Real-Time Finite Element Modeling for Surgery Simulation: An Application to Virtual Suturing

Jeffrey Berkley[1,4], George Turkiyyah[2], Daniel Berg[3], Mark Ganter[4], Suzanne Weghorst[1]
U*niversity of Washington, Seattle, WA*

*1. Human Interface Technology Lab, 2. Department of Civil Engine*ering*, 3. Division of Dermatology, 4. Department of Mechanical Engineering*

Real-time finite element (FE) analysis can be used to represent complex deformable geometries in virtual environments. The need for accurate surgical simulation has spurred the development of many of the new real-time FE methodologies that enable haptic support and real-time deformation. These techniques are computationally intensive and it has proved a challenge to achieve the high modeling resolutions required to accurately represent complex anatomies. The authors present a new real-time methodology based on linear FE analysis that is appropriate for a wide range of surgical simulation applications. A methodology based is proposed that is characterized by high model resolution, low preprocessing time, unrestricted multi-point surface contact and adjustable boundary conditions. These features make the method ideal for modeling suturing, which is an element common to almost every surgical procedure. This paper describes *constraints* in the context of a *Suturing Simulator* currently being developed by the authors.

I. INTRODUCTION

Accurate real-time rendering of deformable objects has proved a difficult challenge. Continuum mechanic based methodologies that account for complex geometries and material properties typically require significant computation. As a result, it is difficult to utilize these methods and achieve the 30 Hz update rates desired for graphical rendering. In virtual reality applications that employ haptic feedback (the sense of touch), the demand is even more taxing, sometimes exceeding visual demands by more than an order of magnitude.

Finite element (FE) modeling is an accurate continuum mechanics based methodology that has served as an industry standard for prototype testing and design. Bridges, cars, ships, airplanes, prosthetic devices and mechanical parts represent only a small sample of products that have depended on the accuracy of FE modeling for development. While conventional FE formulations are not applicable to real-time rendering for graphics or haptics, FE modeling

methodologies that utilize novel preprocessing techniques and alternative real-time solving methodologies are starting to emerge.

Most of the advances in real-time FE modeling have occurred as a result of the demand for realistic surgery simulation. For many medical procedures, there are no efficient means for training a medical student to perform a surgery and practice on real patients is often the only option. It is generally expected that simulation training will one day be as important to medicine as it is now to aviation. However, one of the reasons the medical community is currently reluctant to accept many of the commercial simulators available is that they do not provide sufficient realism. As a means of achieving more accurate deformation and haptic interaction, a number of real-time FE based approaches have been offered in context with surgery simulation.

II. SUTURING IN SURGERY SIMULATION

Suturing is a task fundamental to almost every surgical procedure. For the year of 1999, the number of skin surgery procedures covered by Medicare that required suturing totaled approximately 2.5 million (number based on excisions, repairs, adjacent tissue transfers, skin graphs and island pedicle graphs) [1]. While this number is substantial, it represents only a small portion of the total number of procedures across all disciplines that required suturing in 1999. Because of the broad application, suturing was considered by the authors to be an ideal application for real-time FE modeling.

Surgery on the skin ranges from simple suturing of lacerations to complex tissue movements such as flaps. Training in cutaneous surgery uses the traditional surgical apprenticeship model aided by tools such as suturing boards, pig's foot training courses and/or the use of live animals. For a variety of reasons, these methods are not ideal. At the University of Washington the authors have been developing a suturing simulator based on finite element

modeling methods that allow for real-time haptic interaction and soft tissue deformation [2-4]. The requirements of this suturing simulator have directly influenced the development of our real-time FE methodologies.

Table 1 and figure 1 describe how the authors break down the task of suturing. Given these steps, certain demands are placed on real-time modeling algorithms. The following is a list of the capabilities required to achieve an effective suturing simulator:

Accurate Deformation and Force-Feedback: Every step of table 1 will result in deformation of the tissue. Reaction forces must also be determined for haptic feedback. To accurately model various complex tissue structures, the FE model must be high in resolution (thousands of nodes and elements). This is required for visually appealing graphical rendering of complex geometries. Even at high resolution, the FE model must allow for rapid determination of reaction forces to meet the demands of haptic rendering.

Contact at Any Point on the Model Surface: One way to evaluate a student's suturing skills is by measuring the distance the needle is inserted from the wound and how far sutures are placed from each other. This means that contact must be allowed anywhere on the surface of the model and cannot be limited to the locations of the surface nodes. Otherwise, there will be inherent measurement error when determining suture placement.

Adjustable Boundary Conditions: Sutures can be modeled as stiff springs that only support tension. Springs could be added and removed from the FE stiffness matrix or they could be modeled through the applied force vector. If a large stiffness matrix is preprocessed to assume a format more efficient than the original stiffness matrix, it becomes computationally expensive to make substantial changes to the mesh. For this

reason, the ability to model sutures through changing applied forces is an attractive alternative.

Multipoint Contact: When applying a suture, the needle enters at one point and exits at another. This means that at least two points of displacement must be applied to model needle insertion. Suturing also involves the use of two tools: a needle driver and tweezers or a skin hook (see figure 2). This adds an additional point or points of contact where displacement must be defined. Not only must displacements be prescribed at multiple points, but the appropriate reaction forces must also be determined to provide force-feedback for each tool.

Rapid Preprocessing: Preprocessing can be used to compress FE equations so that high-resolution models can be solved in real-time. Unfortunately, if changes are made to the resting configuration of the FE mesh, all preprocessing must be repeated. While there are many surgical procedures that could be simulated without changing the original mesh, the majority of procedures do require tasks such as cutting. As another example, one can envision a teacher creating a homework set of skin defects that must be repaired. The ability to drag and drop skin defects onto various anatomical models would require rapid updates to the original FE meshes. Preprocessing must therefore be accomplished quickly enough so that down time for mesh alteration does not become a major distraction.

To date, no methodology has been reported in the literature that accommodates all of the objectives defined above. After describing some of the more recent applications of FE modeling to surgery simulation, the authors present a constraint-based methodology, which is appropriate

for suturing simulation. Constraints can also be used to enhance existing methodologies by increasing model resolution and by reducing pre-processing time.

III. APPLICATIONS OF FE MODELING TO SURGERY SIMULATION

Accurate deformation and haptic feedback has been achieved through real-time FE modeling. However, surgery simulation has demands that are different from typical engineering and animation applications. Biological structures have complex anatomies that require significant model resolution for visual appeal and modeling accuracy. Biological materials are also difficult to obtain and model. Besides exhibiting properties such as non-linearity, anisotropy, creep, stress relaxation and visco-elasticity, material properties can vary substantially between subject and even within a small local tissue region on a single subject. On the other hand, FE modeling for surgery simulation may require only selected variables to be computed. In most cases, it is not necessary to calculate displacement at interior model points that are not being rendered. Reaction forces are only required where the user touches the model. Stress and strain calculation may not be essential. The principle of "calculate only what you need" has served as the basis for many real-time FE approaches.

The term "FE modeling" can refer to a wide variety of mathematical representations. Different materials lend themselves to different approaches and real-time analysis has only been achieved within a small fraction of problem domains. It is safe to say that the literature has only just begun to address the many obstacles that must be overcome to achieve truly accurate real-time tissue modeling. Advances must take place in steps, and before addressing some of the more complex issues it would be valuable to achieve adequately high model resolution for simple linear models. After thoroughly solving the problem of real-time linear modeling, we

will be better prepared to address more complex issues such as non-linear property modeling and real-time mesh alteration (such as for cutting).

In FE analysis, the governing matrix equation is

$$Md'' + Dd' + Kd = f \qquad \text{(Eq. 1)}$$

where M is the mass matrix, D is the damping matrix, K is the stiffness matrix, d is the nodal displacement and f is the applied force. The size of each matrix in equation 1 is n x n, where n is three times the total number of nodes in the case of 3D modeling. Each of these matrices is assembled from the elements of an FE mesh (see figure 3). Since the elements are of simple shapes, such as tetrahedrons or hexahedrons, continuum mechanics can be used to define their behavior. The assembly of many simple shaped elements can approximate the behavior of a complex geometry. The process is somewhat analogous to building a model from lagos, where various shapes can be achieved by piecing together lago blocks. The accuracy of an FE model generally increases with the number of elements used to approximate its geometry. A more in depth discussion of the formulation of FE equations can be found in *The Finite Element Method* by Zienkiewicz and Taylor.

