}_{\text{new}}$, or snaps a node there. If $\mathbf{u}_1$ survives in the tissue mesh, it tends to be close to $\mathbf{u}_{\text{new}}$, so we subsequently optimize the position of $\mathbf{u}_1$ (but not $\mathbf{u}_2$) as part of the candidate operation. Because $\mathbf{u}_1$ no longer lies on the needle, it can move freely. We approximate the optimum placement of $\mathbf{u}_1$ by uniformly sampling along the ray that originates at $\mathbf{u}_{\text{new}}$, points directly away from $\mathbf{u}_2$ and terminates at the closest intersection with a face of a tetrahedron in T. In our implementation, we choose the position that

maximizes the mesh quality among the 10 samples.

Our remeshing procedure is summarized in Algorithm 2. Figure 9.5 depicts mesh modifications during needle insertion. The remeshing algorithm relies on the assumption that the needle tip will not move much further than an element's width in a single time step. If we needed to accommodate larger needle movements in a single step, then the large motion could be broken down into smaller substeps with multiple applications of the inexpensive remeshing algorithm.

Algorithm 2 Tissue remeshing at the needle tip

/* Build set S of candidate operations */
$S \leftarrow \{$ node snap, moving needle tip $\mathbf{u}_1$ to $\mathbf{u}_{\text{new}}\}$
$t \leftarrow$ tetrahedron in T containing $\mathbf{u}_{\text{new}}$
if $s_1 = \text{DYNAMIC}+$ (the needle is penetrating) **then**
    $S \leftarrow S \cup$ set of standard operations on t that place a node (other than $\mathbf{u}_1$) at $\mathbf{u}_{\text{new}}$ that is connected to $\mathbf{u}_1$, then optimize the position of $\mathbf{u}_1$ constrained to lie on $\mathbf{u}_{\text{new}}\mathbf{u}_2$
end if
if $\mathbf{u}_1$ was created by the operation atop the stack **then**
    $t_{\text{undo}} \leftarrow$ tetrahedron that will contain $\mathbf{u}_{\text{new}}$ if the operation atop the stack is undone
    $S \leftarrow S \cup$ set of operations that undo the top stack operation, then perform a standard operation on t_{undo} that places a node at $\mathbf{u}_{\text{new}}$ that is connected to (or is) $\mathbf{u}_2$, then optimize the position of the needle node adjoining $\mathbf{u}_{\text{new}}$
else if $s_1 = \text{DYNAMIC}-$ (the needle is retracting) **then**
    $S \leftarrow S \cup$ set of standard operations on t that place a node at $\mathbf{u}_{\text{new}}$ that is connected to (or is) $\mathbf{u}_2$, then freely optimize the position of $\mathbf{u}_1$.
end if
$T' \leftarrow$ set of tetrahedra deleted/changed by operations in S
for each candidate operation $o \in S$ **do**
    Compute the quality vector for the tetrahedra created by o and the tetrahedra in T' not deleted by o
end for
Perform operation that maximizes the quality vector
if optimal operation undoes the top stack operation **then**
    Pop the stack
end if
if optimal operation includes a face/edge/tetrahedron split **then**
    Push the split operation onto the stack
end if

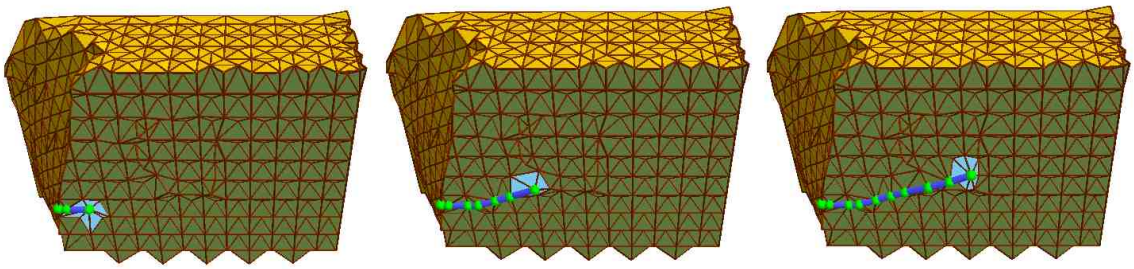

Figure 9.5: Cutaway views of the changing mesh at several times during a simulation. The most recently changed tetrahedra (around the needle tip) are highlighted in blue.

9.1.2 Tip Remeshing Algorithm Analysis

Because of the restricted demands of our application, our remesher is effective at maintaining element quality: in all the simulations we have run, after the first twenty time steps, the mesh quality does not drop below 0.7 times the quality of the worst tetrahedron in the initial mesh and no tetrahedron's dihedral angle has been smaller than 10.3° or larger than 160.0°. During the first few time steps after the needle punctures the skin, it is impossible to prevent the presence of a very short edge connecting the needle tip to the mesh surface, with flat tetrahedra adjoining it; but at all other times, tetrahedron quality is good.

Remeshing cannot always fix inverted elements, but that's not an impediment, because if an element has become inverted, the accuracy and stability of the solution are already compromised. If the needle tip moves beyond its 1-ring neighborhood in a single time step, a tetrahedral element may become inverted. The quality of the mesh resulting from moving the needle tip to a new position in a single step is shown in Figure 9.6. It is visible from the figure that the mesh quality tends to be above zero when the needle does not move too much in a single remeshing step.

In general, we observe from experiments that the algorithm generates mesh with good enough quality even after many cycles of the needle insertion and retraction.

9.2 Needle Reparameterization

Our needle tip remeshing procedure ensures that the tissue mesh has a sequence of nodes and edges that corresponds to the part of the needle inside the tissue. These nodes will be the coupling nodes

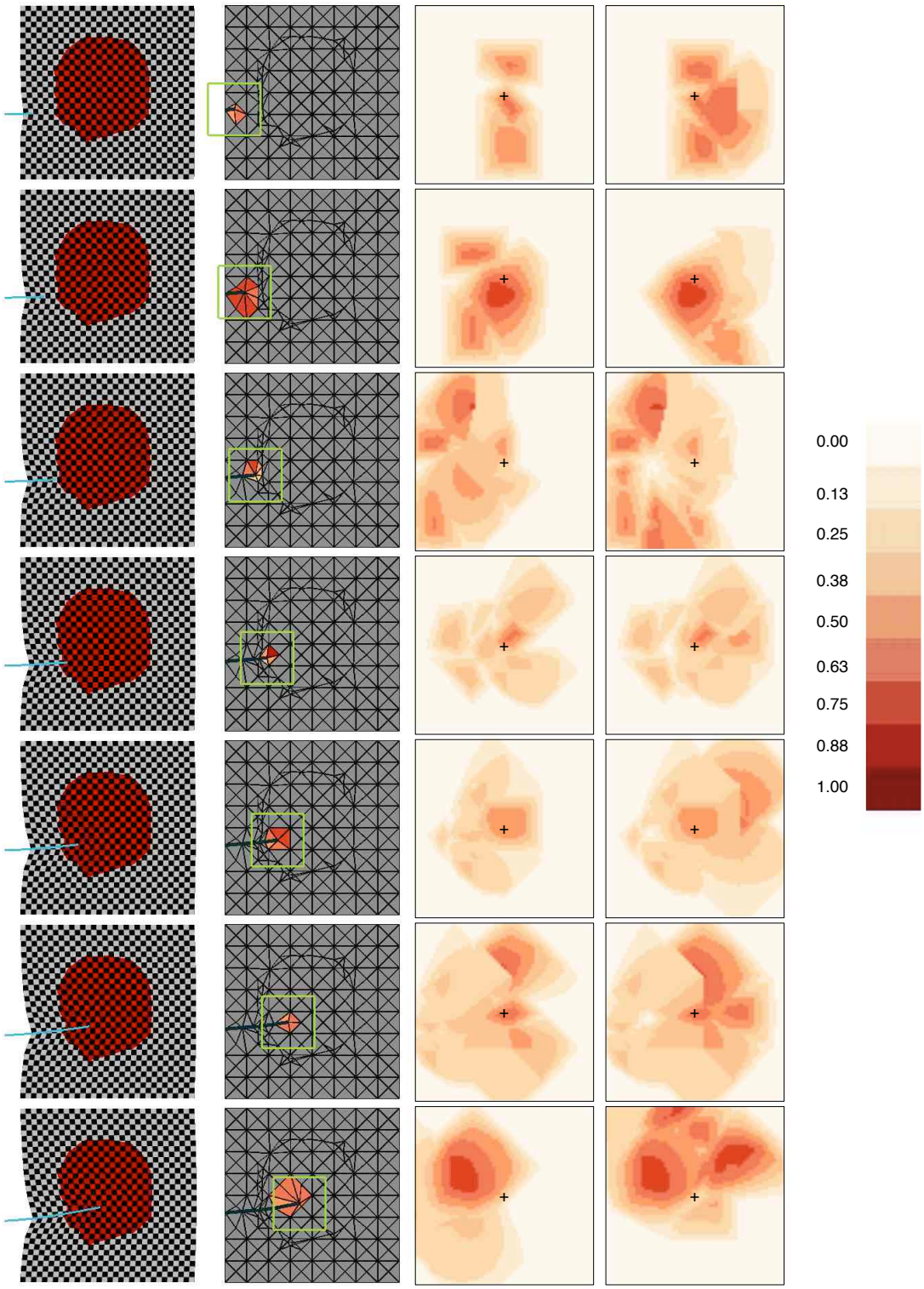

Figure 9.6: Leftmost: Snapshot of simulation in world space. Left middle: Cutaway mesh in material space. Green box shows the region that represents the two right plots. Right middle: Quality of the worst tetrahedron (clamping negative quality to 0, color coded) vs. possible position of the needle tip in the next time step, in the case of needle retraction. Rightmost: Quality of the worst tetrahedron vs. possible position of the needle tip, in the case of needle penetration.

in the next time step, and their positions are determined by the tissue mesh; that is, $\tilde{\mathbf{x}}_i^{k+1} = \hat{\mathbf{x}}_i^{k+1}$. The FREE nodes' positions are determined by the solution to Equation (8.1); that is, $\tilde{\mathbf{x}}_i^{k+1} = \tilde{\mathbf{x}}_i^{*k+1}$.

The needle sliding at a DYNAMIC node implies that it no longer represents the same point on the needle as it did before the time step. Moreover, remeshing creates and deletes nodes, and the simulation does not keep the needle perfectly inextensible, so the needle length varies slightly. Therefore, we reparametrize the needle and interpolate physical quantities from before to after the time step.

We parametrize each node i existing before the time step by its distance d_i^* from the base of the needle, and each node j existing after by its distance d_j from the base after the time step. (We compute these distances as sums of line segment lengths, but one could use the arc length of an interpolating curve instead.) Because the needle is not perfectly inextensible, we scale all the distances after the time step so the values of d^* and d at the needle tip are equal.

To compute the acceleration $\tilde{\mathbf{a}}_j$ at node j after a time step, we build an interpolating function $g(\cdot)$ such that $g(d_i^*) = \tilde{\mathbf{a}}_i^*$, where the right-hand side comes from the solution of Equation (8.1), then set $\tilde{\mathbf{a}}_j = g(d_j)$, as illustrated in Figure 9.7. We use Akima's interpolation [2] to construct the interpolating functions for the parameters used in the needle force computations of Bergou et al.[11] and Spillmann and Teschner [61], such as the twist angle θ_i (the angle of deviation from the Bishop frame at node i) and the rest curvature $\bar{\omega}_i$. Akima's interpolation function is a piecewise cubic C1 continuous polynomial. Each piece is constructed from only next neighbor points and hence is very efficient. We treat the rest lengths $\bar{l}_j$ of the needle edges as a special case, by first constructing $\bar{L}_0^* = 0, \bar{L}_i^* = \sum_1^i \bar{l}_i^*$, then using Akima's interpolation for $\bar{L}$, then setting $\bar{l}_q = \bar{L}_q - \bar{L}_{q-1}$. We use piecewise linear interpolation for velocity and acceleration, since Akima's interpolation is not monotonic and could compromise stability.

