Modeling Realistic Tool-Tissue Interactions with Haptic Feedback: A Learning-based Method

Zachary Pezzementi zap@cs.jhu.edu
Daniel Ursu dursu1@jhu.edu
Sarthak Misra sarthak@jhu.edu
Allison M. Okamura [*] aokamura@jhu.edu

Engineering Research Center for Computer-Integrated Surgical Systems Technology (ERC-CISST)
Laboratory for Computational Science and Robotics (LCSR)
The Johns Hopkins University

Abstract

Surgical simulators present a safe, practical, and ethical method for surgical training. In order to enhance realism and provide the user with an immersive training experience, simulators should have the capability to provide haptic feedback to the user. High-fidelity surgical simulators also require accurate modeling of the interaction between surgical instruments and organs. Linear elasticity-based models are commonly used to simulate tool-tissue interaction due to computational considerations, although real soft tissues exhibit nonlinear viscoelastic behavior. In this paper, we use a learning algorithm to train a linear 2D mass-spring-damper system that behaves similarly to a high-fidelity nonlinear finite element (FE) model. The spring parameters are trained off-line using data from an FE simulation of brain tissue deformation using simultaneous perturbation stochastic approximation, a model-free optimization algorithm. The model is implemented in a real-time soft tissue simulator with haptic interaction provided through the PHANTOM Omni haptic device. Our model's response is significantly closer to the desired response of the FE model than that of a linear heuristic model.

Index Terms: I.6.5 [Computing Methodologies]: Simulation and Modeling—Model Development; H.5.2 [Information Systems]: Information Interfaces and Presentation—User InterfacesHaptic I/O

1 Introduction

Advances in computer hardware and software have made it possible for computation to play a vital role in medicine. The use of efficient and high-fidelity graphics has revolutionized the way medical image data are accessed and reviewed. The same technology makes it possible to create realistic computer simulations of medical procedures. The objective of surgical simulators is to provide medical practitioners with an authentic, immersive virtual or augmented reality environment for the training, practice, and planning of surgical procedures. Accurate and realistic modeling of the interaction between surgical instruments and human organs has been recognized as a key requirement in the development of high-fidelity surgical simulators. Such advanced simulators should also have the capability of providing truthful haptic (force and tactile) feedback to the user.

Continuum mechanics provides a sound physics-based foundation for modeling biological tissues, while Finite Element (FE) modeling is a general computational technique to simulate responses of materials to loads and deformations. The current state of the art for modeling tool-tissue interaction dynamics in surgical simulations is given by nonlinear FE models. Implementations of these models, however, are prohibitively complex or expensive to compute at haptic update rates ($\sim$ 1 kHz), particularly when interactions involve cutting and tearing of tissue [21, 24], and thus mesh re-definition. Hence, most of the past studies within the robotics and haptics communities have assumed linear elastic behavior for modeling tissues. However, human tissues are in general inhomogeneous and anisotropic, and possess nonlinear viscoelastic properties [8]. Furthermore, the responses of linear and nonlinear models are significantly different even for simple tool-tissue interactions [12]. Researchers address the trade-off between accuracy and computational time by either making simplifying assumptions to FE models to allow them to run at interactive rates [5, 23] or adapting simpler models to more closely emulate the FE models [16]. Recent work has focused on applying learning methods to this process, adapting a mass-spring mesh to approximate the reaction observed in real tissue deformation tests, optimizing both the mesh topology [2] and spring stiffnesses [3, 14, 6]. Both simulated annealing [6, 14] and genetic algorithms [2, 3] have been applied to mesh optimization. Since real-world deformation data is not available in plentiful supply, FE models are sometimes used as an intermediary. That is, mass-spring-damper meshes are used to replicate FE model deformation behavior [2, 3, 14]. These approaches are ultimately limited, however, to representing a single linear subspace of the total deformation space.

Figure 1: An overview of modeling realistic tool-tissue interactions with haptic feedback using a learning-based method

In this paper, we propose a novel means of emulating nonlinear tissue behavior using a linear mass-spring-damper mesh with changing spring constants, designed to approximate the behavior of the tissue at multiple deformations. Figure 1 depicts the various aspects of our simulation model. We start by extracting FE simulation data based on experimental studies for various loading/displacement scenarios and provide them to a model-free stochastic learning algorithm. Multiple sets of tissue stiffness parameters are then trained on the local tissue deformation data using the optimization algorithm. Although each individual set of tissue stiffness parameters is linear, each is optimized to a limited range of inputs. Dynamically switching between these sets of tissue parameters results in a piecewise-linear approximation of the nonlinear characteristics of the overall actual tissue deformation. The optimized mesh data is applied towards the design of a prototype haptic tissue deformation simulator, intended to show the striking differences between the nonlinear and linear tissue deformation models. To our knowledge, no published work exists that couples nonlinear elasticity-based modeling to a learning algorithm and shows the feasibility of operation using a haptic device.

The rest of our paper is organized as follows: Section 2 describes the continuum mechanics theory governing nonlinear tissue defor-

[*]This work was supported in part by Johns Hopkins University and NIH grant # R01 EB002004.

Symposium on Haptic Interfaces for Virtual
Environments and Teleoperator Systems 2008
13-14 March, Reno, Nevada, USA
978-1-4244-2005-6/08/$25.00 ©2008 IEEE

mation and Finite Element (FE) simulations based on real tissue data. In Section 3, we describe our optimization learning method. The setup of the haptic simulation follows in Section 4. Finally in Section 5, we discuss the validity of our method and future work.