Before real-time FE approaches existed, some applications simply used FE analysis for predictive value, and did not require real-time computation. Both Larabee and Pieper have used the FE method to predict the results of plastic surgery [5, 6]. Animations of muscle contraction and skin deformation during grasping have been based on the FE method [7, 8]. Waters and Terzoplos have used FE analysis to computer-synthesize animation of facial expressions [9].

Other investigators have used FE analysis to optimize simulators that utilize alternative methods, such as spring models, for providing force-feedback and deformation [10]. Cotin et al

developed a tensor-based methodology that relied on FE modeling for optimization [11, 12]. After creating a tetrahedral based FE model of a liver with commercial software, tensors to control nodal deformation and force-feedback were optimized by exploiting the linear nature of elastic FE deformation results. Each surface node was associated with multiple tensors to account for every possibility of node contact. This methodology does provide a means for approximating FE results for high-resolution models, however a lengthy preprocessing period is required. More recently, anisotropic material properties have been incorporated into a liver model [13].

Bro-Nielsen et al was one of the first to apply the "calculate only what you need" principle to real-time FE based deformation [14]. They applied a process known as condensation (described in depth later) to reduce the problem from a size of n x n to the size of v x v (where v is three times the number of visible nodes for 3D). After condensation, the reduced stiffness matrix was inverted to permit the calculation of visible nodal displacements. This reduced format gives the exact same displacement results that can be obtained with conventional FE analysis. Using a Silicon Graphics ONYX with four MIPS R4400 processors, a model of the lower leg with 250 visible nodes was deformed with a visual update rate of 20 Hz. In this demonstration, a simulated force was applied to only one node, and force-feedback was not supported. Hansen used a similar approach to the modeling of brain tissue, however Hansen's simulator did incorporate force-feedback [15].

James and Pai have achieved real-time interaction through boundary element models (BEMs) [16]. Given the geometry of a model, homogeneous material properties, and a set of boundary conditions, acceptable graphical update rates are achievable by pre-computing the discrete Green's functions of a referenced boundary value problem. A 3D model with n degrees of nodal freedom and s applied boundary conditions can be solved for one point of contact at a rate of $18ns$ flops. Changes in the boundary conditions cost an additional $O(s^3)$ operations and $O(sn)$ operations after

the initial change. A force interpolation scheme was used to approximate forces in-between time steps that allowed a higher haptic rate than visual rate to be supported. While this methodology was not applied to surgical simulation it does indicate an alternative to FE analysis. Unfortunately, boundary element analysis does not accommodate inhomogeneous material properties, so its application to surgery simulation may be limited.

An increasing number of applications are making use of dynamic FE modeling. By using a lumped mass representation, with a similarly derived damping matrix, explicit integration can be used to achieve acceptable graphical rates with run time computation costs scaling linearly with the number of nodes.

The virtual environment for eye-surgery proposed be Sagar et al was one of the earlier applications based on dynamic analysis [17]. The Cornea was modeled with a nonlinear elastic material (Mooney-Rivilin material) and cutting was allowed. Parallel processing was used to separate graphical rendering from mathematical computation in order to increase performance speed. The model contained only a small number of elements, but NURBS (non-uniform rational B-splines) were used to interpolate between the nodes. The simulation did not provide real-time computation in that deformation was only calculated once per second; however, stress distribution color overlays on the cornea provided a means of visual feedback previously unknown to surgery simulators.

Computation can also be reduced in dynamic analysis if only the most significant vibration modes are used to calculate deformation [18]. Because haptics requires a significantly higher update rate than graphics, Zhuang achieved haptic interaction through a force interpolation scheme. Another approach is to apply Spectral Lanczos Decomposition (based on Laplace transforms) rather than standard numerical integration [19]. The reduction in computation costs allows more nodes to be used than possible by direct application of dynamic analysis.

Dynamic modeling coupled with non-linear material properties is starting to receive some recent attention [18, 20, 21]. Wu et al have applied the Mooney-Rivlin model to dynamic analysis with an emphasis on adaptive meshing. This allows greater model resolution at the region of contact with courser representation at distant regions. In a preprocessing stage, a high-resolution mesh is converted to a low-resolution mesh by combining adjacent elements throughout the model. The resulting elemental changes are stored so that elements can be restored at run-time. This allows rapid run-time refinement of the course mesh based on where contact takes place. Using an 800 MHz Pentium III PC, a model with 2,200 vertices and 4,500 membrane elements has been solved at rate of 20 frames per second. Haptic interaction has not yet been added, however, the authors anticipate adding haptics by using a multi-rate scheme to interpolate between time steps [22].

IV. METHODS

The authors have developed a methodology that applies constraints to linear elastic models. The methodology emphasizes high model resolution, multipoint contact, rapid preprocessing and accommodates dynamically changing boundary conditions. Although this method could easily be adapted to dynamic analysis without requiring a lumped mass matrix, the authors feel that the inclusion of dynamic effects is unnecessary for simulating suturing. Suturing requires slow precise concentrated movements, so dynamic contributions are generally negligible.

In order to illustrate the advantages of constraints over other real-time FE methodologies, it is necessary to describe the drawbacks of typical methods. Only the application of static linear analysis in three dimensions will be discussed in the following subsections.

A. LU Decomposition

The linear FE equation described by equation 1 can be reduced if dynamic effects are ignored.

$$Kd = f \qquad \text{(Eq. 2)}$$

In equation 2, K is an nxn stiffness matrix and d and f are nx1 vectors representing nodal displacements and applied force respectively. If the above equation were to be used to simulate suturing, the vectors d and f would typically be determined after the user makes contact with an FE model that represents a soft tissue structure. Usually, displacement will be applied at a set of contact nodes and the reaction forces must be determined to support force feedback at a sufficiently high update rate. The displacements that result at the nodes not in contact must be determined at graphical rates (approximately 30 Hz). The unknown variables are therefore the reaction force at the contact nodes and the displacement of non-contact nodes.

Most force-feedback devices (such as the Phantom) will allow supporting software to define a directional force that is derived from a given tool position. Assume that a relationship can be established between the movement of the tool tip and the displacement of model nodes in contact with the tool. There may also be known applied static forces, such as to represent the pretension of skin, and dynamically changing forces, such as to model previously applied sutures.

The typical approach to solving a set of linear equations, such as those of equation 2, would be LU decomposition [23]. For a 100% dense matrix (no zero values), $1/3n^3$ calculations are required to break an nxn matrix down into lower (L) and upper (U) triangular matrices. Throughout this paper, the word "calculations" will refer only to multiplications and divisions since the computation time required for additions and subtractions is minimal in comparison.

For the purpose of VR simulation, the LU components of K could be determined during a preprocessing stage in order not to affect run-time performance. Forward and backward substitution using the L and U matrices would require between $1/2n^2$ and n^2 calculations, depending on the location of the first nonzero value in the f vector.

The K matrix is usually very sparse, with the sparsity depending on the number of nodes, the geometry of the mesh, the type of elements used, etc. A model with 1,000 nodes may only have nonzero values in about 1% of the entire stiffness matrix (sparsity=0.01). A sparse K matrix will result in sparse LU components, which lend themselves to efficient sparse matrix solving algorithms. The value of using sparse LU solving algorithms becomes more profound as models get larger. When using sparse matrix techniques, a 100 node model might benefit from only a 20% decrease in total calculations, while a 1,000 node model might yield an 80% efficiency increase. (these estimates were established by using the author's sparse LU decomposition C++ code with arbitrary FE meshes). The effectiveness of sparse LU solving algorithms varies for different FE mesh structures and is largely dependent on the sparse makeup of the stiffness matrix. Reordering the mesh can also improve efficiency (i.e. Cholesky Decomposition, reverse Cuthill-McKee reordering, etc.).