9.2.1 Needle Node Splitting and Merging

A needle edge outside the tissue can become too short or too long in two places: where the needle exits the guide sleeve (see Section 10.2), and where the needle enters the tissue. Thus, we merge nodes that are too close together (shorter than half the minimum initial edge length), moving a FREE node onto the node on the surface of the tissue or the end of the sleeve; and we split edges that are too long (over four times the maximum initial edge length), all before reparameterizing. To split an edge, we place a new node at the midpoint of an interpolating cubic curve.

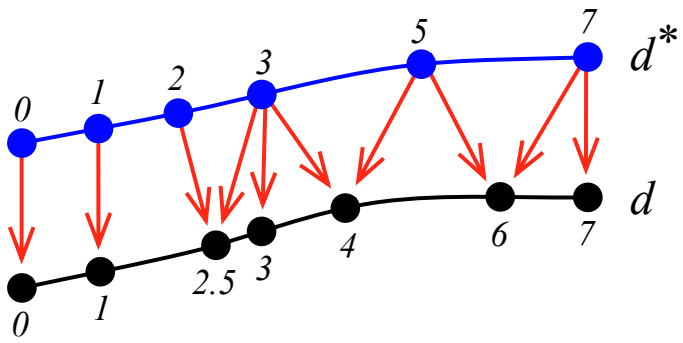

Figure 9.7: Needle reparameterization. The numbers are d^* (top) and d (bottom). The red arrows indicate which old nodes and new nodes' quantities are related via piecewise linear interpolation.

Chapter 10

Bevel tip and Base Manipulation

The simulation method described so far suffices to model a needle whose tip is symmetric, causing it to penetrate tissue in a straight line. However, surgeons can more easily circumvent obstacles by using bevel-tip needles that move on curved paths because the beveled tip compresses tissue asymmetrically [74]. In this chapter, we will describe the extension necessary for simulating a bevel tip needle. There are three manipulations that can be used to control the needle: rotating its base, inserting or retracting the needle through the guide sleeve, and moving the catheter. We will also explain how the simulator can support these manipulations.

10.1 An Extension to Handle a Bevel Tip Needle

Bevel-tip needles can move on curved paths because the beveled tip compresses tissue asymmetrically. As shown in Figure 10.1a, the needle moves from left to right and the bevel tip cut though the tissue and compress the portion of the tissue above it. The compressed tissue hence exerts an elastic force on the needle and causes it to bend. This phenomenon occurs at a scale too small for the tetrahedral mesh to simulate directly.

Instead, we approximate the effect by adding a displacement to the needle tip material coordinate $\mathbf{u}_{\text{new}}$. This will induce an elastic force that will push the tip of the needle laterally. The displacement is added along the second axis of the local coordinate frame of the needle tip, $\mathbf{m}_2$ as defined by Bergou et al. [11], which correspond to the direction that the needle supposed to bend transformed into the material coordinate. The magnitude of the displacement is $h = d \tan \psi$, where d is the

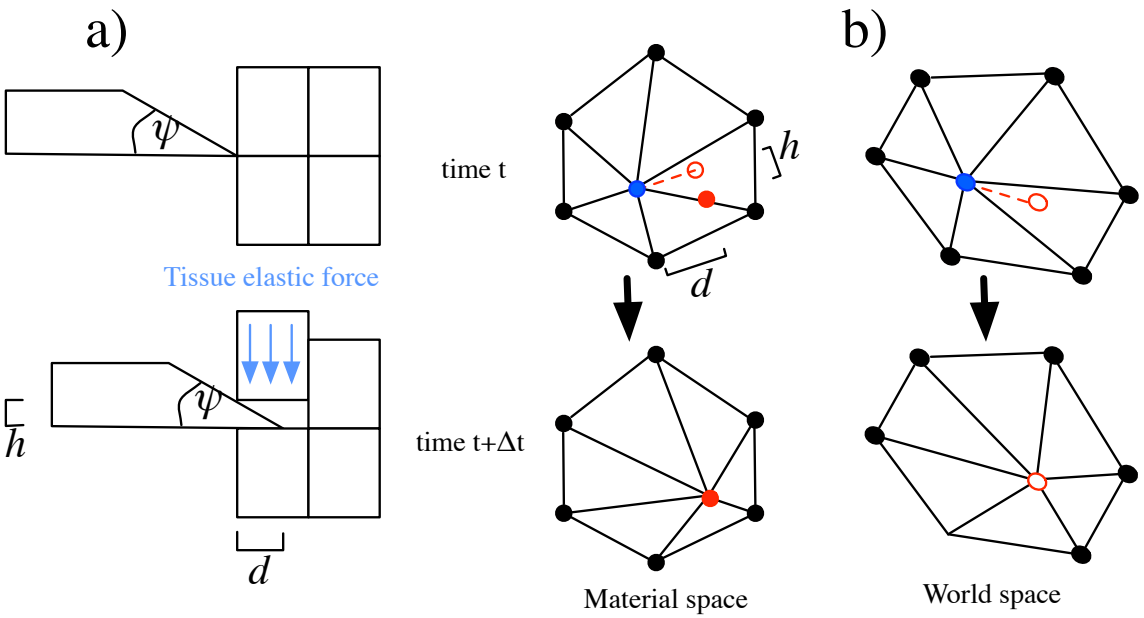

Figure 10.1: a) As the needle penetrates the tissue, the bevel tip compresses the tissue in one direction more than the other, inducing an asymmetric elastic tissue force that causes the needle to bend. As the needle moves a distance d, it displaces the tissue on one side by a distance of $h = d \tan \psi$. b) The current tip position appears as a blue circle. The new tip position appears as a hollow red circle. The new tip position is displaced to the solid red circle in material space, but not in world space. This displacement simulates a strain near the needle tip that will bend the needle in subsequent time steps.

distance the tip moves along the tangential direction $\mathbf{t}_i$ during the current time step, and ψ is the bevel angle depicted in Figure 10.1b.

The amount of the needle bending depends on the stiffness of tissue because for the same displacement, stiffer tissue exerts more elastic force on the needle tip and hence the needle will bend more. This is consistent with the observation made by Webster et al. [73] on the insertion of a Nitinol needle into artificial tissue phantoms.

10.2 Needle Manipulations

Steerable needles offer four control parameters to the surgeon as shown in Figure 10.2:

1. $\mathbf{v}_{\text{sleeve}}$, the rigid-body velocity of the sleeve that holds the base of the needle.

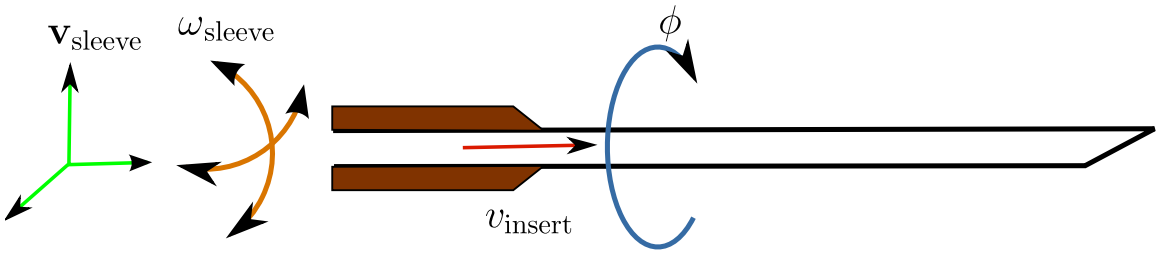

Figure 10.2: Four manipulations: 1) moving the sleeve that holds needle base with velocity $\mathbf{v}_{\text{sleeve}}$; 2) changing the orientation of the sleeve with angular velocity ω_{sleeve}; 3) inserting needle with speed v_{insert}, which will be negative for retraction; 4) twisting the needle with speed ϕ.

2. ω_{sleeve}, the rigid-body angular velocity of the sleeve.

3. v_{insert}, the speed at which the needle advances from the guide sleeve.

4. ϕ, the twisting speed of the needle which will be positive for counter-clockwise twisting and negative for clockwise twisting.

The position and the orientation of the sleeve are integrated over time in a similar fashion to the rigid body simulation presented by Baraff in [10]. The portion of the needle inside the sleeve is mechanically constrained to move with the sleeve. To facilitate this in our simulator, we add two nodes to the needle mesh and fix their velocities that of the sleeve determined by $\mathbf{v}_{\text{sleeve}}$ and ω_{sleeve}. We must constrain two nodes because the direction of the edge at the base of the needle is dictated by the sleeve orientation.

To push/retract the needle out through the sleeve, we increase/decrease the rest length of the second edge (just outside the sleeve) by $v_{\text{insert}} \triangle t$. We split the edge if it is longer than $l_{\max}$ and merge the two end points of the edge if it is shorter than $l_{\min}$. This ensures that the needle mesh is suitable for simulation. We use $l_{\max} = 1$ cm and $l_{\min} = 2$ mm in our experiments.

To rotate the base, we add $\phi \triangle t$ to θ_i at the needle's base node i. The needle simulation propagates this rotation to the bevel tip appropriately.

Chapter 11

Optimizations

To achieve interactive performance for a reasonably high-resolution simulation mesh, we employ several optimization techniques. First we discuss how we accelerate the LCP solver, which is the bottleneck of the simulation. We then discuss how we optimize the conjugate gradient computation. We next explain how we update the non-zero pattern of the sparse matrix dynamically. Finally, we describe how we incrementally update the Jacobian matrices.

11.1 Accelerating the LCP Solver

The simulation's bottleneck is solving linear complementarity problems. The speed of solving LCP depends crucially on rapidly determining the friction state variables s_i. Empirically, these values are temporally coherent if the needle is manipulated smoothly (as it should be for this application). The obvious strategy for guessing the states works well: presume that the state variables remain the same from time step to time step. However, needle reparametrization means that there is not necessarily a one-to-one mapping from nodes in one time step to nodes in another. We guess the state at each new node by using the nearest node from the previous time step.

After solving the coupled system (8.1) for a set of friction states, we check if the constraints (8.8, 8.9, and 8.10) are satisfied. If not, we try to alter the state of each node whose constraint is not satisfied using the following rules. If the node is DYNAMIC, change it to STATIC. If the node is STATIC, change it to DYNAMIC with its sign determined by the coupling constraint force $[1\ 0\ 0]^{\mathsf{T}} \cdot \mathbf{c}_i$.

To avoid trying the same configuration twice, we record those we have tried. If we are about to

try one we have already tried, we instead try the next configuration in lexicographic order, with the node at the tissue entry point acting as the least significant "digit."

With this heuristic, we find a consistent set of friction states in fewer than 1.2 trials on average in our examples. Temporal coherence is lost when the needle switches from cutting to withdrawing, in which case many iterations might be required to find a consistent configuration. We devised two additional optimizations to keep the frame rate high during these transitions.

Our conjugate gradient solver is always initialized with the solution from the previous (unsuccessful) trial. This simple trick significantly cuts down the number of iterations the conjuage gradient method takes to converge to a tolerance τ_{sol} (by a factor of twenty in our prostate examples). Interestingly, we did not attain speedups by initializing the solver to the solution from the previous time step.

We took this optimization a step further by deciding whether to change state variables before the solver fully converges. As soon as the conjugate gradient residual drops below a tolerance τ_{check} (which is larger than τ_{sol}), we check if any constraints are violated, and switch state variables immediately if they are. In our simulation results, $\tau_{\text{check}} = 10^{-3}$ and $\tau_{\text{sol}} = 10^{-5}$.

Finally, if a consistent configuration isn't found within ten trials, we simply accept the last configuration to ensure real-time performance. In our prostate example, our solver found a consistent configuration within ten trials for 99.6% of the time steps. For those that fail, the resulting forces and accelerations are close to the true solution. Moreover, the simulator is likely to find the consistent configuration within the next few time steps, because of the warm start. Our LCP solver is summarized in Algorithm 3.

11.2 Parallel Sparse Conjugate Gradients

We use the OpenMP conjugate gradient implementation, which parallelizes the sparse matrix multiplication and all the vector operations. To improve the cache performance of sparse matrix access, we order the tissue mesh nodes with a reverse Cuthill–McKee ordering [18], which puts the nonzero entries close to the diagonal.