2 Models for Soft Tissue Deformation

In this paper, we present a system in which a high-fidelity, offline deformation model is used to train a lower-fidelity, interactive deformation model. Our first step is therefore to acquire realistic tool-tissue interaction data which can be used by a learning algorithm to train our interactive model. Ideally, data would be acquired from experimental studies on human tissue *in vivo*, recording tool force and tissue deformation. These tests would be performed at several locations on the organ with various penetration depths, tool forces, and loading conditions. Data could also be acquired *in vitro*, either through visual tracking of the displacement of markers embedded in the tissue [7, 10], or through FE simulations. While any such data extraction method would be appropriate, this work uses previous FE simulations based on compression tests done on porcine brain tissue by Miller et al. [11]. In this section, we briefly describe the theory used for modeling of soft tissues and the FE simulations done to generate the data used in learning.

2.1 Nonlinear Elasticity

Human organs in general are inhomogeneous, anisotropic, and exhibit nonlinear viscoelastic properties. Though linear elastic models are frequently used to model tissues for simulating surgical procedures, such models are only valid for materials undergoing small strains. In most surgical procedures, organs are being subjected to large strains. The behavior of materials undergoing large strains ($>$1%-2%) is described by the theory of nonlinear elasticity, e.g. hyperelastic models [8].

A hyperelastic material is characterized by the existence of a strain energy density function, $W(\mathbf{F})$. The stress in the material as a result of deformation can be obtained from

$$\mathbf{P} = \frac{\partial W(\mathbf{F})}{\partial \mathbf{F}}, \tag{1}$$

where $\mathbf{P}$ is the first Piola-Kirchhoff stress tensor and $\mathbf{F}$ is the deformation gradient tensor. The Cauchy stress tensor and first Piola-Kirchhoff stress tensor are related by

$$\mathbf{P}\mathbf{F}^T = J\boldsymbol{\sigma}, \tag{2}$$

where $J = \det(\mathbf{F})$. There are several formulations for the strain energy density function, and we choose to use the Mooney-Rivlin model, which is widely used to model soft tissues [15]. The Mooney-Rivlin model is given in terms of the principal invariants, I_i, for isotropic and incompressible materials as

$$W = C_1 (I_1 - 3) + C_2 (I_2 - 3), \tag{3}$$

where C_1 and C_2 are material constants. The principal invariants are defined in the terms of the right Cauchy-Green tensor, $\mathbf{C} = \mathbf{F}^T\mathbf{F}$, as

$$I_1 = \mathbf{C} : \mathbf{I}, \tag{4}$$

$$I_2 = \frac{1}{2}\left((\mathbf{C} : \mathbf{I})^2 - (\mathbf{C} : \mathbf{C})\right), \tag{5}$$

$$I_3 = \det \mathbf{C}. \tag{6}$$

A least squares fit to the indentor displacement and force data for porcine brain in [11] resulted in Mooney-Rivlin parameters of $C_1 = 263$ Pa and $C_2 = 491$ Pa, which are used in (3). This hyperelasticity-based nonlinear model is sufficient for the purpose of comparing simulation results obtained via the FE method and the learning algorithm.

2.2 Finite Element Simulations

The Mooney-Rivlin hyperelastic model described in (3) was used in FE simulations. The FE simulations were performed using ABAQUS [1], and the simulation output data were input to the learning algorithm. The dimensions of the two-dimensional model were 10 cm $\times$ 10 cm, with the bottom edge fully constrained. The FE nodes were placed 1 cm apart, which is consistent with the model that will be used for haptic rendering, as described in Section 4. The elements were quadrilateral and a medial axis algorithm was used for meshing, with quadratic interpolation basis functions to estimate data between nodes. In order to train the learning algorithm, several static FE simulation cases were conducted. The displacements were applied at the model nodes, and each of the nodal forces and displacements were recorded for every simulation case. Figure 2 depicts contour plots of two of the 18 loading cases used in this study. These cases consisted of forces applied at the midpoints and corners of each of the unfixed edges of the square. At each of the three midpoints, forces were applied perpendicular to the edge, and at both corners, forces were applied at 3 angles, horizontally, vertically, and at a 45° angle. At each angle, two different force magnitudes were used. These cases were considered to cover most major deformation modes a user is likely to encounter through interaction with the mesh.

2.3 Heuristic Model

Van Gelder shows in [22] that exact simulation of an isotropic elastic membrane with a linear mass-spring mesh is impossible, but proposes a heuristic varying according to the geometry of elements incident upon that edge. For mesh edge c with length $|c|$,

$$k_c = \frac{E_2 \sum_e \text{area}(T_e)}{|c|^2} \tag{7}$$

Here E_2 is the two-dimensional Young's modulus and e indexes each of the model elements (triangles) incident upon edge c. We extend this model to provide masses as well in the same manner as in [13], where the mass at point i is given by

$$m_i = \sum_j \frac{1}{4}\rho_j V_j, \tag{8}$$

where j indexes all model elements containing point i, and ρ_j and V_j denote density and volume of element j respectively.

3 Learning Algorithm for Selecting Model Parameters

Mass-spring-damper meshes are by definition linear systems and are thus unable to describe nonlinear deformations. However, using the same theory behind local linearization of nonlinear systems (linearizing a system around a node of interest and estimating the behavior of the system in the neighborhood of that node as linear), the nonlinear behavior of tissue deformation can be captured for local deformations around a linearized node. That is, a specific set of spring constant weights should accurately output the forces derived experimentally via FE analysis, around a small neighborhood of the displacement.