An approximate limit can be determined for model size if LU decomposition were to be used to solve equation 2 in real time. Assume that a computer that performs 20 million floating-point operations per second (flops) is to be used to run a real-time simulation. Also assume that graphical updates are to be maintained at 30 Hz and haptic updates should not fall bellow 300 Hz. Unfortunately, LU decomposition does not facilitate decoupling of graphical and haptic calculations, so model deformation calculations must also be determined at 300 Hz. LU components can be determined during a preprocessing stage in order not to effect run time

performance. Even if an 80% efficiency increase is assumed due to sparse matrix solving algorithms, the maximum number of nodes that could be used given the above requirements is 192 nodes (solving for $(0.2)(n^2)(300)=2.0E+07$ where n is three times the number of nodes for 3D simulation). Such a small model is unsuitable for most surgery simulation applications. These limits also ignore additional software overhead that might be required to run a simulator (i.e. rendering surface polygons and texture maps, collision detection, etc.).

B. Using the Inverse of the Stiffness Matrix

When simulation suturing, the f vector of equation 2 is typically very sparse with nonzero values only occurring at locations pertaining to applied sutures. In addition, there are usually only a small number of contact points where reaction forces must be determined to support force feedback. Because of this, the use of a pre-computed inverse of the stiffness matrix at run time is more efficient than LU decomposition. The inverse of the stiffness matrix can be determined in approximately αn^3 computations, where n is three times the number of nodes and α refers to efficiency gains that can be achieved through sparse LU decomposition techniques. At run time when contact is made with virtual FE object, equation 2 can be rearranged to take the format given below.

$$\begin{Bmatrix} d_{\text{no contact}} \\ d_{\text{contact}} \end{Bmatrix} = \begin{bmatrix} Ki_{aa} & Ki_{ab}^{T} \\ Ki_{ab} & Ki_{bb} \end{bmatrix} \begin{Bmatrix} f_{\text{no contact}} \\ f_{\text{contact}} \end{Bmatrix} \qquad \text{(Eq. 3)}$$

In equation 3, the known variables are d_{contact} and $f_{\text{no contact}}$ and Ki is the inverse of K. It should be noted that Ki is almost 100% dense, so sparse matrix techniques that depend on a sparse Ki are not relevant. The size of each sub-matrix is indicated by subscripts a and b. For three-dimensional analysis, b is three time the number of contact nodes and a is equal to n-b. From equation 3, two equations can be derived to obtain the unknown variables at run-time.

$$f_{\text{contact}} = (Ki_{bb}^{-1})(d_{\text{contact}} - Ki_{ab} f_{\text{no contact}}) \quad \text{(Eq. 4)}$$

$$d_{\text{no contact}} = Ki_{aa} f_{\text{no contact}} + Ki_{ab}^{T} f_{\text{contact}} \quad \text{(Eq. 5)}$$

The format of equation 4 and 5 illustrate how a sparse *f* vector can significantly reduce computation. To obtain reaction forces from equation 4, b^3+b^2+bl calculations are required where l is the number of nonzero $f_{no\ contact}$ variables. To find the nodal displacements, an additional $al+ab$ calculations are necessary.

It is now possible to determine model limitations using a pre-computed inverse of the stiffness matrix and equations 4 and 5. Assume again that a 20 million-flop machine is being utilized with a very primitive suturing simulator. A simple suture might be modeled as a spring that ties together two nodes (modeling sutures through the force vector will be addressed in more detail later). If a maximum of ten sutures are to be applied, l will be no greater than 60 (20 suture nodes multiplied by three degrees of freedom). The derivation of equations 4 and 5 has allowed force feedback calculations to be decoupled from deformation equations, so both graphical and haptic limits are determined separately. If haptics are to be maintained at 300 Hz, contact must be limited to 13 nodes (solving for $(b^3+b^2+60b)(300)$= 2.0E+07 where the maximum number of contact nodes is $b/3$). Haptic calculations are only dependent on the number of contact nodes and not the size of the model. Ignoring any additional software overhead and assuming the that maximum number of nodes are in contact, a graphical update rate of 30 Hz could be maintained for models with 2,257 nodes (solving for $(60a+39a)*30$=2.0E+07 where the maximum number of nodes is $(a+b)/3$). So long as the computer running the simulation had at least 92 Mbytes of RAM to hold the inverted stiffness matrix, a 2,257 node model that allows 13 nodes in simultaneous contact might be effective for

simulating suturing. However, limiting contact only to the location of nodes conflicts with the requirements outlined by the authors in section II.

C. Condensation

Given the simple scenario outlined above, the use of a pre-computed inverse of the stiffness matrix and equations 4 and 5 might be adequate for simulating suturing in some situations. However, performance can be improved if another simple step is taken in the preprocessing stage. In most cases, it is not necessary to determine the displacement of nodes that cannot be seen or touched. It makes sense to remove these nodes from run-time computation. This was the approach taken by Bro-Nielsen et al as discussed in section III.

$$\begin{bmatrix} K_{aa} & K_{ab} \\ K_{ab}{}^{T} & K_{bb} \end{bmatrix} \begin{Bmatrix} d_a \\ d_b \end{Bmatrix} = \begin{Bmatrix} f_a \\ f_b \end{Bmatrix} \quad \text{or} \quad \begin{aligned} K_{aa} * d_a + K_{ab} * d_b &= f_a \\ K_{ab}{}^{T} * d_a + K_{bb} * d_b &= f_b \end{aligned} \qquad \text{(Eq. 6)}$$

In equation 6, *a* corresponds to the portion of the stiffness matrix that will be condensed out and *b* corresponds to the portion that will be kept. If the nodes of the model are ordered so that the visible nodes are listed last, *b* will correspond to the visible nodes. Equation 6 can then be rearranged to take the form of equation 7.

$$\begin{aligned} f_b &= K_{ab}{}^{T} * K_{aa}{}^{-1} * f_a + (K_{bb} - K_{ab}{}^{T} * K_{aa}{}^{-1} * K_{ab}) * d_b \\ f_b &= P_f * f_a + P_d * d_b \end{aligned} \qquad \text{(Eq. 7)}$$

where $P_f = K_{ab}{}^{T} * K_{aa}{}^{-1}$ and $P_d = K_{bb} - K_{ab}{}^{T} * K_{aa}{}^{-1} * K_{ab}$

The size of P_f is *bxa* and the size of P_d is *bxb*. It is assumed that f_a remains constant at run-time. Better yet, f_a will equal zero for most application. Equation 7 can therefore be reduced to the following:

$$\begin{aligned} P_d d_b &= f_b \\ &\text{or} \\ K_{vis} d_{vis} &= f_{vis} \end{aligned} \qquad \text{(Eq. 8)}$$

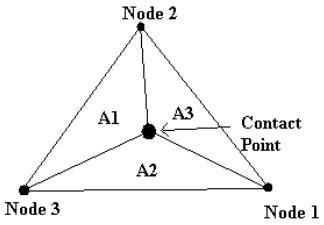
The format of equation 8 is very similar to that of equation 2, however, the stiffness matrix has been reduced from an *n*x*n* matrix to a *v*x*v* matrix, where *v* is three times the number of visible nodes. By sacrificing time during a preprocessing stage for condensation, performance improves at run-time since the number of non-visible nodes will not affect computation speed. If the inverse of K_{vis} were used in the suturing simulator described earlier, a model of 2,257 visible nodes would run in real-time rather than a model of 2,257 total nodes. This can be a significant improvement in model resolution and reduces the negative effects associated with contact limited to node location.

In the interest of rapid preprocessing, it is important to consider the methodology used to obtain the inverse of a condensed matrix. LU decomposition that takes advantage of the sparsity of the stiffness matrix is ideal for large meshes. However, a condensed stiffness matrix can become very dense. In this case, it is better to use Cholesky decomposition, which takes advantage of the fact that an FE stiffness matrix is symmetric and positive definite. For dense matrices, Cholesky decomposition is a factor or two faster than sparse LU decomposition [23].

D. Constraints

The disadvantage of the methods described above is that contact must take place at the nodes. If model resolution is not high enough, the user may notice when contact "jumps" to the location of the nearest node. This can have a detrimental effect on performance. By using the *constraints approach*, contact can be accommodated anywhere on the surface of the model.