We avoid building a new sparse matrix for each solution by treating the coupling terms separately from the other terms in the matrix multiplication. In Equation (8.1), the matrices $\hat{\mathbf{A}}$ and $\tilde{\mathbf{A}}$ are fixed

Algorithm 3 Solve the coupled system, yielding the friction states and the tissue and needle accelerations

```
1: Guess friction states s_i from previous time step
2: Initialize solution to 0
3: for numTrials ← 0 to maxTrials do
4:     Solve linear system (8.1) to tolerance τ_check
5:     if the constraints (8.8), (8.9), and (8.10) are satisfied then
6:         Continue solving linear system to tolerance τ_sol
7:         if the constraints (8.8), (8.9), and (8.10) are satisfied then
8:             return solution
9:         end if
10:    end if
11:    Change friction states s_i to something new (see §11.1)
12:    /* Next iteration will reuse solution from this trial */
13: end for
14: Solve linear system (8.1) to tolerance τ_sol
15: return (approximate) solution
```

during an LCP solution, whereas $\mathbf{R}$ and $\mathbf{Z}$ vary with the coupling states. We never assemble the third row or column of (8.1); we perform that portion of the matrix-vector multiplication element by element.

We also experimented with a diagonal preconditioner which does not seem to speedup the convergence. Incomplete Cholesky Factorization can cut down the number of iterations significantly, but it cost more per iteration and we also need to build it. Using TAUCS [68] for the factorization, we determine experimentally that it does not yield an overall speed improvement.

11.3 Dynamic Update of Sparse Matrix

We store $\hat{\mathbf{A}}$ with a dynamic compressed sparse row (CSR) format that can be updated to reflect tissue remeshing without being reassembled from scratch. The rows of the matrix are stored in a resizable array. Each compressed row of the matrix is stored in a separate resizable array that can be resized to accommodate new edges.

The helper data structures that allow efficient update to the matrix consist of a node free list $\mathbf{L}^g$, a hash table $\mathbf{H}(i,j)$ that stores the index of the non-zero corresponding to the matrix entry (i,j) in the compressed row i and edge free lists $\mathbf{L}_i$ for node i.

As tetrahedra are deleted, created, and modified, we update their contributions to $\hat{\mathbf{A}}$. A node addition can be done by checking if $\mathbf{L}^g$ is empty, if so, we add an extra row entry and push to $\mathbf{L}^g$. We then pop the node's index from $\mathbf{L}^g$. A node i deletion causes an entire row i and column i to be set to zero. It is done by first looping through row i and zero out (i, j) and (j, i) (which is done efficiently by looking up $\mathbf{H}(j, i)$), then update the free list $\mathbf{L}_i$ and $\mathbf{L}_j$, and finally push i to the free list $\mathbf{L}^g$. An edge (i, j) update is done by checking $\mathbf{H}(i, j)$ if the edge exists, if so, we simply update it. If not, we check if $\mathbf{L}_i$ is empty, if so, we add an extra edge entry to row i and push to $\mathbf{L}_i$. We then pop the new edge from $\mathbf{L}_i$, update $\mathbf{H}(i, j)$ and then update the entry value.

For efficiency, we do not reallocate memory for a resizable array every time a new node (or edge in a row) is added. We instead reallocate by doubling the physical size of the array and maintain the current size of the array explicitly. Effectively, this will require only one memory allocation once every time the number of nodes (or edges in a row) is doubled and is amortized to a small constant time per new node (or edge) addition.

Moreover, unused column entries in an active row are set to zero. Because remeshing changes only a small portion of the mesh, the time consumed by these zero entries is negligible compared to rebuilding the matrix. The most common remeshing operation is to snap a node to the needle tip, which entails no change to the sparse matrix structure.

11.4 Parallel Incremental Jacobian Update

We also parallelize the computation of the forces and the Jacobian matrices. To reduce lock contention, we partition the mesh with METIS [31], so that the entries modified by one thread do not overlap much with those of any other.

We observe that the Jacobians tend to change slowly over time, because most tetrahedra deform little during a time step. Therefore, we only update a tetrahedron's Jacobian matrix's contributions to $\hat{\mathbf{A}}$ (as determined by Equation (4.4)) when it has changed enough since its last update. The test is whether

$$e = \left(\max_{i,j} |\triangle U_{ij}| \max_{i,j} |\triangle V_{ij}| \right)^2 \max_{i,j} |N_{ij}| \max_{i,j} |B_{ij}| \max(\lambda, \alpha) \tag{11.1}$$

exceeds a threshold e_{thres}, where U and V are the SVD rotation matrices mentioned in Section 5.3, N is a matrix whose columns are area-weighted normal vectors to the tetrahedron faces, and B is

e_{thres}	% updated	F&J (ms)	Total frame time (ms)	% rel error
0	100.00	12.42	36.30	0.00
100	7.96	3.83	29.90	0.06
1000	3.65	2.52	28.92	0.19
10000	0.58	2.12	28.64	0.16

Table 11.1: For several values of e_{thres}, the % of tetrahedra whose Jacobian matrices are updated per time step, the time required for the force and Jacobian computations (F&J), the total frame time, and the relative error in the tip position after four seconds of needle insertion (200 time steps). Values are for the prostate mesh ($\sim$13,375 tetrahedra) running with seven threads on an 8-core processor. The relative error for e_{thres} of 10000 is less than that of 1000 just by chance.

the tetrahedron's barycentric matrix. λ and α are the second coefficients for computing the Piola–Kirchhoff stress and the damping stress in Irving et al. [29], respectively. Equation 11.1 is an approximate upper bound on the change in any entry of the Jacobian matrix due to changes in U and V, for both the elastic and damping forces. Table 11.1 shows the running time reductions and the relative errors in needle tip position caused by lazy updating for several values of e_{thres}. Lazy updating reduces the tissue force and Jacobian computation time by 83% and the frame time by 20%, with no more than a 0.2% relative error in the needle tip position.

Chapter 12

Results and Discussion

We implemented the simulator in C++ and use it to simulate various scenarios. First, we simulate the prostate brachytherapy, where the needle is inserted to plant radioactive seeds in the prostrate. We then demonstrate the steering capability of the bevel tip needle in an artifical example with two cylindrical obstacles. We then verify the simulation result against a real world experiment on inserting a needle into a tissue phantom. We next discuss two applications of the simulator in motion planning research projects and finally suggest future directions for the research.

12.1 Examples

The video accompanying this thesis can be downloaded at http://graphics.cs.berkeley.edu/papers/Chentanez-ISN-2009-08/Chentanez2009-ISN-hires.mov

12.1.1 Prostate Brachytherapy

Figure 12.1 illustrates our simulations of a prostate cancer brachytherapy procedure, wherein a needle implants radioactive seeds in the prostrate gland, for both a flexible bevel-tip needle (a) and a stiff symmetric-tip needle (b). We use isosurface stuffing [36] to generate a tetrahedral mesh encompassing an anatomically accurate model of the epidermis, dermis, hypodermis, urethra, prostate, perineum, pelvic bone, bulbourethral glands, vasa deferentia, seminal vesicles, and bladder. The external mesh boundary conforms to the skin, and internal triangular faces conform to the prostate boundary, separating regions with dissimilar material properties. Material parameters

for the prostate and other tissues are inferred from Krouskop et al. [35]. The bone is constrained not to move. Figure 12.2 and Figure 12.3 show screenshots from the animation of the needle being inserted into and removed from the tissue for the bevel tip flexible needle and the symmetric tip stiff needle, respectively. The needles are manipulated both gently and vigorously, to demonstrate the simulator's stability.

12.1.2 Obstacle Avoidance

We also constructed an artificial example, shown in Figure 12.4, where we simulate the ability of the bevel-tip flexible needle to be steered by rotating its base to avoid fixed cylindrical obstacles and hit a target in a deformable tissue. Figure 12.5 and Figure 12.6 show screenshots of the obstacle avoidance animation from the side view and from the back view respectively.

12.1.3 Evaluation

To evaluate our simulator, we compare the simulated insertion of steerable needles into gel tissue phantoms with real-world experiments performed at Johns Hopkins University in 2005, as shown in Figure 12.7. In these experiments, a robotic device [73] inserts a bevel-tip flexible needle of diameter 0.83 mm into a $27.1 \times 26.5 \times 3.9$ cm gelatin block. Fiducial markers are placed on the surface of the gel phantom to track the deformations. The material's Young's modulus (E) and Poisson's ratio (v) were measured using compression tests on $1.77 \times 1.77 \times 1.77$ cm samples. The average and standard deviation (S.D.) are $\bar{E} = 125.89$ kPa with S.D. $= 15.551$ kPa and $\bar{v} = 0.453$ with S.D. $= 0.007$. We use the average values in our simulation.

The remaining simulation parameters (tissue damping, needle parameters) were tuned by hand to match a *single-bend* experiment, where the needle is inserted and retracted from the tissue without twisting. The same parameters are then used without re-tuning in the second *double-bend* experiment, where the needle is twisted 180° halfway through insertion. Figure 12.8 and Figure 12.9 show the screenshots of the videos with the simulated markers and needles overlaid on top for the single-bend and the double-bend experiments respectively.

We ran the simulation, tracked the positions of simulated markers as computed by barycentric interpolation on the tetrahedra, and compared their positions to the physical markers in the experiment.

	$\bar{E}-2$ S.D.	$\bar{E}-$S.D.	$\bar{E}$	$\bar{E}+$S.D.	$\bar{E}+2$ S.D.
$\bar{v}-2$ S.D.	0.924	0.763	0.741	0.795	0.888
$\bar{v}-$S.D.	0.919	0.768	0.755	0.801	0.885
$\bar{v}$	0.916	0.761	0.753	0.809	0.893
$\bar{v}+$S.D.	0.910	0.759	0.755	0.814	0.894
$\bar{v}+2$ S.D.	0.900	0.761	0.752	0.842	0.929

Table 12.1: Sensitivity of the marker positions' error with respect to the changes in E and v, for the single-bend experiment. E and v are varied from -2 S.D. to $+2$ S.D. from the average values. We can see that the error is relatively more sensitive to the change in E than in v.

The trajectory of the real needle and the simulated needle match to video resolution for both experiments. The root-mean-squared error of the marker positions over time in the single-bend experiment is 0.75 mm, with 88.3% of errors under 1 mm, and 97.8% of errors under 2 mm. The root-mean-squared error in the double-bend example is 1.3 mm, with 90.5% of errors under 2 mm, and 97.2% of errors under 3 mm. Figure 12.10 shows the histogram of the marker tracking error of both experiments. The maximum error over all markers and all time steps for the single-bend experiment is 4.5 mm; for the double-bend, it is 5.3 mm. The simulation also matches the experiments qualitatively, as the companion video shows. Some of the marker error is attributable to the vision algorithm. It fails to track several markers (which we exclude from the videos), and several others are tracked incorrectly so they appear to jitter about. We also ran multiple simulations that varied the values of E and v up to 2 standard deviations to explore how sensitive the marker position errors were with respect to the changes in E and v. The resulting errors are shown in Tables 12.1 and 12.2. We can see that the error is relatively more sensitive to the change in E than in v. The error is also lower when v is slightly lower than $\bar{v}$ for both the single-bend and the double-bend experiments. The error is also lower when we set E to be slightly higher than $\bar{E}$ in the double-bend experiment.

12.1.4 Running Time

All timings in are on an 8-core 3.0 GHz Intel Xeon with 16 GB RAM. Table 12.3 gives the running times for the different simulations discussed in this section. Observe that speeds for the steerable needle simulation in prostate tissue range from about 7 frames per second on one core to about 25 frames per second on seven cores. The LCP solver dominates the running time, whereas the local

	$\bar{E}-2$ S.D.	$\bar{E}-$S.D.	$\bar{E}$	$\bar{E}+$S.D.	$\bar{E}+2$ S.D.
$\bar{v}-2$ S.D.	1.684	1.445	1.276	1.262	1.313
$\bar{v}-$S.D.	1.702	1.427	1.295	1.265	1.334
$\bar{v}$	1.721	1.468	1.309	1.290	1.340
$\bar{v}+$S.D.	1.762	1.465	1.308	1.304	1.344
$\bar{v}+2$ S.D.	1.758	1.511	1.365	1.426	1.406

Table 12.2: Sensitivity of the marker positions' error with respect to the changes in E and v, for the double-bend experiment. E and v are varied from -2 S.D. to $+2$ S.D. from the average values. We can see that the error is relatively more sensitive to the change in E than in v.

remeshing step takes a negligible amount of time. Table 12.4 shows the number of LCP trials per time step and the number of conjugate gradient iterations per linear solution for each example.