In order to obtain the optimized mass-spring-damper meshes for particular forces and displacements of interest, a model-free stochastic optimization algorithm is used. Model-free optimization algorithms can be implemented for arbitrarily complex systems and have been shown to have good convergeance properties [19]. The use of "model-free" is to be taken literally in the sense that no hidden or implicit modeling is required, which eliminates the system characterization and identification processes, and thus the need to allocate time and resources to determine an adequate model of the

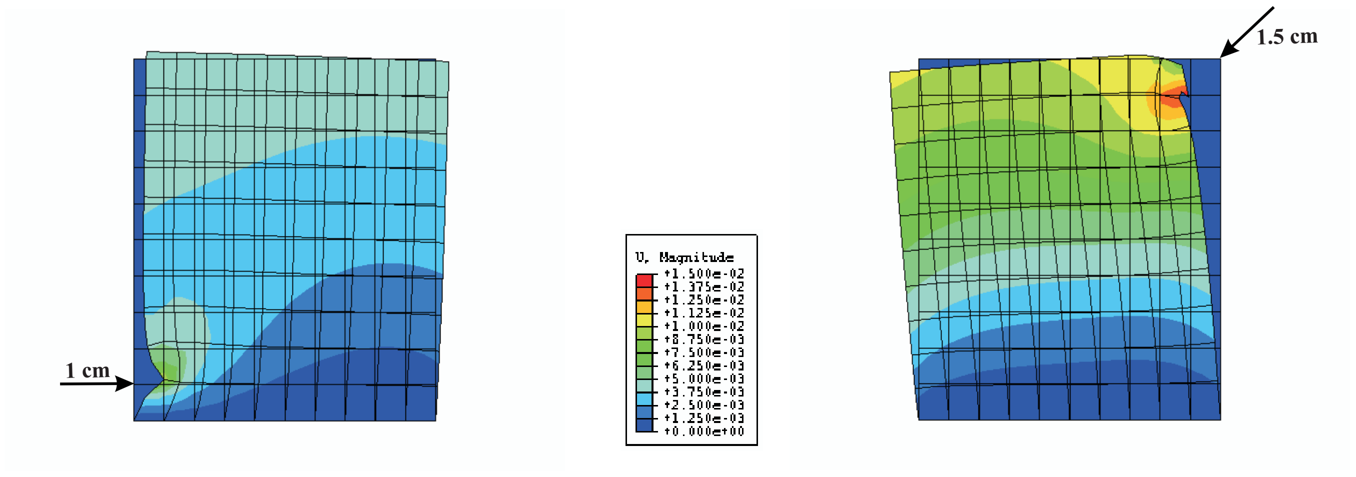

Figure 2: Example FE simulation contour plots used to train the learning algorithm: (left) Displacement of 1 cm magnitude applied horizontally to a node on the left edge (right) displacement of 1.5 cm magnitude applied to the right-corner node at an angle of 45°. (center) Legend which describes the color coding of the figures, which represents the displacement of the elements. Both plots are overlaid on the undeformed model.

underlying system and evaluate its validity. Two major advantages of the model-free approach over traditional optimization algorithms are that it (1) tends to better handle changes in the underlying system as it is not tied to a prior model, and (2) tends to be more robust in the case of widely varying control inputs. However, care must be taken to ensure that the mathematical solution obtained makes physical sense and does not result in unrealistic system parameters. Such constraints can be imposed either on the minimization (loss) function of the optimization algorithm via penalty methods [19], or by limiting the domain from which the algorithm is allowed to choose its solution set. In this work, the only parameters optimized were the spring weights, but the optimization could be extended to include the node masses or damping constants without difficulty.

Simultaneous Perturbation Stochastic Approximation (SPSA) was chosen as the model-free optimization algorithm because of its flexibility. The algorithm is based on the principle of gradient estimation of a loss function, $\mathscr{L}(\boldsymbol{\theta}_k)$, whose evolution in time is unknown, where $\boldsymbol{\theta} \in \mathbb{R}^n$ are the parameters to be optimized. For a set number of iterations, k, these parameters are perturbed in random directions using a random symmetric probability mass function whose value around 0 is 0. Loss functions, $\mathscr{L}(\boldsymbol{\theta}_{+k})$ and $\mathscr{L}(\boldsymbol{\theta}_{-k})$, corresponding to each perturbed parameter state are computed at each iteration. Then a gradient estimate is created by subtracting the two loss functions, $\mathscr{L}(\boldsymbol{\theta}_{+k})$ and $\mathscr{L}(\boldsymbol{\theta}_{-k})$, and dividing by the random noise elements used to perturb the system. The procedure is followed for k iterations or until some minimization goal, $\boldsymbol{\theta}^* \in \mathbb{R}^n$, has been reached. For an in-depth discussion of SPSA, the reader is referred to [20].

In order to optimize the mass-spring-damper mesh, the elements, $\boldsymbol{\theta} \in \mathbb{R}^{220}$, were chosen to be the vector of 220 spring constants of the 11×11 node 2-D system described in Section 4. The data for optimization was obtained from FE simulations of brain tissue at specific displacements in the X and Y directions, $\mathbf{u}(x,y)$. The loss function to be minimized,

$$\mathscr{L}(\boldsymbol{\theta}_k) = \sqrt{\left(\mathbf{f}_{\text{opt}} - \mathbf{f}_{\text{act}_k}\right)^T \mathbf{Q} \left(\mathbf{f}_{\text{opt}} - \mathbf{f}_{\text{act}_k}\right)}, \tag{9}$$

was designed as a weighted root mean square (RMS) error function between $\mathbf{f}_{\text{act}_k}$ and $\mathbf{f}_{\text{opt}}$, where $\mathbf{f}_{\text{act}_k}$ is the force output of the spring matrix, $\mathbf{K}$, with actual spring constants $\boldsymbol{\theta}_k$ at iteration k, at the specific displacement $\mathbf{u}(x,y)$

$$\mathbf{f}_{\text{act}_k} = \mathbf{K}(\boldsymbol{\theta}_k)\,\mathbf{u}(x,y). \tag{10}$$

$\mathbf{f}_{\text{opt}}$ is the force solved by ABAQUS for that same displacement. In (9), $\mathbf{Q}$ is a diagonal matrix of entries ranging from 0 to 1, representing the relative importance of some weights in influencing the output over others. Specifically, the entries of $\mathbf{Q}$ were chosen such that the springs in the neighborhood of the displaced mesh region are more important than springs farther away from the region. As such, the 3×3 grid of springs in the immediate vicinity of the deformation were assigned random importance weights between 0.95 and 0.99. The remainder of the grid was assigned random weights between 0.9 and 0.8. For each training session, the entries of $\mathbf{Q}$ were changed using the rules above, based on location of displacement in the mesh, in order to correspond to the heuristic model. Thus, the areas of the mesh with the largest resulting displacements are weighted to most accurately resemble the truth model.