Given a contact point on the face of any surface tetrahedral element, three nodes on that face can be constrained through a shape function [24].



$$x = \frac{A_1}{A}x_1 + \frac{A_2}{A}x_2 + \frac{A_3}{A}x_3$$

or (Eq. 9)

$$x = \lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3$$

In equation 9, the triangle formed by three nodes on a tetrahedral element face can be sub-sectioned into three individual triangles given the position of a contact point. The same shape functions can be used to map the *y* and *z* dimensions. Note that when the contact point is at node 1, $\frac{A_1}{A} = \lambda_1 = 1,\ \lambda_2 = \lambda_3 = 0$ and equation 9 yields $x = x_1$. Shape functions are unique for each element type, so the above equation is only relevant to 2D triangular or 3D tetrahedral elements.

Because of the linear nature of the problem and the fact that similar shape functions are used to create tetrahedral elements, these shape functions also can map displacement as shown in equation 10.

$$\begin{bmatrix} \lambda_1 & 0 & 0 & \lambda_2 & 0 & 0 & \lambda_3 & 0 & 0 \\ 0 & \lambda_1 & 0 & 0 & \lambda_2 & 0 & 0 & \lambda_3 & 0 \\ 0 & 0 & \lambda_1 & 0 & 0 & \lambda_2 & 0 & 0 & \lambda_3 \end{bmatrix} \begin{Bmatrix} dx_1 \\ dy_1 \\ dz_1 \\ dx_2 \\ dy_2 \\ dz_2 \\ dx_3 \\ dy_3 \\ dz_3 \end{Bmatrix} = \begin{Bmatrix} dx \\ dy \\ dz \end{Bmatrix} = d_{contact} \quad \text{(Eq. 10)}$$

The subscripts of equation 10 refer to the constrained nodes on the element face where contact occurs. To set up a constraint problem, the shape functions of equation 10 are used, however, they are written to reflect every node in the finite element model. The matrix above is expanded with zeros in matrix locations that correspond to nodes not constrained by the shape functions. Using the displacement vector d of equation 2 and C to define the expanded matrix holding the shape functions and additional zeros, the displacement at a contact point can be defined in the following:

$$Cd = d_{\text{contact}} \tag{Eq. 11}$$

The entire constraint problem can now be written by combining equation 2 and 11.

$$\begin{bmatrix} K & C^T \\ C & 0 \end{bmatrix} \begin{Bmatrix} d \\ f_{\text{contact}} \end{Bmatrix} = \begin{Bmatrix} f \\ d_{\text{contact}} \end{Bmatrix} \tag{Eq. 12}$$

The unknowns of equation 12 are f_{contact} and d, while f and d_{contact} are known after contact occurs. While the inverse of the matrix in equation 12 could be used to solve for f_{contact} and d, this would be ill advised since the variables of C are not defined until contact occurs. For large models, it would not be possible to determine the inverse of this matrix in real time.

$$f_{\text{contact}} = (CK^{-1}C^T)^{-1}(CK^{-1}f - d_{contact}) \tag{Eq. 13}$$

$$d = K^{-1}(f - C^T f_{\text{contact}}) \tag{Eq. 14}$$

Equations 13 and 14 are derived from equation 12 and allow the reaction force at a contact point to be determined along with the displacement of the nodes. This unique arrangement promotes the use of sparse matrix solving routines. The order in which these equations are solved is crucial for minimizing computation expense. Contact at only one point will first be considered to simplify explanation.

In equation 13, sparse matrices and vectors enclose K^{-1}. Only nonzero elements of C are multiplied with K^{-1}. However, it is not necessary to multiply nonzero values of C with every column of K^{-1}. Many rows in C^{T} and f are filled with only zeros. For this reason columns of K^{-1} that correspond to these nonzero rows can be ignored. An example of this is illustrated in the following:

$$\begin{bmatrix} \lambda_1 & 0 & 0 & 0 & 0 & 0 & \lambda_2 & 0 & 0 & \lambda_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & \lambda_1 & 0 & 0 & 0 & 0 & 0 & \lambda_2 & 0 & 0 & \lambda_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & \lambda_1 & 0 & 0 & 0 & 0 & 0 & \lambda_2 & 0 & 0 & \lambda_3 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} K^{-1}_{1,1} & K^{-1}_{1,2} & K^{-1}_{1,3} & K^{-1}_{1,4} & K^{-1}_{1,5} & K^{-1}_{1,6} & K^{-1}_{1,7} & K^{-1}_{1,8} & K^{-1}_{1,9} & K^{-1}_{1,10} & K^{-1}_{1,11} & K^{-1}_{1,12} & K^{-1}_{1,13} & K^{-1}_{1,14} & K^{-1}_{1,15} \\ K^{-1}_{2,1} & . & & & & & & & & & & & & & . \\ K^{-1}_{3,1} & & . & & & & & & & & & & & & . \\ K^{-1}_{4,1} & & & . & & & & & & & & & & & . \\ K^{-1}_{5,1} & & & & . & & & & & & & & & & . \\ K^{-1}_{6,1} & & & & & . & & & & & & & & & . \\ K^{-1}_{7,1} & & & & & & . & & & & & & & & . \\ K^{-1}_{8,1} & & & & & & & . & & & & & & & . \\ K^{-1}_{9,1} & & & & & & & & . & & & & & & . \\ K^{-1}_{10,1} & & & & & & & & & . & & & & & . \\ K^{-1}_{11,1} & & & & & & & & & & . & & & & . \\ K^{-1}_{12,1} & & & & & & & & & & & . & & & . \\ K^{-1}_{13,1} & & & & & & & & & & & & . & & . \\ K^{-1}_{14,1} & & & & & & & & & & & & & . & . \\ K^{-1}_{15,1} & . & . & . & . & . & . & . & . & . & . & . & . & . & . & K^{-1}_{15,15} \end{bmatrix} \begin{Bmatrix} f_{1x} \\ f_{1y} \\ f_{1z} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ f_{2x} \\ f_{2x} \\ f_{2x} \\ 0 \\ 0 \\ 0 \end{Bmatrix} \quad \text{(Eq. 15a)}$$

which is the same as

$$\begin{bmatrix} \lambda_1 & 0 & 0 & \lambda_2 & 0 & 0 & \lambda_3 & 0 & 0 \\ 0 & \lambda_1 & 0 & 0 & \lambda_2 & 0 & 0 & \lambda_3 & 0 \\ 0 & 0 & \lambda_1 & 0 & 0 & \lambda_2 & 0 & 0 & \lambda_3 \end{bmatrix} \begin{bmatrix} K^{-1}_{1,1} & K^{-1}_{1,2} & K^{-1}_{1,3} & K^{-1}_{1,10} & K^{-1}_{1,11} & K^{-1}_{1,12} \\ K^{-1}_{2,1} & K^{-1}_{2,2} & K^{-1}_{2,3} & K^{-1}_{2,10} & K^{-1}_{2,11} & K^{-1}_{2,12} \\ K^{-1}_{3,1} & K^{-1}_{3,2} & K^{-1}_{3,3} & K^{-1}_{3,10} & K^{-1}_{3,11} & K^{-1}_{3,12} \\ K^{-1}_{7,1} & K^{-1}_{7,2} & K^{-1}_{7,3} & K^{-1}_{7,10} & K^{-1}_{7,11} & K^{-1}_{7,12} \\ K^{-1}_{8,1} & K^{-1}_{8,2} & K^{-1}_{8,3} & K^{-1}_{8,10} & K^{-1}_{8,11} & K^{-1}_{8,12} \\ K^{-1}_{9,1} & K^{-1}_{9,2} & K^{-1}_{9,3} & K^{-1}_{9,10} & K^{-1}_{9,11} & K^{-1}_{9,12} \\ K^{-1}_{10,1} & K^{-1}_{10,1} & K^{-1}_{10,1} & K^{-1}_{10,10} & K^{-1}_{10,11} & K^{-1}_{10,12} \\ K^{-1}_{11,1} & K^{-1}_{11,2} & K^{-1}_{11,3} & K^{-1}_{11,10} & K^{-1}_{11,11} & K^{-1}_{11,12} \\ K^{-1}_{12,1} & K^{-1}_{12,2} & K^{-1}_{12,3} & K^{-1}_{12,10} & K^{-1}_{12,11} & K^{-1}_{12,12} \end{bmatrix} \begin{Bmatrix} f_{1x} \\ f_{1y} \\ f_{1z} \\ f_{2x} \\ f_{2y} \\ f_{2z} \end{Bmatrix} \quad \text{(Eq. 15b)}$$

By multiplying only nonzero values, the total calculations required to solve equation 15 is reduced from 720 to 72. Table 2 and 3 describes the order in which equations 13 and 14 should be solved and the number of calculations is defined for each step.