12.2 Applications

Our simulator has successfully been used in various surgical planning research as described briefly below.

12.2.1 Feedback Control for Steerable Needles on Helical Paths in 3D Deformable Tissue

During the writing of this thesis, members of our project group have developed feedback control software that uses our simulator to maneuver a steerable needle to a target in a deformable tissue. Figure 12.11 shows several states during needle insertion. The light blue helical paths are trajectories chosen by the planner at different instants in time.

The planner assumes that the needle will be inserted with constant velocity and constant base rotation speed which would result in a helical path if the tissue did not deform. Given the current position and orientation of the needle tip, the planner determines the helical path that intersects the target or is closest to the target using branch-and-bound and interval analysis techniques. It then uses the corresponding insertion speed and rotation speed as the control signal. In the next time step, due to the tissue deformation (computed by our simulator), and the uncertainty in measure-

Name	Threads	Total ms	LCP	Tissue Simulation	Needle Simulation	Remeshing
Prostate, Flexible Needle	1	130.9	108.8	13.4	1.3	0.5
Prostate, Flexible Needle	2	77.7	62.3	7.7	1.6	0.5
Prostate, Flexible Needle	3	64.1	51.1	4.7	1.4	0.4
Prostate, Flexible Needle	4	56.6	44.8	3.9	1.4	0.7
Prostate, Flexible Needle	5	47.6	36.7	3.3	1.8	0.4
Prostate, Flexible Needle	6	39.6	28.8	3.0	1.4	0.5
Prostate, Flexible Needle	7	38.5	28.3	2.2	1.4	0.3
Prostate, Stiff Needle	7	38.2	28.7	2.1	1.1	0.5
Two Cylinders	7	24.0	14.4	1.9	0.6	0.9
single-bend	7	22.8	13.2	1.1	0.4	0.9
double-bend	7	33.0	23.6	1.1	0.5	0.5

Table 12.3: Timings (ms) for several examples with different numbers of threads. Examples include the prostate mesh with flexible bevel-tip needle or stiff symmetric-tip needle, the two-cylinder example, and the tissue phantom verification experiments with single-bend and double-bend. The frame time (Total) is divided into the LCP solution (LCP), the force and Jacobian computations for tissue (Tissue) and needle (Needle), and remeshing and needle reparameterization (Remesh). The number of tetrahedra/vertices for prostate, two cylinders, and tissue phantoms are 13,375/2,763, 3,248/813, and 2,280/672, respectively.

ments, the needle position and orientation may deviate from the predicted result. The plan is then recomputed and the new control signal would be used to make trajectory corrections. This feedback loop is repeated until the tip reach the target or the distance of the tip to the target can no longer be decreased. The detailed description of the algorithm and the discussion about its accuracy can be found in the paper by Hauser et al. [28].

12.2.2 Guiding Medical Needles with Single-Point Tissue Manipulation

Our simulator is also used in the work on guiding the medical needle using an external manipulation. It is an 3D extension of a work on 2D needle guiding by Torabi et al. [69].

The paper discusses the problem of steering a rigid needle during prostate brachytherapy by pushing the tissue externally using various types of manipulators. It considers moving a single manipulator

	LCP			CG		
Name	Min	Average	Max	Min	Average	Max
Prostate, Flexible Needle	1	1.41	10	18	205	652
Prostate, Stiff Needle	1	1.23	4	12	225	536
Two Cylinders	1	1.02	2	42	213	474
single-bend	1	1.17	4	46	234	343
double-bend	1	1.04	3	54	413	686

Table 12.4: Minimum, average, maximum number of trials required for the LCP solver and number of conjugate gradient (CG) iterations for each linear solution for examples running on seven threads.

at a time. The algorithm proceeds in phases, where in each phase, the controller tries to move the position of one point inside a tissue, so called *point of interest*, onto the needle path. In the initial phases, the goal is to move sensitive tissues away from the needle path to avoid them being damaged by the needle. In the final phase, the target location is the point of interest and is moved onto the needle path. In each phase, the controller determines the force at the manipulator based on the manipulator force profile and the tissue model and uses PI feedback to attempt to reduce the distance between the point of interest and the needle path.

The algorithm uses a stochastic optimization to pick the needle path that minimizes the cost of the plan which takes into consideration the amount of force required to complete it. For each candidate needle path, the algorithm chooses the points of interest by intersecting a cylinder around the needle path with the geometry of the sensitive tissues and selects the points that locally maximize the penetration depth. It then selects the manipulation point and the type of manipulator to use for each point of interest by minimizing a cost metric.

Torabi and his colleagues are now working on using our simulator to extend the result to 3D for both rigid needle and steerable needle. The screenshots from a preliminary result are shown in Figure 12.12.

12.3 Future Work

There are several interesting ways to extend the versatility of our simulator. The biomedical community is considering needles with pre-bent tips, giving them a much smaller turning radius [53].

Because the bend is not at the needle tip, simulating the needle entails placing a mesh node at a fixed location in the needle that moves with the needle. This necessitates a change to our remeshing algorithm that makes it difficult to maintain high mesh quality. The pre-bent tip can damage the tissue near the tip when the needle is twisted. The damage is also likely to happen at a scale below what the simulation mesh can capture. To approximate the effect of the damage on the tissue deformation, the material properties of the tissue near the tip may need to be modified.

A second extension would be to use graded meshes, which would allow us to model a larger tissue yet have higher resolution near the needle. The tissue mesh would initially be relatively coarse. When the needle comes into contact with the tissue, the tissue mesh is then refined near the needle tip to be able to capture the delicate interactions of the tissue and the needle. When the needle is retracted, the tissue mesh is coarsened again to speed up the running time and reduce memory usage. The difficulty of this extension is to deal with the interaction between the tissue mesh refinement and the local remeshing algorithm while ensuring a good mesh quality.

A third extension would be to model the torsional stiffness of the needle and the torsional friction that couples a twisting needle to the surrounding tissue. These are responsible for the lag in the amount of twist of the needle tip compared to the needle base observed by Webster et al. [71]. Currently, we treat only sliding friction. To treat torsional friction as well, we would have to reformulate the force computation and take torque around the needle shaft into account. One may also need to introduce the torsional friction state at each of the shared nodes.

A fourth extension would be to extend the remeshing to support the simulation of a procedure in which several needles are inserted to fix the tissue in place. Our simulator can already handle simulating multiple needles that never touch common elements or are inserted one at a time and retracted in the reverse order. More general uses would require a more sophisticated remeshing algorithm that keeps track of the stack of operations for each needle separately. When a mesh modification operation is considered, one must check if the operation would invalidate any of the operation in the stacks. If so, either the operation must be disallowed or a modification to the invalidated stack must be done.

The techniques reported in this thesis may also apply to simulating surgical sutures and staples where similar one-dimensional structures penetrate into soft tissue which was considered in a recent work by Guébert et al. [27].

a)

b)

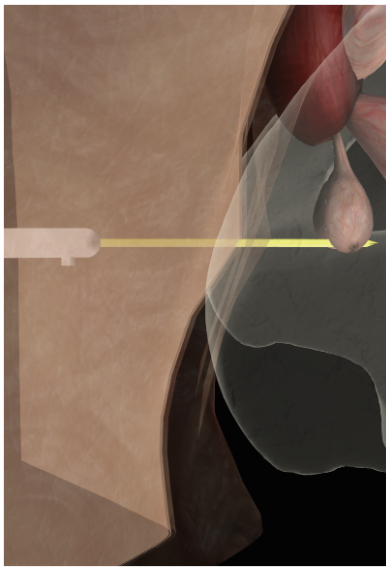

Figure 12.1: Screenshots from our prostate brachytherapy simulator. A needle is inserted from the left through the epidermis and dermis into the prostate gland. a) Bevel-tip flexible needle. b) Symmetric-tip stiff needle.

Figure 12.2: Screenshots of the animation of prostate brachytherapy with a bevel tip flexible needle. The tissue deformation is visualized with the gray lattice.

Figure 12.3: Screenshots of the animation of prostate brachytherapy with a symmetric tip stiff needle. The tissue deformation is visualized with the gray lattice.

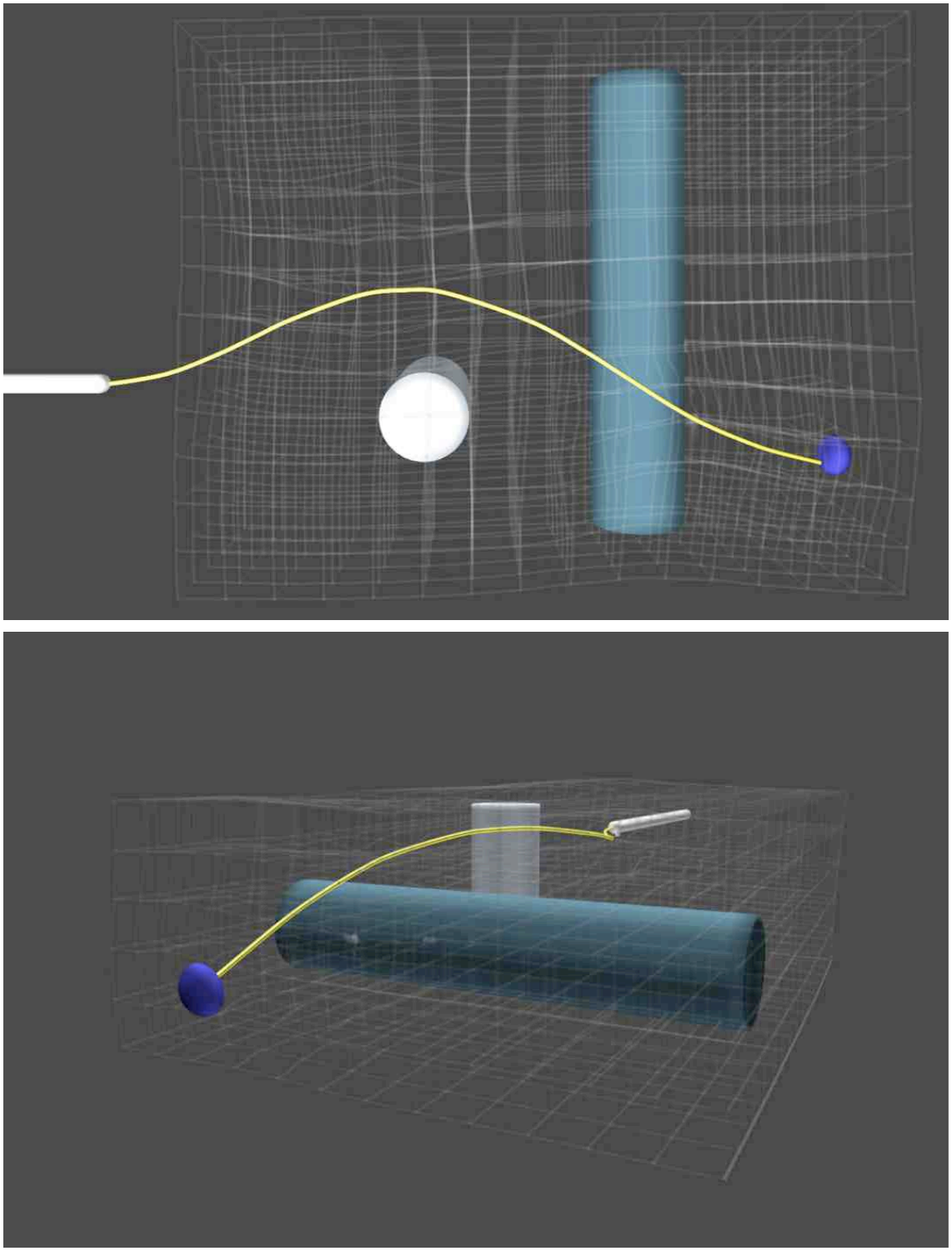

Figure 12.4: An example showing the bevel-tip flexible needle steering to avoid two fixed cylindrical obstacles. The base of the needle is rotated to change the direction of bending.