Moreover, constraints were imposed regarding how much the original spring weights could be perturbed by the SPSA algorithm. As such, the perturbation magnitudes were limited to a value no larger than the spring constant picked by the heuristic described in Section 2.3, 0.75N/m. This was done to ensure that the spring constants picked to approximate $\mathbf{f}_{opt}$ would not vary too much from their original values and lead to noticeable discontinuities in displacement as the user force changed during the haptic rendering. The gains of the perturbation magnitudes were also designed to decrease as a function of the iteration number of the algorithm, but were limited to a lower bound of 0.001 to ensure the algorithm did not remain static or find an unstable solution with negative spring constants. The reason this measure was undertaken was to ensure that the noisy search algorithm was given a lot of initial freedom to look for a minimum, limited to 0.75 in the first few iterations of the algorithm, but would then converge to the solution path chosen.

The maximum and minimum values of the springs were also limited by the training algorithm. Any negative spring constant was reverted to its original value and the training process was restarted once more. In only three of the eighteen cases did the algorithm have to start over. Therefore, the physical modes of vibration we were solving for were in almost all cases naturally close to the initial conditions of the $\mathbf{K}$ matrix, and the solution did not have to be tampered with to get an artificially optimized system. Figure 4 depicts the reduction in error between $\mathbf{f}_{\text{act}}$ and $\mathbf{f}_{\text{opt}}$ for locally trained displacements, with the aforementioned constraints active on the system. Figure 3 shows the resolved forces resulting from the displacements depicted in Figure 2 for the three models. The so-

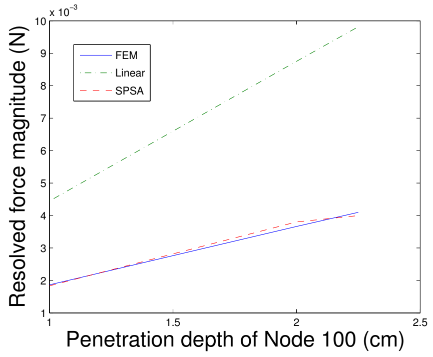

Figure 3: Displacement mesh node loaded in Figure 2 vs resolved force for the FE model, the heuristic from Section 2.3, and the mesh optimized by SPSA.

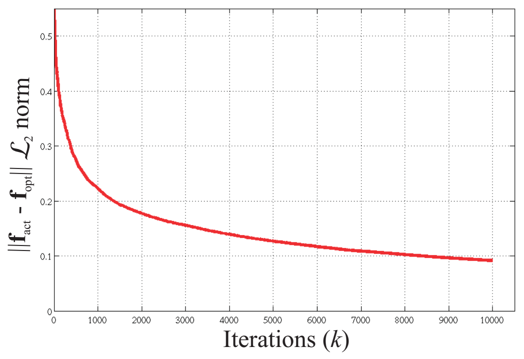

Figure 4: The learning curve of the SPSA algorithm; the $\mathscr{L}_2$-norm of the difference of f_{act} and f_{opt} shows a sharp decrease in error in the first $5{,}000$ iterations, and subsequently gravitates below 0.1 for higher iteration values. Error minimization is crucial in these first couple of thousand iterations in order to maintain a fast-response learning algorithm.