In the case where there is more than one contact point, equation 9 is used to add additional rows to C. Only one point of contact should occur per element face, and each contact point will add three rows to C. Multiple contact points on a single face can be averaged. As an example, three points of contact on three separate element faces would require C to have nine rows and there would be 27 nonzero values.

Assume again that a computer that performs 20 million flops is to be used for simulating suturing. Unlike the earlier scenarios described, sutures can now be placed anywhere on the model. If a suture is modeled as a simple spring that ties two points together and the force exerted at each point is distributed between three nodes, there will be 18 applied forces per suture (to be discussed in more detail in section V). If a maximum of ten sutures are to be applied, l will be no greater than 180. From table 2, it can be seen that $27c^3+63c^2+12cl$ operations are required to determine the reaction forces at the contact nodes, where c is the number of contact points. With this scenario, a total of 10 simultaneous contact points could be applied at a haptic update rate of 300 Hz (solving for $(18c^3+63c^2+12cl)(300)=2.0E+07$ where l equals 180). If 12 points of contact exist, a maximum number of 1,058 nodes could be accommodated if graphical updates are to be maintained at 30 Hz (solving for $(9c+n(3c+l))(30)= 2.0E+07$ where l equals 180 and c is equal to 10). If the inverse of K_{vis} is used in equations 12 through 14 instead of the inverse of K, a model with 1,058 visible nodes could be solved in real-time. The number of contact points is not affected by using the inverse of K_{vis} instead of K. Suturing can accurately be simulated under these limitations and the requirement defined in section III can be satisfied.

At first glance, it may seem that we have lost model resolution by using constraints instead of the inverse of a condensed stiffness matrix. However, in the case described earlier,

180 dynamically changing forces are being applied rather than just 60 as described for node contact. This was necessary to allow 10 sutures to be placed anywhere on the model. If only 60 forces were applied, 2,314 surface node models could be solved in real-time with simultaneous contact at 10 points (as compared to 2,257 surface nodes when using condensation and the inverse of the stiffness matrix). Therefore, for this type of scenario, a bit more resolution has been gained in addition to the ability to touch a model anywhere on its surface.

V. SUTURING SIMULATION APPLICATION

The suturing simulator that is currently under development by the authors does not yet reflect all the suture application steps of table 1. This is a work in progress, however, even in its present state, the proof of concept for many features of the method outlined above can be demonstrated.

The authors have created a *Fast FE Modeling Software Platform* that allows interaction with real-time FE models based on the constraints approach. A 1 GHz Athelon processor PC is being utilized with an Oxygen GVX video card. Currently, only one *Phantom Premium* device is being used for force-feedback. Shutter glasses allow models to be viewed in stereo through a monitor or through a *Reach-In* system that allows virtual images to be co-located with the users hands as shown in figure 7. Models can also be viewed in an augmented reality environment through software based on the AR Tool Kit [25-28]. In the augmented environment, deformation results are sent over the Internet and can be viewed remotely. This could allow for remote instruction and peer evaluation of a surgical procedure.

The suturing simulator typically utilizes a model of a *hand* that has a laceration on the palm. Figure 5 shows multiple image captures of the arm as the laceration is closed with three sutures. This model was developed from MRI scans which were used to generate an implicit

model. The implicit model was meshed with a technique called point repulsion [29, 30]. The details of the authors' model generation process have been described in a previous publication [2]. The bone surface is represented with fixed boundary nodes. The various soft tissue layers have not yet been segmented and are currently represented as one homogenous tissue. Material properties were roughly approximated using values from the literature [31]. Nodal resolution is highest near the wound for greater modeling accuracy at the region of interest.

There are various options for viewing the model in the Fast FE Modeling Software Platform. One useful feature is real-time stress-strain visualization. Since it is important to minimize the stress inflicted on tissue during every surgical procedure, it is helpful to be able to visualize these stresses. Not only does stress-strain monitoring allow peak tissue stresses to be recorded for procedure assessment, but also the final results of a procedure can be evaluated through the color plots of stress and strain. Excessive tissue stress can lead to scarring and improper suture placement can be identified through the visualization of excessive stress concentrations (see figure 6).

Virtual tools determine how you interact with a model. A real-time FE model can be manipulated by applying forces and/or displacements. When building a suturing simulator it is necessary to develop complex tools that imitate needles and sutures. Determining how needles and sutures impart forces and displacements to tissue is by no means trivial, however, the constraints approach is appropriate for accommodating such complex interaction.

The *needle driver* tool is loaded as a dynamic link library. It is based on a generic parent class that follows a structure common to all tools used in the Fast FE Modeling Software Platform. Displayable polygons, texture map and touchable points are typical class elements. Contact with the needle driver is initially limited to the tip of the needle. When the needle tip

touches the surface of a model, a vector representing the orientation of the needle with respect to the model surface is determined. The dot product between the needle orientation vector and the surface normal is multiplied with the applied force magnitude. If this value exceeds a prescribed needle puncture force, the needle "pops" through the skin. This process emphasizes proper needle orientation which is one criteria commonly used in evaluating suturing. The user has the option to visualize the needle orientation vector to assist with aligning the needle. When visualizing the needle alignment vector, the needle color will change to indicate proper alignment.

In the current simulator, the needle enters and exits at the same point. While the needle can be inserted anywhere on the model surface, multipoint interaction has not yet been incorporated as outlined in the steps of table 1. After puncture, the contact point is secured to the shaft of the needle. Some simple rules that incorporate friction and needle rotation determine how the contact point slides along the needle. Complete needle rotation causes the needle to pass out of the tissue.

Sutures are graphically represented as line vectors. After passing the needle through the tissue on one side of the wound, a suture is rendered as a line that connects the needle to the first anchor point. The suture does not offer any resistance at this stage. After the needle has been passed through the tissue on the opposite side of the wound, the suture can be pulled tight in order to close the wound. The first anchor point on the skin is considered fixed to the end of the suture. The second anchor point is treated as a "pulley" that allows the suture to slide through, but will also bear a load. This way tension applied to the suture can draw the two anchor points together. An applied displacement based on pulling the needle moves the first anchor point and an applied force resulting from resistance at the "pulley" moves the second anchor point.

Pressing a key on the keyboard cuts the suture and automatically ties the knot. This type of modeling was used because the current simulator system only has one force feedback device.

After the suture is tied, it is modeled as a stiff spring that only supports tension. The force that the suture applies is based on the displacement of the two suture anchoring points. These applied suture forces are not solved simultaneously with model deformation, but are based on deformations determined during previous time steps. An average force is resolved over ten time steps and is then inserted into the applied force vector. Averaging over multiple steps is crucial for avoiding instabilities. Since suture forces are applied on face points rather than at nodes, a suture force is distributed between face nodes based on shape functions after needle contact (see equation 9). Sutures can be placed anywhere on the model and do not have to be placed in a position that closes the wound. Any large number of sutures can be applied because contact currently occurs at no more than one point at a time.

The suturing process described above was initially modeled in this simple fashion because the simulation system utilized only contains one force-feedback device. The addition of a second force-feedback device will allow much more realistic modeling as outlined in table 1. However, even the current simulator has value in that it allows criteria such as tissue stress, needle orientation and suture placement to be evaluated. The prime emphasis of the simulator is to demonstrate that accurate deformation and force-feedback can be achieved through the constraints approach.

VI. DISCUSSION AND CONCLUSIONS

The constraints approach has proved to be well suited for simulating suturing. Even though the hand model used contains only 863 total nodes, well below what is capable using constraints, informal inquiry of dermatological surgeons indicates that model resolution and

deformation accuracy is suitable for suture simulation training. The deformation of the current arm model can easily be rendered at 30 Hz and force-feedback is maintained above 1000 Hz (which is a requirement when using a phantom force feedback device).