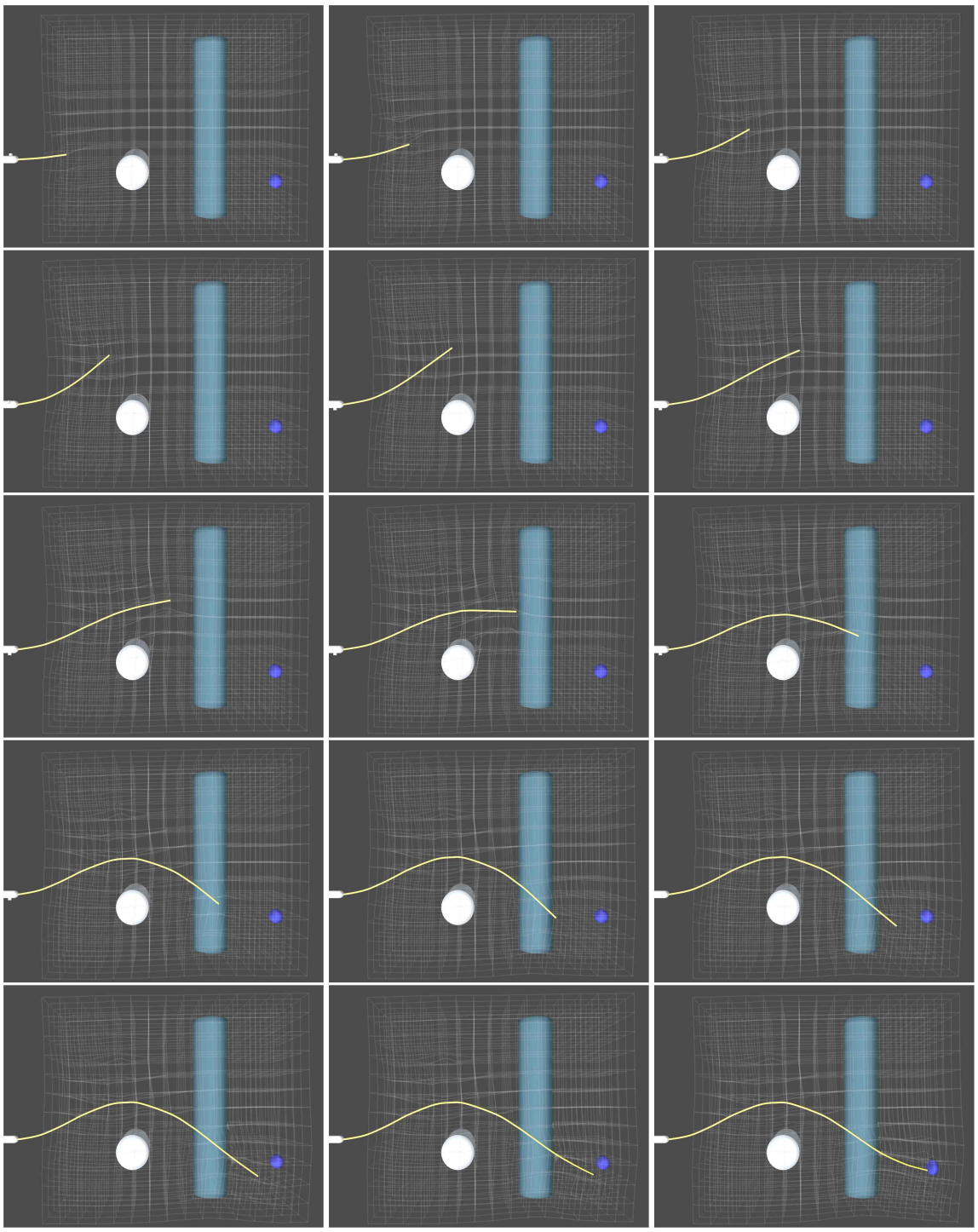

Figure 12.5: Screenshots of an animation of needle insertion into an artificial tissue with two embedded cylindrical obstacles (side view). The example demonstrates the ability of the bevel-tip steerable needle to avoid an obstacle. The tissue deformation is visualized with the gray lattice.

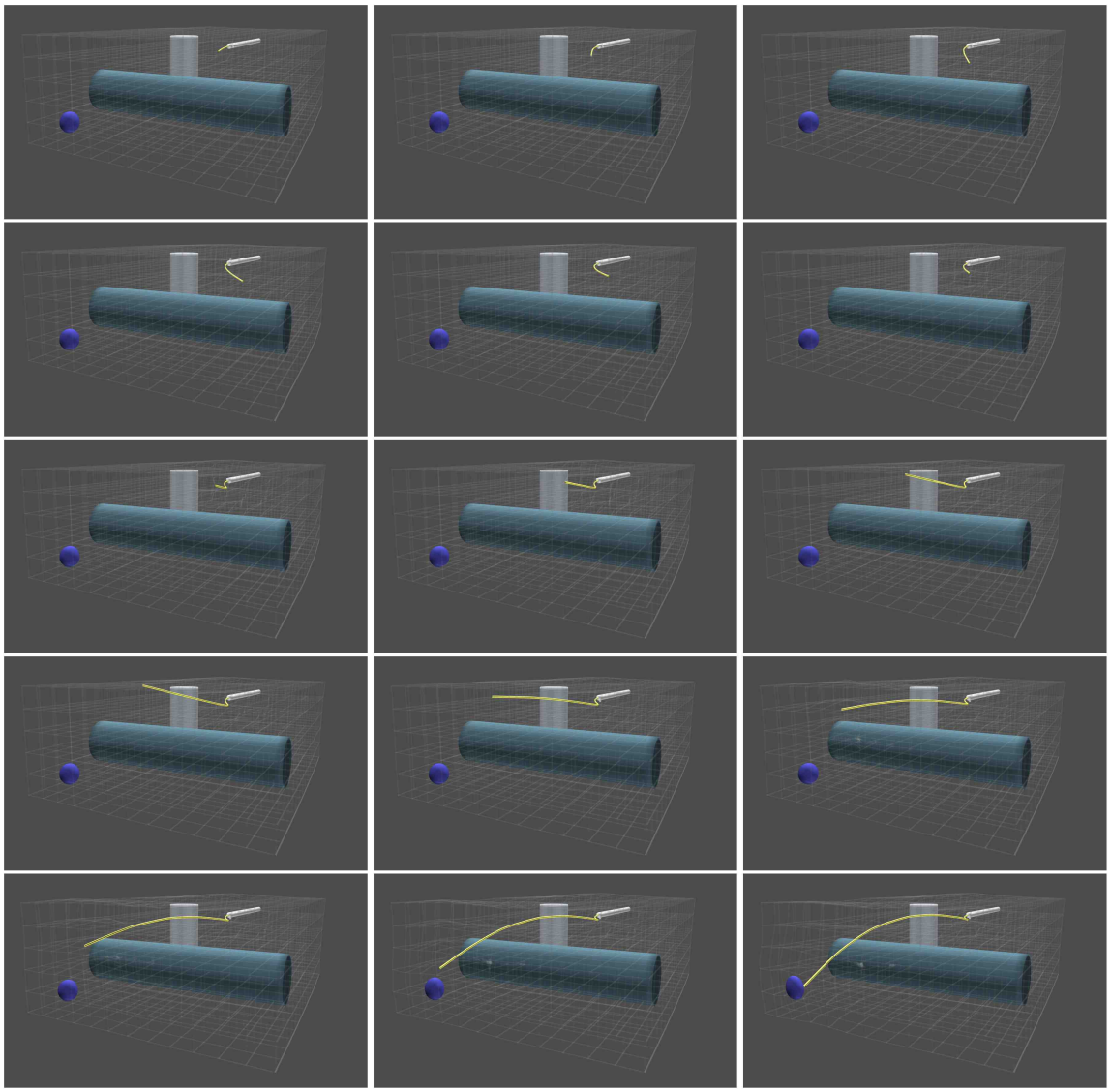

Figure 12.6: Screenshots of an animation of needle insertion into an artificial tissue with two embedded cylindrical obstacles (back view). The example demonstrates the ability of the bevel-tip steerable needle to avoid an obstacle. The tissue deformation is visualized with the gray lattice.

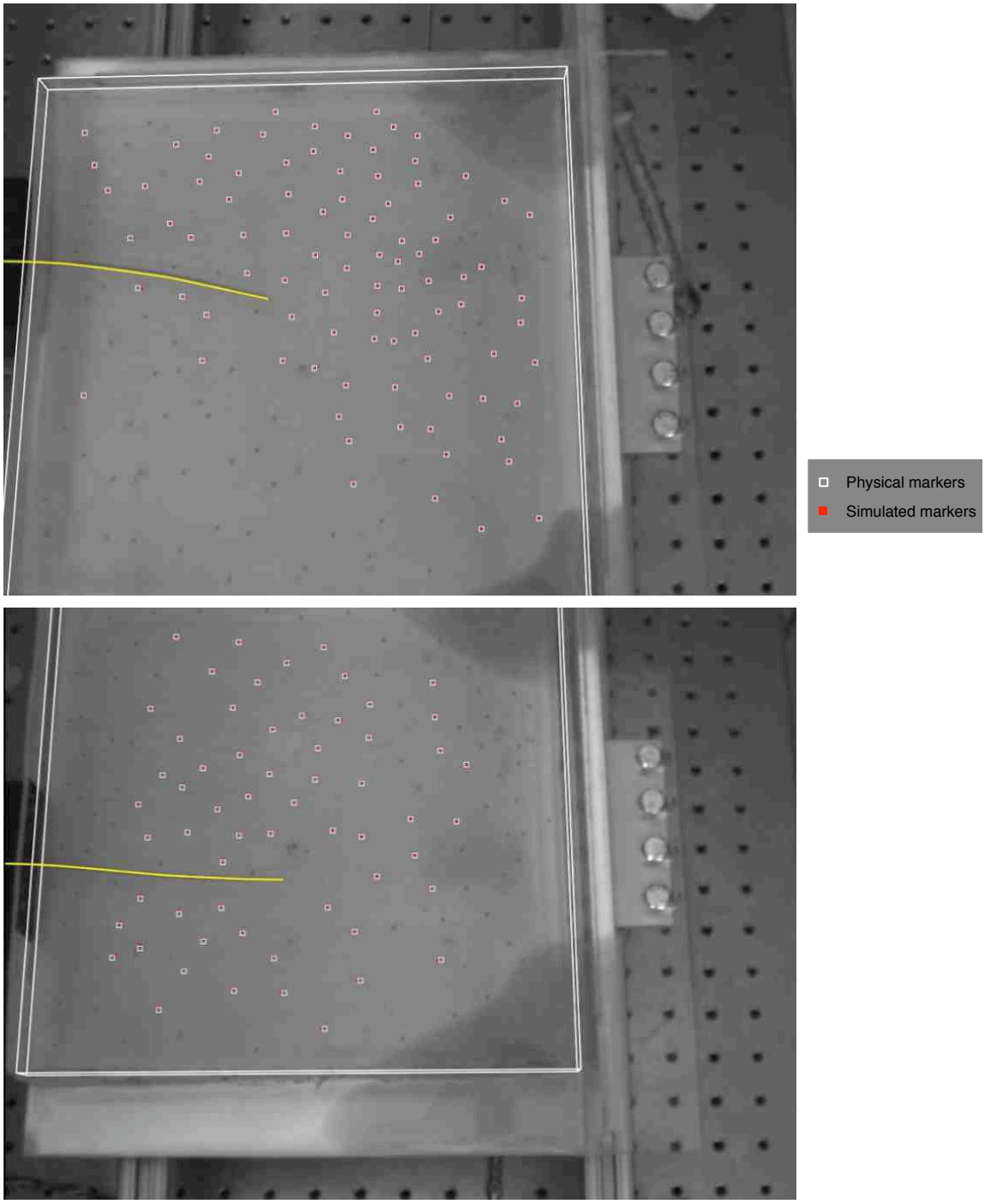

Figure 12.7: Comparison of our simulation with two experiments. The simulated needle appears in yellow. The white squares are the physical markers, and the red dots are the simulated markers. Our root-mean-squared simulation error (compared with the experiment) is 0.75 mm for the *single-bend* experiment (left), and 1.30 mm for the *double-bend* experiment (right). The *double-bend* experiment used the *single-bend* parameters without re-tuning.