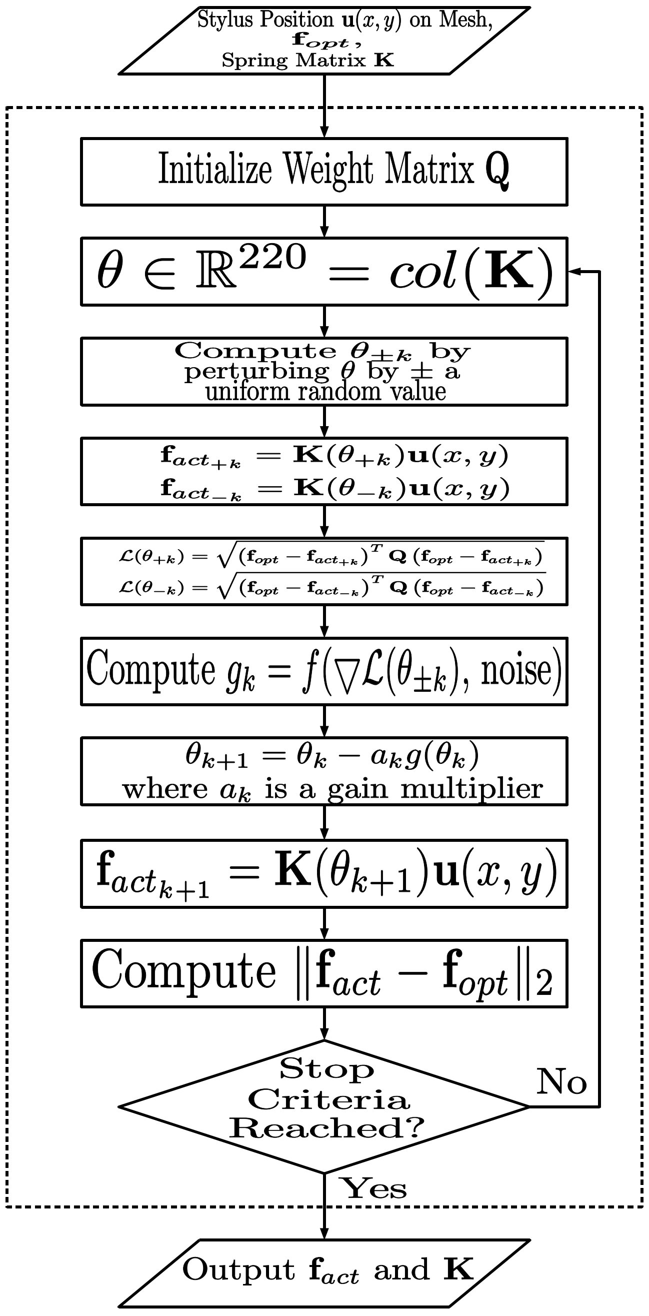
Stylus Position $\mathbf{u}(x,y)$ on Mesh, $\mathbf{f}_{opt}$, Spring Matrix $\mathbf{K}$
Initialize Weight Matrix $\mathbf{Q}$
$\theta \in \mathbb{R}^{220} = col(\mathbf{K})$
Compute $\theta_{\pm k}$ by perturbing θ by $\pm$ a uniform random value
$\mathbf{f}_{act_{+k}} = \mathbf{K}(\theta_{+k})\mathbf{u}(x,y)$
$\mathbf{f}_{act_{-k}} = \mathbf{K}(\theta_{-k})\mathbf{u}(x,y)$
$\mathcal{L}(\theta_{+k}) = \sqrt{(\mathbf{f}_{opt} - \mathbf{f}_{act_{+k}})^T \mathbf{Q} (\mathbf{f}_{opt} - \mathbf{f}_{act_{+k}})}$
$\mathcal{L}(\theta_{-k}) = \sqrt{(\mathbf{f}_{opt} - \mathbf{f}_{act_{-k}})^T \mathbf{Q} (\mathbf{f}_{opt} - \mathbf{f}_{act_{-k}})}$
Compute $g_k = f(\nabla\mathcal{L}(\theta_{\pm k}), \text{noise})$
$\theta_{k+1} = \theta_k - a_k g(\theta_k)$ where a_k is a gain multiplier
$\mathbf{f}_{act_{k+1}} = \mathbf{K}(\theta_{k+1})\mathbf{u}(x,y)$
Compute $||\mathbf{f}_{act} - \mathbf{f}_{opt}||_2$
Stop Criteria Reached?
No
Yes
Output $\mathbf{f}_{act}$ and $\mathbf{K}$

Figure 5: Flowchart describing the SPSA algorithm used to compute the spring weights contained in $\mathbf{K}$, and the actual force fed back to the haptic stylus. The dashed box represents the set of processes performed by the SPSA algorithm. The "Stop Criteria" can be any set of constraints that indicate termination of execution, such as number of iterations, computation time, or magnitude of error, as represented by the L2 norm shown in Figure. 4

lution of the SPSA algorithm is physically feasible, i.e. the spring constants, especially around the neighborhood of the displacement, differ little from the heuristic described in Section 2.3 and no unrealistically high spring weights nor algorithm-altered spring weights are encountered using the optimized spring model.

In order for the model to deform in a continuous fashion, the spring constants must also change from training session to training session in an incrementally increasing/decreasing fashion and do not over-deviate from the values they are assigned at other training sessions whose displacements are similar in magnitude or direction. For example, having an optimized spring constant change from 0.7 to 3.0 to 0.05 N/m for displacements of 1 cm, 2 cm, and 3 cm, respectively, in homogeneous tissue does not make physical sense, because if the displacement increases monotonically, so should the force. This is all the more crucial for spring constants in the neighborhood of the applied force (i.e. the contact point), because as the user increases the displacement applied, the case described above would yield an unnatural discontinuity in force, and the simulation would thus cease to be realistic.

In order to eliminate this problem, the optimization algorithm was constrained especially for the weights in the neighborhood of the applied displacement by: (1) using a penalty based weight matrix, $\mathbf{Q}$, to apply a high gain to weights in the neighborhood of the applied force, thereby making the RMS more heavily influenced by those optimized values, and, (2) limiting the amount of freedom the model-free SPSA algorithm had in choosing a valid optimal solution for the spring weights. The latter constraint, for the magnitude of the random perturbation that could be applied to a spring constant during optimization, ensured that the algorithm did not deviate too much from the overall linear solution of the system, thereby en-

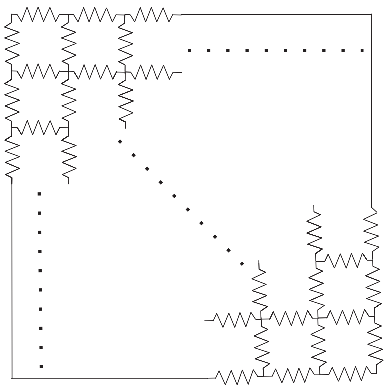

Figure 6: Mass-spring system used where the spring constants were derived from the learning algorithm.

suring a smooth albeit discrete change between the different trained spring matrices corresponding to the different applied forces to nonlinear tissue. The overall effect of these constraints was to ensure that in the neighborhood of the force application, the spring constants would behave in a physically relevant way that would create a continuous, high fidelity haptic rendering; i.e. displacements of 1 cm, 2 cm, and 3 cm would lead to nonlinearly increasing spring constant weights of 0.05, 0.7, and 3.0, respectively. Other potential constraints on the algorithm were omitted, to allow for a reasonable amount of flexibility during training.