While modeling size limitations may be comparable for node and face contact, simulation performance improves when using face contact. The authors observed this when comparing the current simulator to previous simulators created by the authors where contact was limited to the surface nodes. When contact is limited to the nodes, a model with higher node resolution is required. Otherwise, adjacent nodes will "jump" to the point of contact and this sudden deformation shift and reaction force is distracting. In addition, models with higher nodal resolution require more preprocessing time and demand extra storage and memory. Face contact also results in better surgical tool interaction with soft tissue, such as when positioning a needle for insertion. Accurate contact can be important for a variety of surgical procedures.

Creating a fast FE model through constraints requires minimal preprocessing time. When using a 1 GHz Athelon processor based PC, it took 4 minutes and 9 seconds to condense the original stiffness matrix and take the inverse of the resulting condensed matrix. All preprocessing was performed using a single thread application. Significant performance gains could be achieved using parallel processing if a machine with multiple processors was utilized.

The constraints approach allows for applied forces that change dynamically. This is what made it possible for us to model the sutures without altering the FE stiffness matrix. The ability to apply loads as needed enables robust interaction with the model. Not only do flexible boundary conditions allow for better modeling of tool interaction, but environmental factors, such as gravity, might also be approximated through the applied force vector.

Work is underway to increase the modeling accuracy of the suturing tools. Multipoint contact will allow suturing to be simulated with the steps described in table 1. It is the intention the authors to add a second force-feedback device so that both a needle driver and skin hook can be used. Farther down the road, the authors plan to tackle the complex issue of knot tying. The sutures themselves could be modeled with fast FE analysis.

The major limitation of the constraints approach is that the original undeformed mesh cannot be easily altered. Substantial changes to the original mesh structure mean that the entire stiffness matrix must be updated, condensed and inverted again. Most surgical procedures involve the cutting of soft tissue, which necessitates real-time mesh alterations. Non-linear material property representation also requires updates in real-time to the FE stiffness matrix. In its present state, the constraints approach is not appropriate for such modeling scenarios. For this reason the authors are developing alternative fast FE methodologies that allow real-time mesh updates. Initial research indicates that these new methodologies will not allow the modeling resolution achievable through constraints. However, modeling resolution will be sufficient for many simulation applications. Most surgical procedures involve isolated cutting in particular regions of interest. These regions could be modeled with dynamically changing meshes while exterior regions could be modeled with constraints. It is expected that these hybrid models will meet the demands of a wider variety of surgery simulation applications.

VII. BIBLIOGRAPHY

1. *An index of numerous reconstructions based on the Visible Human Dataset can be found at www.nlm.gov/research/visible/visible_human.html*, .
2. Berkley, J., Oppenheimer, P., Weghorst, S, Berg, D., Raugi, G, Haynor, D., Ganter, M., Brooking, C., Turkiyyah, G. *Creating Fast Finite Element Models from Medical Images*. in *Medicine Meets Virtual Reality 2000,*. 2000. Newport beach, CA.
3. Berkley, J., Weghorst, S., Gladstone, H., Raugi, G., Berg, D., Ganter, M., *Banded Matrix Approach to Finite Element Modeling for Soft Tissue Simulation.* Virtual Reality, 1999. **4**: p. 203-212.
4. Berg, D., Raugi, G, Gladstone, H., Berkley, J., Ganter, M., Turkiyyah, G. *Virtual Reality Simulators for Dermatologic Surgery Measuring Their Validity As A Teaching Tool*. in *Medicine Meets Virtual Reality 2001*. 2001. Newport Beach, CA.
5. Larrabee, W.F., Jr. and D. Sutton, *A finite element model of skin deformation. II. An experimental model of skin deformation.* Laryngoscope, 1986. **96**(4): p. 406-12.
6. Pieper, S., D. Laub, and J. Rosen, *A finite element facial model for simulating plastic surgery.* Plast Reconstructive Surg, 1995. **96**(5): p. 1100-1105.
7. Gourret, J., N. Thalmann, and D. Thalmann. *Simulation of object and hand skin deformation in a grasping task*. in *SIGGRAPH*. 1989.
8. Chen, D. and D. Zeltzer. *Pump it up: Computer animation of of biomechanically based model of muscle using the finite element method*. in *SIGGRAPH*. 1992.
9. Waters, K. and D. Terzolpolos, *The computer synthesis of expressive faces.* Philos. trans. R. Soc. Lond., 1992. **335**(1273): p. 87.
10. Keeve, G. and E. Girod. *Craniofacial surgery simulation*. in *Proceedings of 4th International Conference on Visualization in Biomedical Computing*. 1996.
11. Cotin, S., *et al.*, *Geometric and physical representations for a simulator of hepatic surgery.* Stud Health Technol Inform, 1996. **29**: p. 139-51.
12. Cotin, S., H. delingette, and N. Ayache, *Efficient linera elastic models of soft tissue for real-time surgery simulation*, . 1998, INRIA, Institute national de Recherche en Informatique et en Automatique.
13. Picinbono, G., *et al. Anisotropic elasticity and force extrapolation to improve realism of surgery simulation*. in *IEEE International Conference on Robotics and Automation*. 2000. San Francisco.
14. Bro-Nielsen, M., *Fast finite elements for surgery simulation.* Stud Health Technol Inform, 1997. **39**: p. 395-400.
15. Hansen, K. and O. Larsen, *Using region-of-interest based finite element modeling for brain-surgery simulation.* Lecture Notes in Computer Science, 1998. **1496**: p. 305.
16. James, D. and D. Pai, *A Unified treatment of Elastostatic and Rigid Contact Simulation for Real Time Haptics.* The Electronic Journal of Haptics Research (www.haptics-e.org), 2001.

17. Sagar, M. and D. Bullivant. *A Virtual Environment and Model of the Eye for Surgical Simulation*. in *SIGGRAPH*. 1996.

18. Zhuang, Y. and J. Canny. *Haptic interaction with global deformations*. in *IEEE International Conference on Robotics and Automation*. 2000. San Francisco.

19. Basdogan, C. *Real-Time Simulation of Dynamically Deformable Finite Element Models Using Modal Analysis and Spectral Lanczos Decomposition Methods*. in *Medicine Meets Virtual Reality*. 2001. Newport Beach, CA.

20. Wu, X., Goktekin,T., Tendrick, F., *Adaptive Nonlinear Finite Elements for Deformable Body simulation Using Dynamic Progressive Meshes.* Eurographics, 2001. **20**(3).

21. Szekely, G., , vol. 9: 3, pp. 236-255, 2000., *Virtual Reality-Based Simulation of Endoscopic Surgery.* Presence, 2000. **9**(3): p. 236-255.

22. Cavusoglu, M.C., Tendrik, F. *Multirate Simulation for High Fidelity Haptic Interaction with Deformable Objects in Virtual Environments*. in *IEEE Intl. Conf. Robotics and Automation*. 2000.

23. Press, W.H., *et al.*, *Numerical Recipes in C*. 2nd ed. 1992, Cambridge: Cambridge University press.

24. Zienkiewicz, O.C. and R.L. Taylor, *The Finite Element Method*. 4th ed. Vol. 1. 1994, London: McGraw-Hill Book Company.

25. Kato, H., Billinghurst M. *Marker Tracking and HMD Calibration for a video-based Augmented Reality Conferencing System*. in *2nd International Workshop on Augmented Reality (IWAR 99)*. 1999.

26. Billinghurst, M. and H. Kato, *Real World Teleconferencing.* Proceedings of CHI '99, 1999(May 19-20, Pittsburgh, PA USA).

27. Billinghurst, M. and H. Kato, *Shared Space.* http://www.hitl.washington.edu/research/shared_space/download/, 1999.

28. Billinghurst, M., et al. *Mixing Realities in Shared Space: An Augmented Reality Interface for Collaborative Computing*. in *IEEE International Conference on Multimedia and Expo (ICME2000)*. 2000.

29. Lim, C.T., *et al.*, *Object Reconstruction from Layered Data using Implicit Solid Modelling.* Journal of Manufacturing Systems special issue on Layered Manufacturing, 1997. **16**(4): p. 260-272.

30. Brooking, C., *Point repulsion in implicit solids.* In-house publication, Department of Mechanical Engineering, University of Washington, brooking@u.washington.edu., 1999.