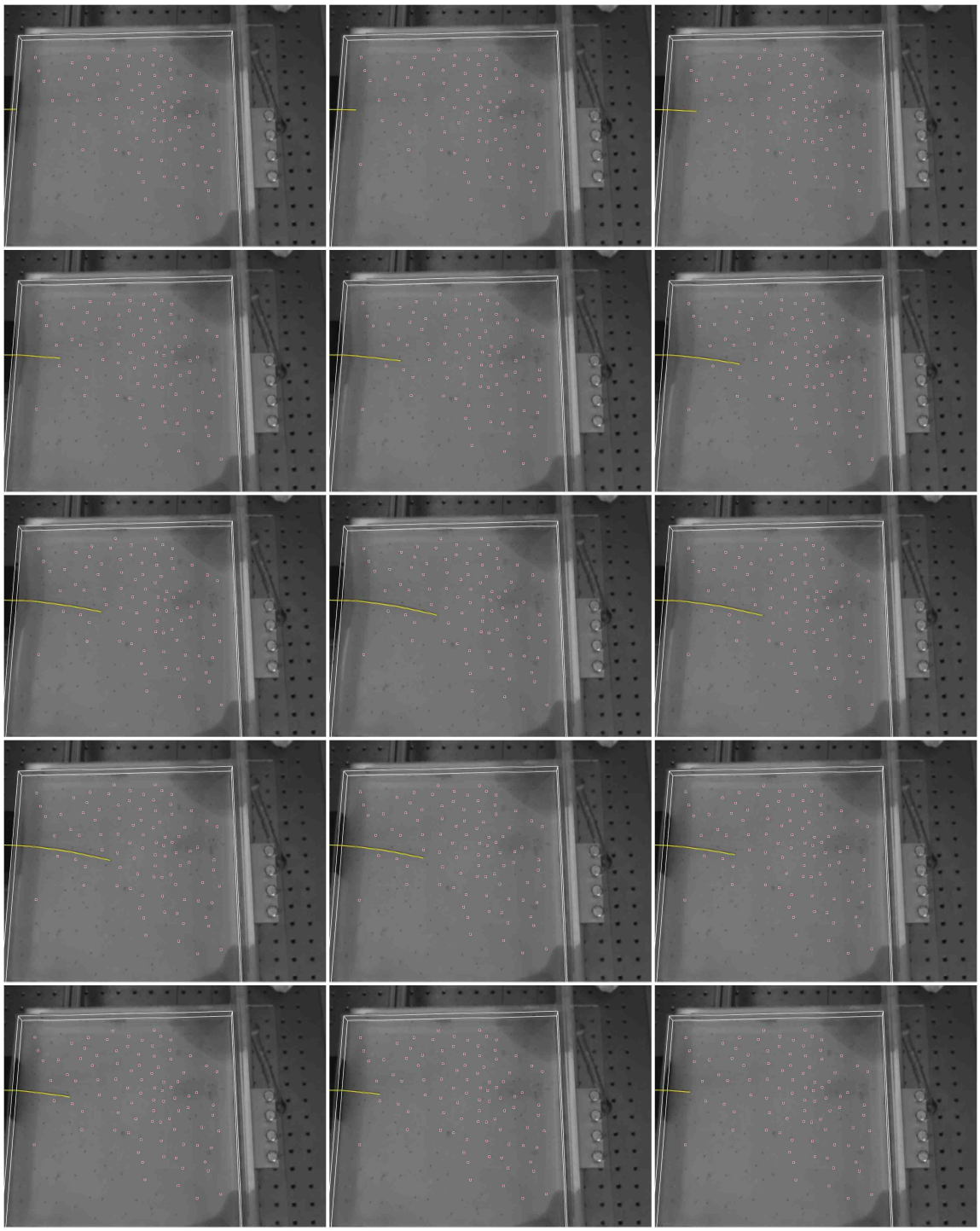

Figure 12.8: Screenshots of our simulator result rendered on top of a real-world experiment of inserting a bevel tip needle into a gel phantom.

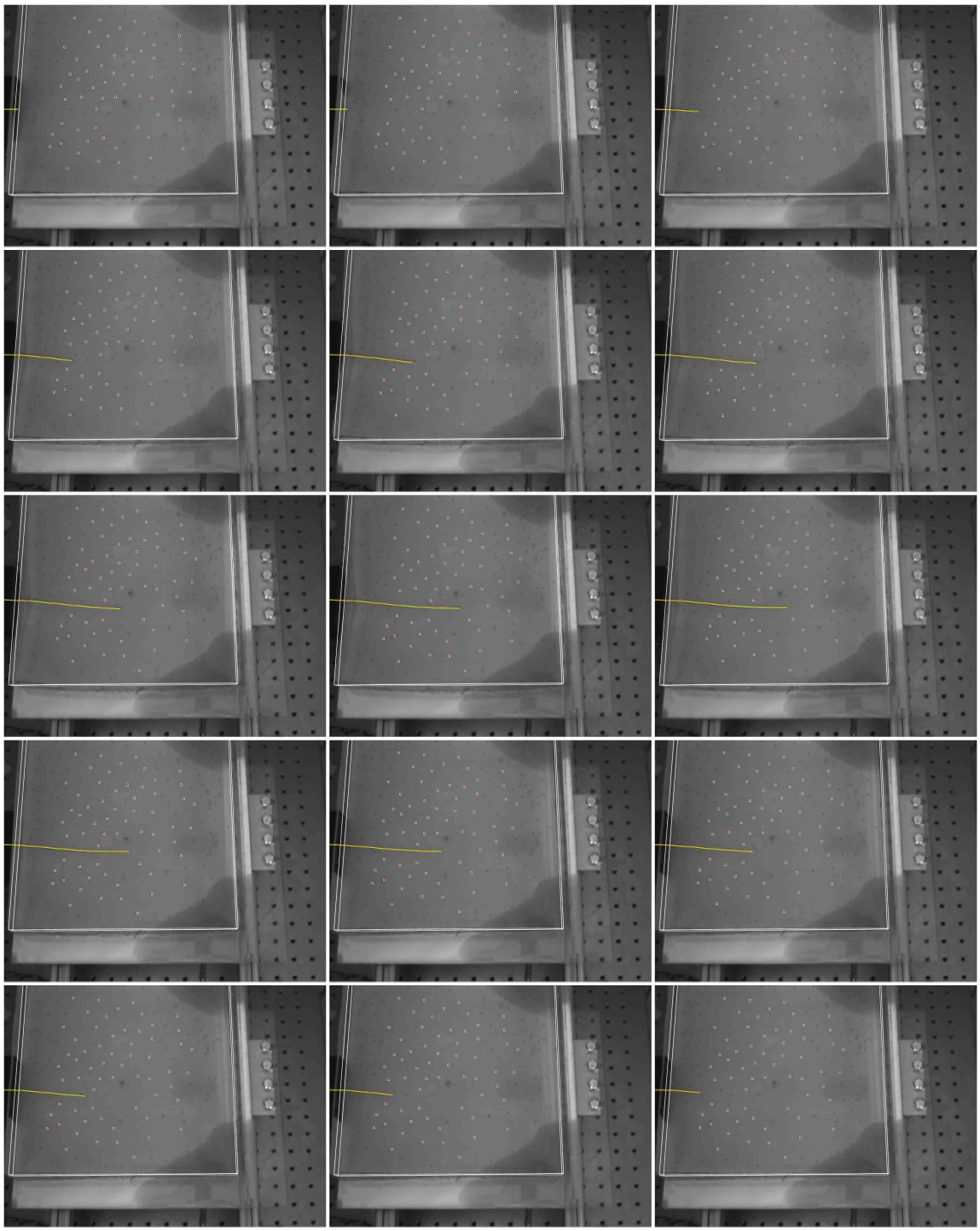

Figure 12.9: Screenshots of our simulator result rendered on top of a real-world experiment of inserting a bevel tip needle into a gel phantom. The needle is twisted 180° halfway through insertion.

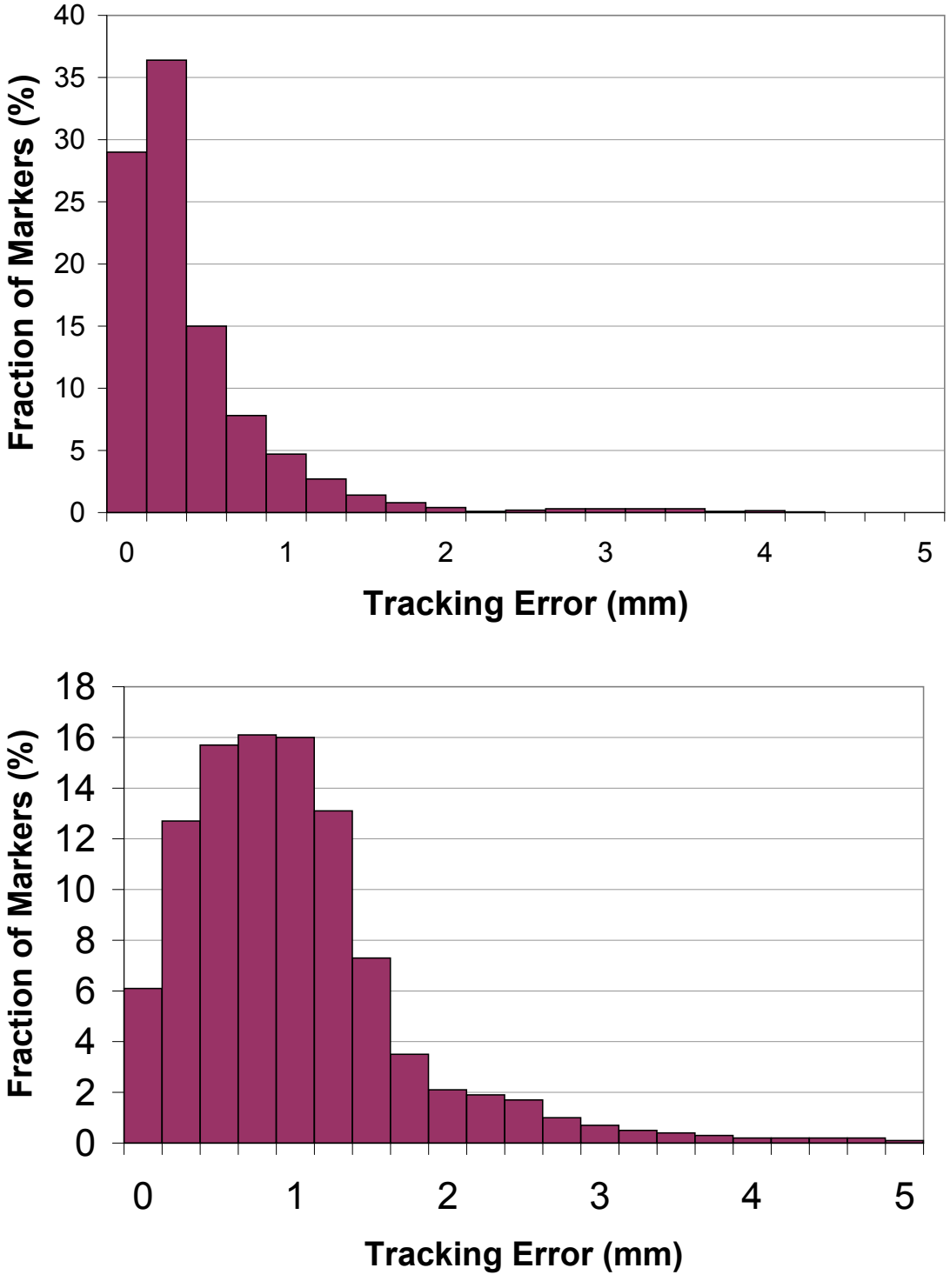

Figure 12.10: Histogram of the marker tracking error for the single-bend (top) and the double-bend (bottom) needle insertion experiments.