4 Real-time Haptic Rendering

Our simulation was written in C++ using the Boost libraries [4] for fast vector and sparse matrix operations. Our test machine ran Fedora Core 5 Linux and had a dual core 3.0 GHz Pentium 4 processor and 1 GB of RAM and was able to simultaneously update two meshes of 121 nodes each at greater than 1 kHz. The display device was a PHANTOM Omni [18], although any convenient 2-DOF or higher impedance-type haptic device could be used. Interaction with the Omni was through SensAble's OpenHaptics Toolkit [17].

A 2-D mass-spring-damper mesh, as shown in Figure 6, was defined to correspond to the FE model used for the "truth model" by using the same topology of nodes. These nodes were then connected to all adjacent nodes in a 4-way connected neighborhood, spaced 1 cm apart. A node could therefore have a maximum of 4 springs, two connected to its initially horizontal neighbors and two connected to its initially vertical neighbors. Spring constants were then initialized according to the deformable tissue modeling heuristic of Section 2.3.

4.1 Deformation

Internally, node positions were stored as vectors (of size 2*N*), and node masses $\mathbf{M}$, and spring constants, $\mathbf{K}_l$, as sparse matrices of size (of size 2*N*-x-2*N* and 2*M*-by-2*M* respectively), where *N* is the number of nodes in the mesh, *M* the number of springs connecting them, and l indexes into the set of 18 locally-optimized sets of spring constants available. A damping matrix, $\mathbf{B}$, with very small values, was also added to maintain stability. Updates to node positions were calculated as in [9] according to

$$\Delta\mathbf{x}_t = \mathbf{x}_t - \mathbf{x}_0, \tag{11}$$

$$\mathbf{M}\ddot{\mathbf{x}}_t = \mathbf{K}_l\mathbf{\Delta x}_t + \mathbf{B}\dot{\mathbf{x}}_t + \mathbf{f}_t, \tag{12}$$

where $\mathbf{x}$ is the location of mesh nodes at time t, $\Delta\mathbf{x}$ is the current displacement of the nodes from their rest position, $\mathbf{f}$ is the vector of exogenous forces being applied to all nodes in the mesh, and $\ddot{\mathbf{x}}$ and $\dot{\mathbf{x}}$ are the resulting node accelerations and velocities respectively. For the point contact we were simulating, only two entries of $\mathbf{f}$ would ever be non-zero. The system was solved by inverting the diagonal matrix $\mathbf{M}$ and numerically integrating according to

$$\ddot{\mathbf{x}}_t = \mathbf{M}^{-1}\left(\mathbf{K}_l\mathbf{\Delta x}_t + \mathbf{B}\dot{\mathbf{x}}_t + \mathbf{f}_t\right) \tag{13}$$

$$\dot{\mathbf{x}}_t = \dot{\mathbf{x}}_{t-1} + 0.5\left(\ddot{\mathbf{x}}_{t-1} + \ddot{\mathbf{x}}_t\right) \tag{14}$$

$$\mathbf{x}_t = \mathbf{x}_{t-1} + 0.5\left(\dot{\mathbf{x}}_{t-1} + \dot{\mathbf{x}}_t\right). \tag{15}$$

The selection of l is described in Section 4.3.

4.2 Collision Detection

Our collision detection algorithm made use of a master-proxy model [25]. For the purpose of detecting collisions, only the nodes making up the outer boundary of the mesh were considered. The haptic device was modeled as a point in space located at the center of rotation of the Omni's gimbal, henceforth referred to as the master position $\mathbf{x}_m$. The proxy position $\mathbf{x}_p$ was the same as the master position except when contact was made with the mesh, at which point the proxy remained on the surface of the mesh at the point of contact. A penetration force was then generated through a virtual coupling, with constant of proportionality k_{C}, between the master and the mesh boundary, as $f = k_{\mathrm{C}}\left(\mathbf{x}_p - \mathbf{x}_m\right)$, creating the haptic interaction force.

Collision detection consisted of detecting when the master penetrated the mesh and maintaining the proxy position on the boundary as the mesh moved. The proxy must be prevented from penetrating the mesh at each time step between updates of the position of the mesh, even when the mesh boundary moves to contain $\mathbf{x}_p$. Interactions were therefore broken into several cases, and the update at a time step depended on whether $\mathbf{x}_m$ and $\mathbf{x}_p$ lay on the interior of the mesh and whether there was contact with the mesh at the previous time step.

The nominal interaction case occurred when $\mathbf{x}_p$ was on the exterior of the mesh and there was no contact at the previous time step. Then a segment-segment check was performed in which a line segment s_{mp} was constructed from $\mathbf{x}_p$ to $\mathbf{x}_m$ and each segment of the boundary, s_i was checked for intersection with this segment. For each segment s_i intersected by s_{mp}, let the endpoints of this segment be called $s_i(0)$ and $s_i(1)$. Then the intersection point int_i is given by

$$\mathrm{int}_i = s_i(0)w(0) + s_i(1)w(1) \tag{16}$$

$$1 = w(0) + w(1) \tag{17}$$

$\mathbf{x}_p$ was set to $\mathrm{argmax}_i||\mathbf{x}_m - \mathrm{int}_i||$, the intersection point of largest penetration distance if one existed, otherwise to $\mathbf{x}_m$. The interaction force $f_{\mathrm{I}}(j)$, $j = 0,1$ was applied to the mesh at the endpoints of the selected segment $s_i(j)$ as $f_{\mathrm{I}}(j) = -w(1-j)f$, and w and i were stored for future reference.