31. Berkley, J., *Determining soft tissue material properties for the purpose of finite element modeling of the below kinee amputee residual limb.*, in *Engineering*. 1997, Northwestern University: Chicago.

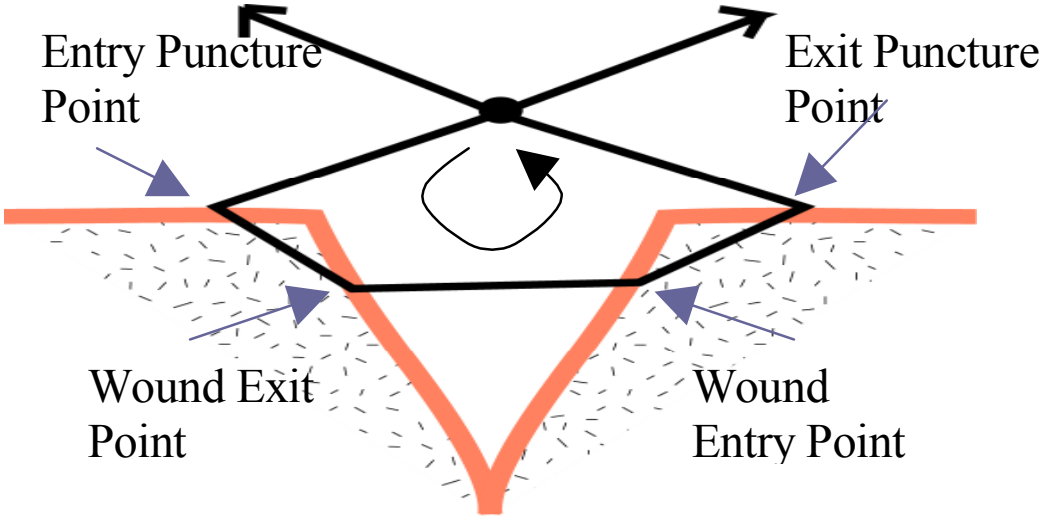


Figure 1. A graphical representation (side view) of a suture with four connecting points on the wound and a knot. Assume the needle was inserted on the left side of the wound and pulled through on the right.

Step 1	If the applied needle force normal to the skin surface exceeds a prescribed puncture force, the needle is allowed to pass through the skin surface. Rotation of the needle cause the "entry puncture point" to slide along the axis of the needle. The displacement at the entry puncture point, and resulting deformation of the tissue, is determined by displacement of the needle.
Step 2	When the needle passes out of the wound, the displacement of the "wound exit point" becomes confined to the needle. Rotation of the needle causes the wound exit point to slide along the axis of the needle.
Step 3	The needle is released by the needle holder and then grabbed again from within the wound.
Step 4	The needle is pulled completely through the tissue. As the needle travels completely through the entry puncture point and though the wound exit point, the confined displacements of each of these points are released from the needle.
Step 5	The needle is pushed through the inside of the wound on the opposite side. Rotation of the needle cause the "wound entry point" to slide along the axis of the needle. Displacement of this point is confined to the needle.
Step 6	The needle must now be pushed out through the skin surface. If the applied force normal to the skin surface exceeds a prescribed puncture force, the needle is allowed to pass out through the skin surface. The "exit puncture point" becomes confined to the needle along with the wound entry point.
Step 7	The needle is released and the grabbed again from outside the skin surface.
Step 8	The needle is pulled completely through the tissue. As the needle travels through the wound entry point and through the exit puncture point, the confined displacements of each of these points are released from the needle.
Step 9	A knot is tied and the suture is pulled tight closing the wound.

Table 1. The required modeling steps for the application of one suture to an existing laceration or excision.

	Portion of Equation 13	*Number of Computations*	*Comments*
Step 1	$[C][K^{-1}]$	$27c^2$	Only multiply with columns pertaining to non-zero rows of C^T.
Step 2	$[CK^{-1}][C^T]$	$27c^2$	
Step 3	$[CK^{-1}C^T]^{-1}$	$18c^3$	
Step 4	$[C][K^{-1}]$	$9cl$	Only multiply with columns pertaining to non-zero rows of f.
Step 5	$[CK^{-1}][f]$	$3cl$	
Step 6	$f_{contact} = [(CK^{-1}C^T)^{-1}][(CK^{-1}f - d_{contact})]$	$9c^2$	
	Total	$18c^3+63c^2+12cl$	

Table 2. The table above describes the order in which equation 13 should be solved. *c* refers to the number of contact points and *l* refers to the number of nonzero values in the force vector. Instead of using K, K_{vis} from equation 8 could be used instead, but there will be no performance increase. Brackets indicate that the enclosed portion has been solved in a previous step. Additions and subtractions have been ignored when defining the number of computations. It is assumed that each contact point is distributed over three unique nodes. It is also assumed that the row numbers pertaining to nonzero rows in C^T are completely different than the row numbers pertaining to nonzero rows of *f*. Otherwise, the number of computation could be reduced. The equations above assume a 3D problem.

	Portion of Equation 14	*Number of Computations*
Step 1	$[C^T][f_{contact}]$	$9c$
Step 2	$d = [K^{-1}][f - C^T f_{contact}]$	$n(3c+l)$
	Total	$9c+ n(3c+l)$

Table 3. The table above describes the order in which equation 14 should be solved. *c* refers to the number of contact points, *l* refers to the number of nonzero values in the force vector and *n* refers to the width/height of *K*. Instead of using *K*, K_{vis} from equation 8 could be used instead and *n* would equal three times the number of visible nodes. Brackets indicate that the enclosed portion has been solved in a previous step. Additions and subtractions have been ignored when defining the number of computations. It is assumed that each contact point is distributed over three unique nodes. The equations above assume a 3D problem.

Figure 2. A needle driver and tweezers are used to apply a suture. Contact takes place at multiple points, which must be accounted for in simulation.

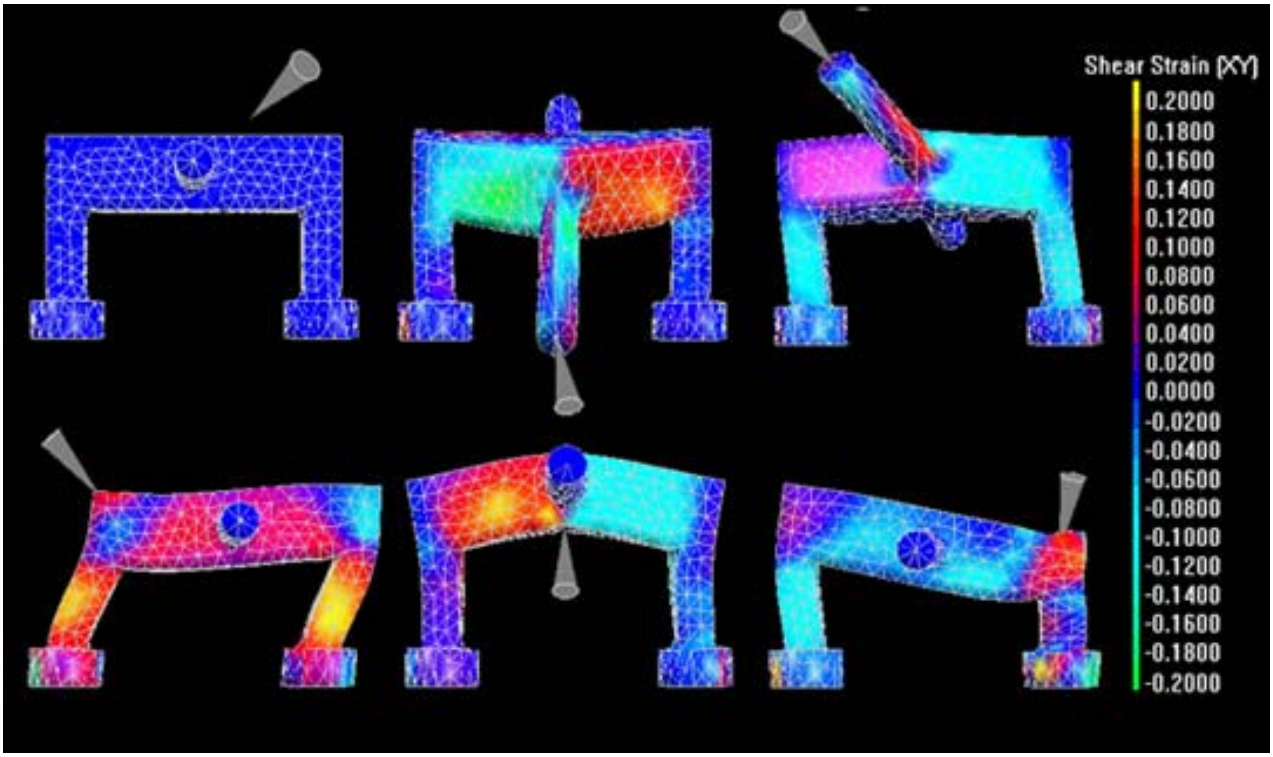


Figure 3. Shows an application created by the authors for real-time FE modeling in engineering analysis. The FE model is composed of tetrahedral elements and nodes that mark the corners of each element. A reaction force is calculated at the contact point, which is used for haptic feedback. The color plot is representative of the shear strain in the XY direction.