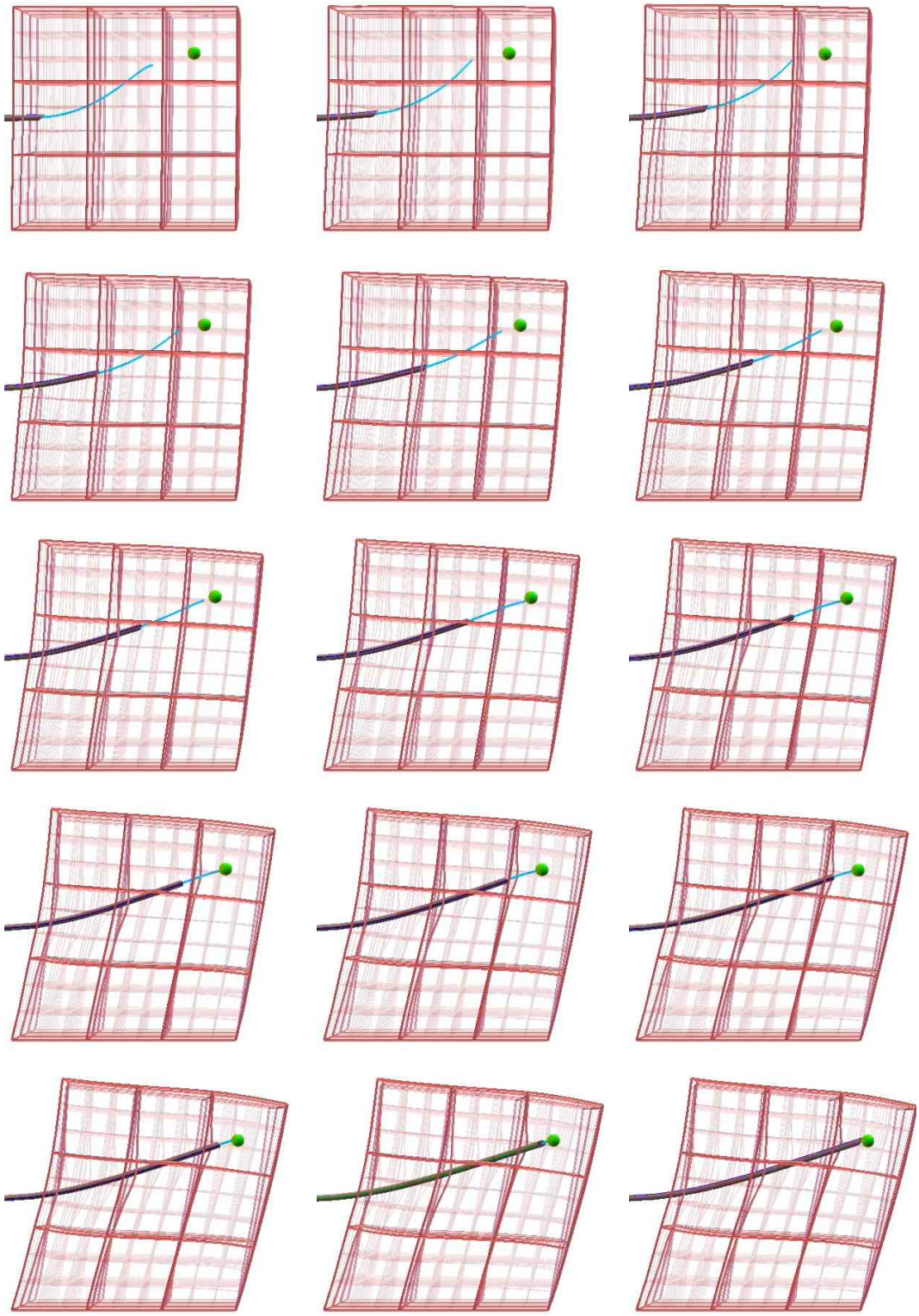

Figure 12.11: Screenshots of an animation showing how the planning algorithm finds a path by which a steerable needle can reach a target. The thin blue curve shows the current predicted trajectory, ignoring tissue deformation.

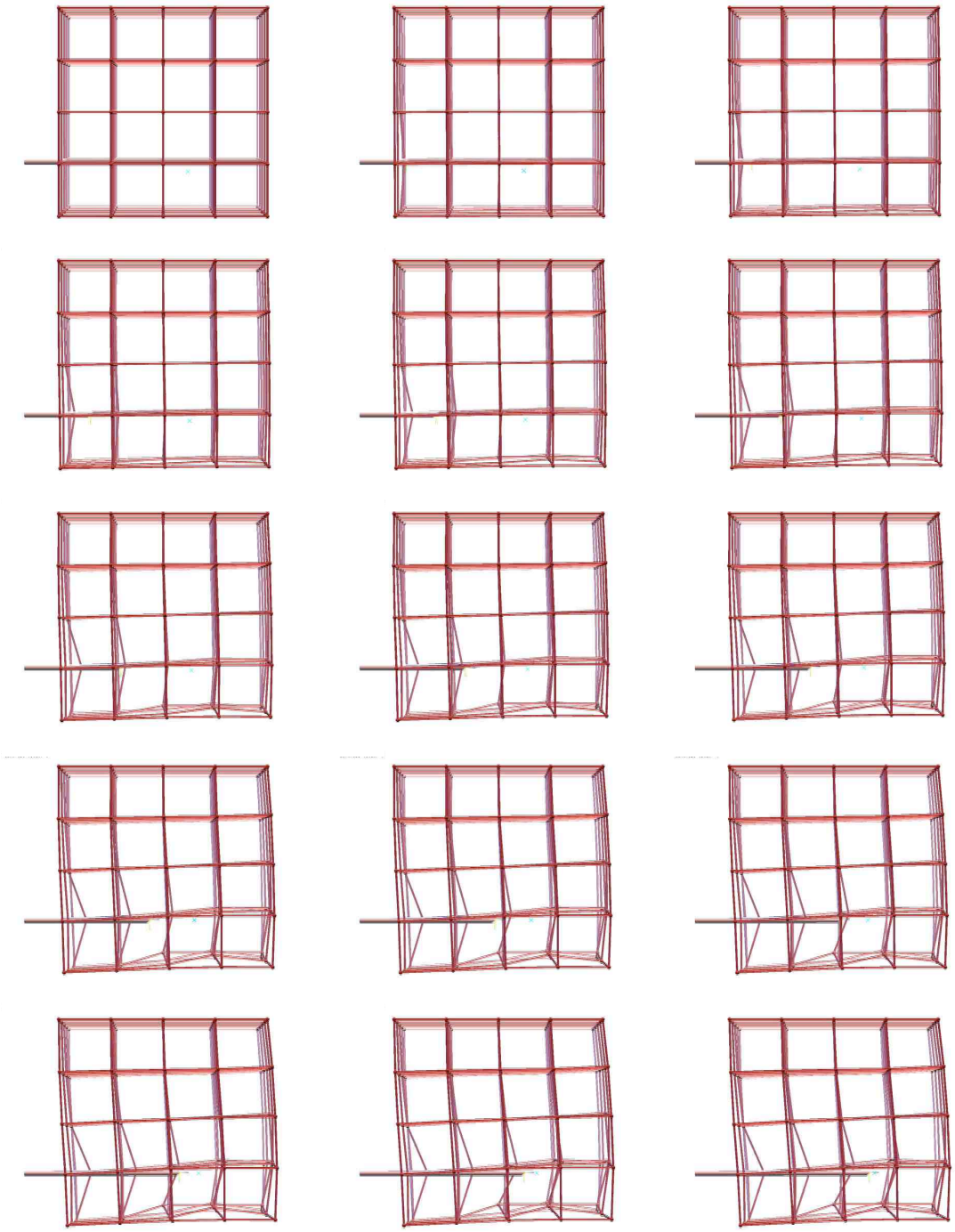

Figure 12.12: A planning algorithm computes the needle insertion location, the manipulation location and the force that should be applied to deform the tissue externally so that the needle insertion can reach the target. The figure shows the screenshots during a simulation where the tissue is manipulated at a single point on the bottom right.

Bibliography

[1] Niki Abolhassani, Rajni V. Patel, and Mehrdad Moallem. Needle insertion into soft tissue: A survey. *Medical Engineering & Physics*, 29(4):413–431, May 2007.

[2] Hiroshi Akima. A new method of interpolation and smooth curve fitting based on local procedures. *J. ACM*, 17(4):589–602, October 1970.

[3] Jérémie Allard, Stéphane Cotin, Franćois Faure, Pierre-Jean Bensoussan, Franćois Poyer, Christian Duriez, Hervé Delingette, and Laurent Grisoni. SOFA—An open source framework for medical simulation. In *Medicine Meets Virtual Reality 15*, pages 13–18. IOS Press, February 2007.

[4] Ron Alterovitz and Ken Goldberg. *Motion Planning in Medicine: Optimization and Simulation Algorithms for Image-Guided Procedures*, volume 50 of *Springer Tracts in Advanced Robotics*. Springer, Berlin, Germany, 2008.

[5] Ron Alterovitz, Ken Goldberg, and Allison M. Okamura. Planning for steerable bevel-tip needle insertion through 2D soft tissue with obstacles. In *IEEE International Conference on Robotics and Automation*, pages 1652–1657, April 2005.

[6] Ron Alterovitz, Jean Pouliot, Richard Taschereau, I-Chow Hsu, and Ken Goldberg. Simulating needle insertion and radioactive seed implantation for prostate brachytherapy. In *Medicine Meets Virtual Reality 11*, pages 19–25. IOS Press, January 2003.

[7] Ron Alterovitz, Thierry Siméon, and Ken Goldberg. The stochastic motion roadmap: A sampling framework for planning with Markov motion uncertainty. In *Robotics: Science and Systems III*, pages 233–241, June 2007.

[8] S. S. Antman. *Nonlinear Problems of Elasticity*. Springer Verlag, 1995.

[9] J. A. Bærentzen and H. Aanæs. Computing discrete signed distance fields from triangle meshes. Technical report, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, 2002.

[10] David Baraff. An introduction to physically based modeling, rigid body simulation i - unconstrained rigid body dynamics.

[11] Miklós Bergou, Max Wardetzky, Stephen Robinson, Basile Audoly, and Eitan Grinspun. Discrete elastic rods. *ACM Transactions on Graphics*, 27(3):63:1–63:12, August 2008.

[12] Florence Bertails. Linear time super-helices. *Computer Graphics Forum*, 28(2):417–426, April 2009.

[13] Florence Bertails, Basile Audoly, Marie-Paule Cani, Bernard Querleux, Frédéric Leroy, and Jean-Luc Lévêque. Super-helices for predicting the dynamics of natural hair. *ACM Transaction on Graphics*, 25(3):1180–1187, July 2006.

[14] M. Cenk Çavuşoğlu, Tolga G. Goktekin, Frank Tendick, and Shankar Sastry. GiPSi: An open source/open architecture software development framework for surgical simulation. In *Medicine Meets Virtual Reality 12*, pages 46–48, 2004.

[15] Nuttapong Chentanez, Ron Alterovitz, Daniel Ritchie, Lita Cho, Kris K. Hauser, Ken Goldberg, Jonathan R. Shewchuk, and James F. O'Brien. Interactive simulation of surgical needle insertion and steering. In *Proceedings of ACM SIGGRAPH 2009*, pages 88:1–10, Aug 2009.

[16] Robert D. Cook. *Finite Element Modeling for Stress Analysis*. John Wiley & Sonsl, 1995.

[17] Jessica R. Crouch, Chad M. Schneider, Josh Wainer, and Allison M. Okamura. A velocity-dependent model for needle insertion in soft tissue. In *Medical Image Computing and Computer-Assisted Intervention*, volume 3749 of *LNCS*, pages 624–632. Springer, Berlin, October 2005.