If instead there was contact on the previous time step, but again $\mathbf{x}_m$ lay inside and $\mathbf{x}_p$ outside the mesh, then $\mathbf{x}_p$ was simply updated to lie in the same position with respect to the surface of the mesh as in the last time step: $\mathbf{x}_p = s_i(0)w(0) + s_i(1)w(1)$. The result is an infinite friction model which allows no slip along the surface of the mesh, although this could be modified for whatever surface properties are desired.

In the case where the proxy was inside the mesh, the master was on the interior and there was contact during the last time step, then the mesh moved to contain the proxy while the user was pushing into the mesh. It is desirable to maintain as consistent a contact as possible, so the proxy was maintained at the same position on the surface as before. The remaining cases represented situations where the mesh has moved to envelop the proxy while the master remained still or actually moved away from the mesh. In these cases, the proxy was projected onto the nearest point of each segment of the

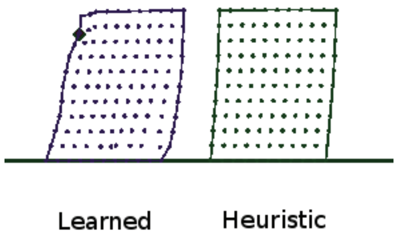

Figure 7: Response of our learned elastic model versus that of a heuristic (linear) elastic model. Using the PHANTOM Omni stylus, the user applies a load to either of the models and the corresponding deformation responses of both systems are shown.

boundary $n(s_i)$, set to the closest one, $\mathbf{x}_p = \text{argmin}_i ||\mathbf{x}_p - n(s_i)||$, and w and i updated accordingly.

4.3 Dynamic Selection of Mesh Constants

The variable l in (12) and (13) is selected according to the magnitude and direction of f to update the mesh according to the set of spring constants trained from the most similar input. In our case, the mesh was trained on forces applied to a few key points on the mesh. At each point, forces were applied at one or more angles at two magnitudes. The selection of the most similar input therefore consisted of finding the nearest application point on which the mesh was trained, followed by the angle at that application point closest to that currently observed, followed finally by the nearest magnitude force at that application point and angle.

It is possible when using this method to switch between sets of spring weights with very different values, especially when modeling a highly non-linear material with a small number of meshes. If the mesh is displaced from its rest position, any change to the spring stiffnesses is likely to result in a change in the potential energy of the system, and therefore to both its dynamics and to the force rendered to the user. These changes, of course, allow a greater degree of non-linearity to be modeled, but they may also cause discontinuities in the response at the time of switching between meshes. Although this effect was not observed in our implementation, we developed a method to mitigate its effects. Rather than setting the mesh values at time t to K_l as in (13), one may maintain a a weighted moving average of spring constants, which we'll denote $\mathbf{K}^t$ at time t. Then (13)becomes

$$\ddot{\mathbf{x}}_t = \mathbf{M}^{-1}\left(\mathbf{K}^t\Delta\mathbf{x}_t + \mathbf{B}\dot{\mathbf{x}}_t + \mathbf{f}_t\right) \tag{18}$$

and we update $\mathbf{K}^t$ according to

$$\mathbf{K}^t = \gamma\mathbf{K}_l + (1-\gamma)\mathbf{K}^{t-1} \tag{19}$$

where the innovation factor, γ, determines the rate at which the spring stiffnesses change to those of the new mesh. This method generalizes well to any number of meshes and avoids the overhead of maintaining a history of meshes selected at each time step.

4.4 Display

The mesh generation algorithm described above gave smooth and continuous deformation results at nominal (> 1 kHz) haptic rendering rates.

Figure 7 shows a graphical display of the simulation. In this demonstration, two mass-spring-damper meshes with identical topologies are displayed, labeled “learned” and “heuristic”. The heuristic mesh has spring constants set by the heuristic in Section 2.3, while our mesh had various sets of spring constants set by our learning algorithm and selected in real time according to the interaction forces. Haptic interaction was then allowed with either of the two meshes, with the haptic cursor displayed graphically as a diamond, so that pushing on either mesh would exert an equivalent force on the nodes of the other mesh as well. This coupling allowed simultaneous display of the different reactions of each mesh for comparison. The 3D workspace of the Omni was mapped to the 2D domain by simply ignoring motion in and out of the image plane.

5 Discussion

In this paper, we presented a new method for the optimization and haptic rendering of tissue deformation. We have shown that a model-free optimization approach to determine spring constants yields satisfactory results; our optimized mesh is able to match the force derived from FE analysis about 5 times better than the heuristic described in Section 2.3, where we have considered the experimentally derived FE data as the standard in accuracy for nonlinear brain tissue deformation behavior. The most crucial aspect of this work is the physical accuracy of the optimized mass-spring-damper system.

The future work for this research can be divided into four main categories. First, additional features can be added to the SPSA learning method. The learning algorithm can be extended to train a 3D mesh, and also a formula could describe the optimal learning gains and entries of the priority weight matrix, $\mathbf{Q}$, as a function of the displacement applied and mesh size being trained. Moreover, the SPSA learning method could be implemented with real-time parameter changes. Since the FEM data obtained from ABAQUS are generated offline, we allowed the run time of our algorithm to be fairly high, in order to focus on implementing accurate spring constant solutions, $\boldsymbol{\theta}$, for the respective training deformation data sets. However, inspecting Figure 4, it can be noted that the bulk of the optimization happens in the first 1000 iterations of the optimization algorithm, which can be run in fractions of a second on a dedicated computer. One can investigate how well the model-free training algorithm performs in truly online adaptive settings and once again develop a heuristic for SPSA gain parameters that trades off between accuracy and speed of execution. Second, real force data obtained from tissue deformation via a series of indentation and/or shear experiments could be used for training data, without the FE model as an intermediary, thereby providing the SPSA algorithm with the most accurate brain tissue deformation data. These experiments would also set up the framework for the inclusion of other tissue types into the simulation (e.g. liver, muscle, etc.). Third, the system could learn meshes with springs consisting of nonlinear functions of displacement and location in the mesh as well as the mass matrix. Finally, the system could be extended to 3D rendering. We began with a 2D implementation to demonstrate the validity of the approach, but most real applications would require a 3D virtual environment. The main difficulty involved in this extension is the exponential increase in the size of the parameter space and the resulting computational complexity. However, parallelization of the required matrix operations could result in large computational gains.