(a) (b)

Figure 4. a) Shows the Suturing Simulator in use within the Reach-In environment. b) The author remotely views the skin surgery simulator through a head mounted display. Remote viewing through augmented reality was accomplished by using the AR Tool Kit in conjunction with remote viewing socket software developed by Jeff Berkley.

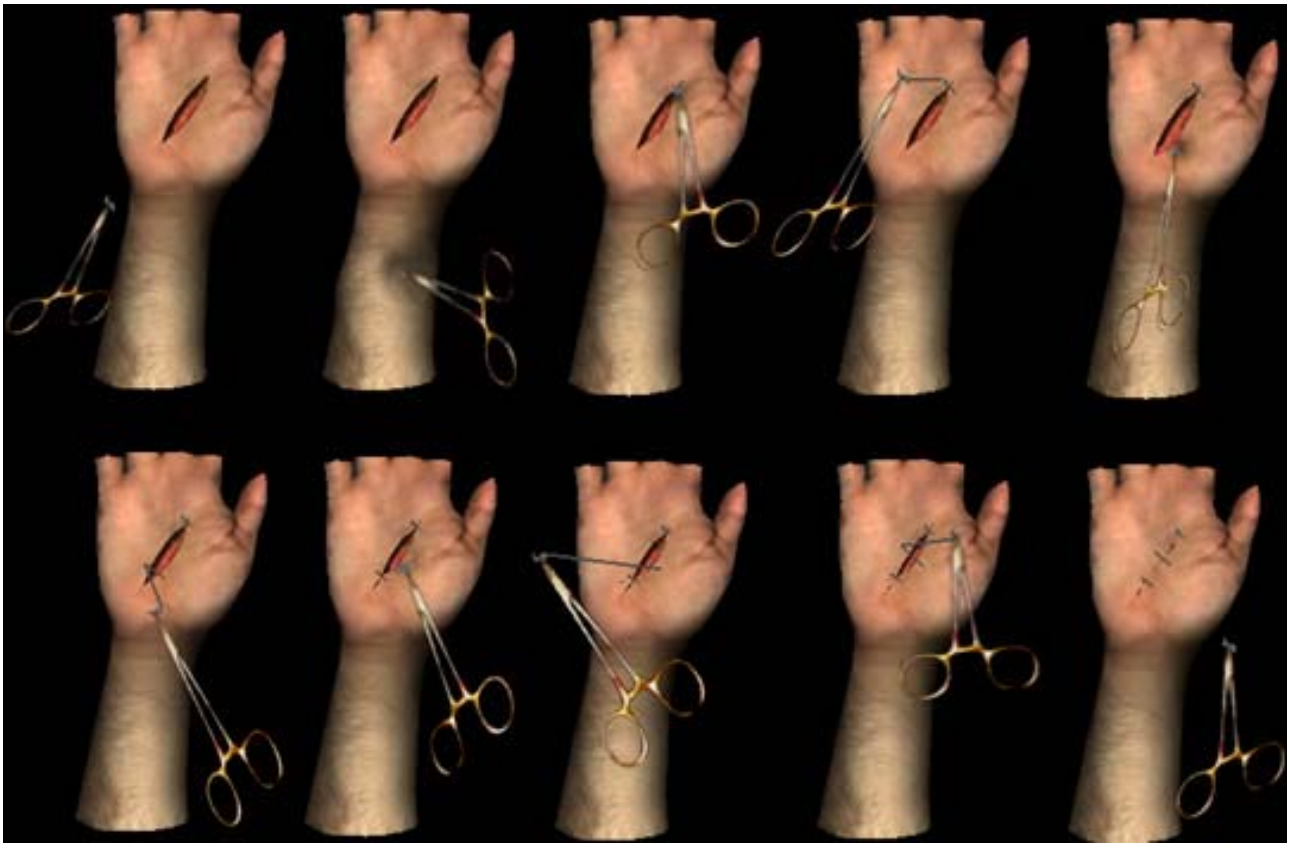
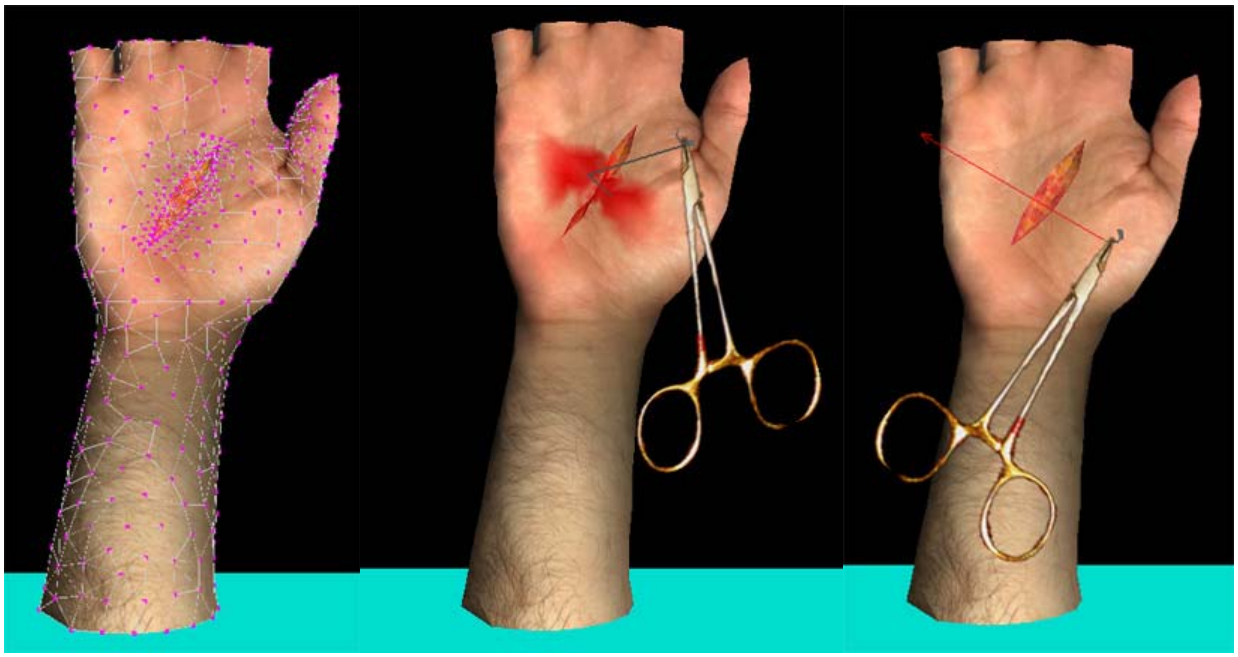


Figure 5. Shows a *hand* model as three sutures are applied to close an excision.



a **b** **c**

Figure 6. a) Shows the overlying mesh of a *hand* model with 863 nodes of which 624 nodes lie on the surface. Displacements are determined for the visible nodes and an additional 100 non-visible nodes that correspond to surface elements in order to allow real time stress/strain visualization. Greater element resolution exists at the wound to provide greater accuracy at the region of interest. b) Shows the arm model with stress magnitude color mapping. c) The red vector extending from the needle can be used to help the user orient the needle perpendicular to the skin for proper needle insertion.