[18] E. Cuthill and J. McKee. Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 24th National Conference*, pages 157–172, New York, 1969. ACM.

[19] Ehsan Dehghan and Septimiu E. Salcudean. Needle insertion point and orientation optimization in non-linear tissue with application to brachytherapy. In *2007 IEEE International Conference on Robotics and Automation*, pages 2267–2272, Rome, April 2007.

[20] Simon P. DiMaio and S. E. Salcudean. Needle insertion modelling and simulation. *IEEE Transactions on Robotics and Automation*, 19(5):864–875, October 2003.

[21] Simon P. DiMaio and S. E. Salcudean. Needle steering and motion planning in soft tissues. *IEEE Transactions on Biomedical Engineering*, 52(6):965–974, June 2005.

[22] Carlos A. Felippa. A systematic approach to the element-independent corotational dynamics of finite elements. Technical Report CU-CAS-00-03, Center for Aerospace Structures, Colorado, 2000.

[23] Y. C. Fung. *A First Course in Continuum Mechanics*. Prentice Hall, 3rd edition, 1994.

[24] Anthony G. Gallagher, E. Matt Ritter, Howard Champion, Gerald Higgins, Marvin P. Fried, Gerald Moses, C. Daniel Smith, and Richard M. Satava. Virtual reality simulation for the operating room: Proficiency-based training as a paradigm shift in surgical skills training. *Annals of Surgery*, 241(2):364–372, February 2005.

[25] Orcun Goksel, Septimiu E. Salcudean, and Simon P. DiMaio. 3D simulation of needle-tissue interaction with application to prostate brachytherapy. *Computer Aided Surgery*, 11(6):279–288, November 2006.

[26] M. Grégoire and E. Schömer. Interactive simulation of one-dimensional flexible parts. *Proceedings of the 2006 Symposium on Solid and Physical Modeling*, June 2006.

[27] Christophe Guébert, Christian Duriez, Stéphane Cotin, Jérémie Allard, and Laurent Grisoni. Suturing simulation based on complementarity constraints. In *Proceedings of SCA 2009 (poster)*, August 2009.

[28] Kris Hauser, Ron Alterovitz, Nuttapong Chentanez, Allison Okamura, and Ken Goldberg. Feedback control for steering needles through 3D deformable tissue using helical paths. In *Robotics: Science and Systems V*, Seattle, Washington, June 2009.

[29] G. Irving, J. Teran, and R. Fedkiw. Invertible finite elements for robust simulation of large deformation. In *Proceedings of the 2004 Symposium on Computer Animation*, pages 131–140, Grenoble, France, 2004.

[30] Yasushi Ito, Alan M. Shih, and Bharat K. Soni. Reliable isotropic tetrahedral mesh generation

based on an advancing front method. In *International Meshing Roundtable*, pages 95–106, 2004.

[31] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, August 1998.

[32] Lily Kharevych, Patrick Mullen, Houman Owhadi, and Mathieu Desbrun. Numerical coarsening of inhomogeneous elastic materials. In *SIGGRAPH '09: ACM SIGGRAPH 2009 papers*, pages 1–8, New York, NY, USA, 2009. ACM.

[33] Bryan Matthew Klingner and Jonathan Richard Shewchuk. Aggressive tetrahedral mesh improvement. In *Proceedings of the 16th International Meshing Roundtable*, pages 3–23, Seattle, Washington, October 2007.

[34] L. T. Kohn, J. M. Corrigan, and M. S. Donaldson. *To Err Is Human: Building a Safer Health System*. New York: National Academy, 2000.

[35] T. A. Krouskop, T. M. Wheeler, F. Kallel, B. S. Garria, and T. Hall. Elastic moduli of breast and prostate tissues under compression. *Ultrasonic Imaging*, 20(4):260–274, October 1998.

[36] François Labelle and Jonathan Richard Shewchuk. Isosurface stuffing: Fast tetrahedral meshes with good dihedral angles. *ACM Transactions on Graphics*, 26(3):57:1–57:10, July 2007.

[37] Randall J. Leveque. *Finite Volume Methods for Hyperbolic Problems*. Cambridge University Press, 2002.

[38] Alex Lindblad and George Turkiyyah. A physically-based framework for real-time haptic cutting and interaction with 3D continuum models. In *Proceedings of the 2007 Symposium on Solid and Physical Modeling*, pages 421–429, New York, June 2007. ACM.

[39] Achim Loock and Elmar Schömer. A virtual environment for interactive assembly simulation: From rigid bodies to deformable cables. In *5th World Multiconference on Systemics, Cybernetics and Informatics*, pages 325–332, July 2001.

[40] Maud Marchal, E. Promayon, and J. Troccaz. Simulating prostate surgical procedures with a discrete soft tissue model. In *Third Eurographics Workshop in Virtual Reality Interactions and Physical Simulations*, pages 109–118, November 2006.

[41] Roummel F. Marcia. On solving sparse symmetric linear systems whose definiteness is unknown. *Applied Numerical Mathematics*, 58(4):449–458, April 2008.

[42] Matthias Müller and Markus H. Gross. Interactive virtual materials. In *Graphics Interface 2004*, pages 239–246, London, Ontario, Canada, 2004.

[43] Han-Wen Nienhuys and A. Frank van der Stappen. A surgery simulation supporting cuts and finite element deformation. In *Medical Image Computing and Computer-Assisted Intervention, 4th International Conference*, pages 153–160, October 2001.

[44] Han-Wen Nienhuys and A. Frank van der Stappen. Interactive needle insertions in 3D nonlinear material. Technical Report UU-CS-2003-019, Institute of Information and Computing Sciences, Utrecht University, 2003.

[45] James F. O'Brien and Jessica K. Hodgins. Graphical modeling and animation of brittle fracture. In *Computer Graphics (SIGGRAPH '99 Proceedings)*, pages 137–146, New York, August 1999. ACM Press.

[46] M. D. O'Leary, C. Simone, T. Washio, K. Yoshinaka, and A. M. Okamura. Robotic needle insertion: Effects of friction and needle geometry. In *IEEE International Conference on Robotics and Automation*, pages 1774–1780, September 2003.

[47] Dinesh K. Pai. STRANDS: Interactive simulation of thin solids using Cosserat models. *Computer Graphics Forum*, 21(3):347–352, September 2002.

[48] C. C. Paige and Michael A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 12(4):617–629, September 1975.

[49] C. Paloc, A. Faraci, and F. Bello. Online remeshing for soft tissue simulation in surgical training. *IEEE Computer Graphics & Applications*, 26(6):24–34, November 2006.

[50] V. N. Parthasarathy, C. M. Graichen, and A. F. Hathaway. A comparison of tetrahedron quality measures. *Finite Elements in Analysis and Design*, 15(3):255–261, January 1994.

[51] Guillaume Picinbono, Hervé Delingette, and Nicholas Ayache. Non-linear anisotropic elasticity for real-time surgery simulation. *Graphical Models*, 65(5):305–321, September 2003.

[52] Cao D. Q., Liu Dongsheng, and Wang Charles. H. t. Three dimensional nonlinear dynamics

of slender structures: Cosserat rod element approach. *International Journal of Solids and Structures*, 43(3-4):760–783, 2006.

[53] Kyle B. Reed, Vinutha Kallem, Ron Alterovitz, Ken Goldberg, Allison M. Okamura, and Noah J. Cowan. Integrated planning and image-guided control for planar needle steering. In *Proceedings of the Second IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics*, pages 819–824, Scottsdale, Arizona, October 2008.

[54] K. Renze and J. Oliver. Generalized unstructured decimation. *IEEE Computer Graphics & Applications*, 16(6):24–32, 1996.

[55] Richard M. Satava. Identification and reduction of surgical error using simulation. *Minimally Invasive Therapy & Allied Technologies*, 14(4–5):257–261, September 2005.

[56] Neal E. Seymour, Anthony G. Gallagher, Sanziana A. Roman, Michael K. O'Brien, Vipin K. Bansal, Dana K. Andersen, and Richard M. Satava. Virtual reality training improves operating room performance: Results of a randomized, double-blinded study. *Annals of Surgery*, 236(4):458–463, October 2002.

[57] Jonathan Richard Shewchuk. Tetrahedral mesh generation by Delaunay refinement. In *Proceedings of the Fourteenth Annual Symposium on Computational Geometry*, pages 86–95, Minneapolis, Minnesota, June 1998. Association for Computing Machinery.

[58] Eftychios Sifakis, Tamar Shinar, Geoffrey Irving, and Ronald Fedkiw. Hybrid simulation of deformable solids. In *Proceedings of the 2007 Symposium on Computer Animation*, pages 81–90, 2007.

[59] J. C. Simo. A finite strain beam formulation. The three-dimensional dynamic problem. *Computer Methods in Applied Mechanics and Engineering*, 49:55–70, 1985.

[60] C. Simone and A. M. Okamura. Modeling of needle insertion forces for robot-assisted percutaneous therapy. In *IEEE International Conference on Robotics and Automation*, pages 2085–2091, May 2002.

[61] J. Spillmann and M. Teschner. CORDE: Cosserat rod elements for the dynamic simulation of one-dimensional elastic objects. In *Proceedings of the 2007 Symposium on Computer Animation*, pages 63–72. Eurographics Association, 2007.

[62] Jonas Spillmann and Matthias Teschner. An adaptive contact model for the robust simulation of knots. *Computer Graphics Forum*, 27(2):497–506, April 2008.

[63] R. Taschereau, J. Pouliot, J. Roy, and D. Tremblay. Seed misplacement and stabilizing needles in transperineal permanent prostate implants. *Radiotherapy and Oncology*, 55(1):59–63, April 2000.

[64] Russel H. Taylor. A perspective on medical robotics. *Proceedings of the IEEE*, 94(9):1652–1664, September 2006.

[65] J. Teran, S. Blemker, V. Ng Thow Hing, and R. Fedkiw. Finite volume methods for the simulation of skeletal muscle. In *the ACM SIGGRAPH Symposium on Computer Animation*, pages 68–74, July 2003.

[66] Demetri Terzopoulos and Kurt Fleischer. Modeling inelastic deformation: viscolelasticity, plasticity, fracture. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 269–278, New York, NY, USA, 1988. ACM.

[67] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. In *Computer Graphics (SIGGRAPH '87 Proceedings)*, pages 205–214, 1987.

[68] Sivan Toledo. Taucs: A library of sparse linear solvers. http://www.tau.ac.il/ stoledo/taucs/, 2003.

[69] Maysam Torabi, Kris Hauser, Ron Alterovitz, Vincent Duindam, and Ken Goldberg. Guiding medical needles using single-point tissue manipulation. May 2009.

[70] Xiaogang Wang and Aaron Fenster. A virtual reality based 3D real-time interactive brachytherapy simulation of needle insertion and seed implantation. In *2004 IEEE International Symposium on Biomedical Imaging*, pages 280–283, April 2004.

[71] Robert J. Webster III, Noah J. Cowan, Gregory Chirikjian, and Allison M. Okamura. Nonholonomic modeling of needle steering. In *Proc. 9th International Symposium on Experimental Robotics*, June 2004.

[72] Robert J. Webster III, Jin Seob Kim, Noah J. Cowan, Gregory S. Chirikjian, and Allison M. Okamura. Nonholonomic modeling of needle steering. *Int. Journal of Robotics Research*, 25(5–6):509–525, May 2006.

[73] Robert J. Webster III, Jasenka Memisevic, and Allison M. Okamura. Design considerations for robotic needle steering. In *2005 IEEE International Conference on Robotics and Automation*, pages 3588–3594, Barcelona, Spain, April 2005.

[74] Robert J. Webster III, Allison M. Okamura, Noah J. Cowan, Gregory S. Chirikjian, Ken Goldberg, and Ron Alterovitz. Distal bevel-tip needle control device and algorithm. U.S. patent application number 11/436,995, May 2005.

[75] Martin Wicke, Daniel Ritchie, Bryan M. Klingner, Sebastian Burke, Jonathan R. Shewchuk, and James F. O'Brien. Dynamic local remeshing for elastoplastic simulation. In *Proceedings of ACM SIGGRAPH 2010*, pages 1–11, July 2010.