References

[1] ABAQUS Inc. Rising Sun Mills, 166 Valley Street, Providence, RI 02909 USA.

[2] G. Bianchi, M. Harders, and G. Székely. Mesh topology identification for mass-spring models. In R. E. Ellis and T. M. Peters, editors, *MICCAI (1)*, volume 2878 of *Lecture Notes in Computer Science*, pages 50–58. Springer, 2003.

[3] G. Bianchi, B. Solenthaler, G. Székely, and M. Harders. Simultaneous topology and stiffness identification for mass-spring models based on fem reference deformations. In C. Barillot, D. R. Haynor, and P. Hel-

lier, editors, *MICCAI (2)*, volume 3217 of *Lecture Notes in Computer Science*, pages 293–301. Springer, 2004.

[4] Boost.org. *Boost C++ Library Documentation*, December 2005. Available via http://www.boost.org/.

[5] S. Cotin, H. Delingette, and N. Ayache. A hybrid elastic model for real-time cutting, deformations, and force feedback for surgery training and simulation. *The Visual Computer*, 16(7):437–452, 2000.

[6] O. Deussen, L. Kobbelt, and P. Tucke. Using simulated annealing to obtain good nodal approximations of deformable objects. In D. Terzopoulos and D. Thalmann, editors, *Computer Animation and Simulation '95*, pages 30–43. Springer-Verlag, 1995.

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

[8] Y. C. Fung. *Biomechanics: mechanical properties of living tissues*. Springer-Verlag, second edition, 1993.

[9] S. F. Gibson and B. Mirtich. A survey of deformable modeling in computer graphics. Technical report, MERL - A Mitsubishi Electric Research Laboratory, 1997.

[10] A. E. Kerdok, S. M. Cotin, M. P. Ottensmeyer, A. M. Galea, R. D. Howe, and S. L. Dawson. Truth cube: Establishing physical standard for soft tissue simulation. *Medical Image Analysis*, 7:283–291, 2003.

[11] K. Miller, K. Chinzei, G. Orssengo, and P. Bednarz. Mechanical properties of brain tissue in-vivo: experiment and computer simulation. *Journal of Biomechanics*, 33(11):1369 – 1376, 2000.

[12] S. Misra, A. M. Okamura, and K. T. Ramesh. Force feedback is noticeably different for linear versus nonlinear elastic tissue models. In *Second Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (World Haptics)*, pages 519–524, March 2007.

[13] W. Mollemans, F. Schutyser, J. Cleynenbreugel, and P. Suetens. Tetrahedral mass spring model for fast soft tissue deformation. In *International Symposium on Surgery Simulation and Soft Tissue Modeling*, pages 145–154, 2003.

[14] D. Morris. *Haptics and Physical Simulation for Virtual Bone Surgery*. PhD thesis, Department of Computer Science, Stanford University, 2006.

[15] R. W. Ogden. *Non-linear elastic deformations*. Ellis Horwood Ltd., Chichester, UK, first edition, 1984.

[16] C. Paloc, F. Bello, R. I. Kitney, and A. Darzi. Online multiresolution volumetric mass spring model for real time soft tissue deformation. In *Proceedings of the 5th International Conference on Medical Image Computing and Computer-Assisted Intervention-Part II*, pages 219–226, 2002.

[17] SensAble Technologies. *OpenHaptics Toolkit version 2.0 Programmer's Guide*, 2005.

[18] SensAble Technologies. *Specifications for the PHANTOM Omni(TM) haptic device*, 2005.

[19] J. C. Spall. An overview of the simultaneous perturbation method for efficient optimization. Technical Report 1, Johns Hopkins APL, 1998.

[20] J. C. Spall. *Introduction to stochastic search and optimization: estimation, simulation, and control*. Wiley, Hoboken, USA, first edition, 2003.

[21] G. Székely, C. Brechbühler, J. Dual, R. Enzler, J. Hug, R. Hutter, N. Ironmonger, M. Kauer, V. Meier, P. Niederer, A. Rhomberg, P. Schmid, G. Schweitzer, M. Thaler, V. Vuskovic, G. Tröster, U. Haller, and M. Bajka. Virtual reality-based simulation of endoscopic surgery. *Presence: Teleoperators & Virtual Environments*, 9(3):310–333, 2000.

[22] A. Van Gelder. Approximate simulation of elastic membranes by triangulated spring meshes. *Journal of Graphics Tools*, 3(2):21–41, 1998.

[23] X. Wu, M. S. Downes, T. Goktekin, and F. Tendick. Adaptive nonlinear finite elements for deformable body simulation using dynamic progressive meshes. In A. Chalmers and T.-M. Rhyne, editors, *Computer Graphics Forum*, volume 20(3), pages 349–358. Blackwell Publishing, 2001.

[24] H. Zhong, M. Wachowiak, and T. Peters. A real time finite element based tissue simulation method incorporating nonlinear elastic behavior. *Computer Methods in Biomechanics and Biomedical Engineering*, 8(3):177–189, 2005.

[25] C. Zilles and J. Salisbury. A constraint based god-object method for haptic display. In *Proceedings of the IEE/RSJ International Conference on Intelligent Robots and Systems, Human Robot Interaction, and Cooperative Robots*, volume 3, pages 146–151, 1995.