

# Adaptive Finite Element Technique for Cutting in Surgical Simulation

Hualiang Zhong\*, Mark P. Wachowiak and Terry M. Peters

Imaging Research Laboratories, Robarts Research Institute  
100 Perth Drive, London, ON, Canada, N6A 5K8

## ABSTRACT

Pre-computed finite element methods are valuable because of their extreme speed and high accuracy for soft tissue modeling, but they are not suitable for surgical incision simulation. In this paper we present an adaptive algorithm for finite element computation based on a preprocessing approach. It inverts the global stiffness matrix in a pre-computing stage and then simulates each cutting step by updating two lists of basic components iteratively with some localization techniques. This method allows a fast and physically accurate simulation of incision procedures.

**Keywords:** Surgery Simulation, Finite Element Method, High Performance Computing

## 1. INTRODUCTION

A surgical simulation system can provide surgeons with a virtual environment for surgical training and rehearsal and can avoid using live subjects or animals for training purposes. Usually such a simulator should be able to simulate interventional actions such as probing, pulling, cutting and suturing, therefore accurately describing the response of soft tissue to these actions is important for building a realistic simulation system. Finite element methods (FEM) may provide a physics-based accurate approximation to the soft tissue deformation, but because such approaches require the solution of a large set of equations, their speed is generally a bottleneck, limiting real-time performance. To accelerate the speed of FEM-based approaches, several groups have developed heuristic models such as Chain-Mail [1] or mass-spring models [2;3], but since such models lack a rigorous physical basis, it is not easy to incorporate biophysical properties such as incompressibility, anisotropy or nonlinear elasticity into them. Recently many real-time finite element techniques have emerged in different applications to improve their real-time computing efficiency. Generally speaking, these techniques can be divided into two categories according to whether or not a preprocessing step is employed.

In order to approach the real time requirement of FEM in surgery simulation, Bro-Nielsen [4] first separates all the nodes into either surface or internal components, and then uses a condensation approach to reduce the size of the stiffness matrix. In [5], James and Pai employed a boundary element method to model a deformable object based on linear elasticity, while also allowing changes in boundary constraints in real time simulation based on the Woodbury formula [6]. Later Cotin [7] created a tensor for each free node so that during animation, the displacements of the contacted nodes could easily be used to 'predict' the displacements of the free nodes. This approach has achieved satisfactory speeds, and its computation only involves  $O(n)$  multiplication in a real time simulation.

However, these methods have serious restrictions. A typical example is that in the simulation of surgical incision, the topology of the mesh changes and the global stiffness matrix and its inverse must be updated. We know that updating these matrices is computationally expensive (at least  $O(n^2)$ ), so the above pre-computed finite element methods (PFE) are usually not considered to be applicable to the simulation of surgical incisions.

To account for these restrictions, Nienhuys used the conjugate gradient method [8-10] to simulate surgical cutting manipulations based on the sparse property of the stiffness matrix. This method iteratively updates the displacement vector so that the system moves to a configuration with minimal elastic energy. While this model places few restrictions on the topological changes of its mesh, or on the re-configuration of the global system, its efficiency is not adequate to meet the requirement for real time simulations, especially for a large system of equations.

---

\* hzhong@imaging.robarts.ca, phone 1-519-663-5777 ext. 34223, fax 1-519-663-3900; <http://www.imaging.robarts.ca>

In this paper, we propose a new pre-computed finite element algorithm that can be used in surgery incision simulations, and yet still achieve a satisfactory computing speed. Since for most applications, the dynamic components are small and a static solution looks quite realistic [4;11], here we consider the simulation of an incision manipulation as a steady-state problem.

## 2. AN ADAPTIVE ALGORITHM FOR FINITE ELEMENT COMPUTATION

The finite element method divides the problem domain into discrete elements with each element containing several nodes. The displacements of the internal points in the element are obtained by interpolating the nodes' displacements. The unknown displacements of these nodes are subjected to an elastic model, imposed force, and boundary constraints, and are usually calculated by solving a set of linear or non-linear algebraic equations. Here we only consider the linear equations derived from a both geometrically and elastically linear finite element model.

### 2.1. Three Dimensional Finite Element Method

In a full three-dimensional analysis, a strain tensor measures the change of distance between a deformed state and its reference state. In the linear case, the six strain components can be written as

$$\boldsymbol{\varepsilon} = [\varepsilon_x, \varepsilon_y, \varepsilon_z, \gamma_{xy}, \gamma_{yz}, \gamma_{zx}]^T = \left[ \frac{\partial u}{\partial x}, \frac{\partial v}{\partial y}, \frac{\partial w}{\partial z}, \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}, \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}, \frac{\partial w}{\partial x} + \frac{\partial u}{\partial z} \right]^T. \quad (1)$$

These components can be further substituted into the displacement term of each node for a given element. For a linear isotropic and homogenous elastic material model [12], the link between the strain and stress is assumed to be governed by the Hooke's law  $\boldsymbol{\sigma} = [G]\boldsymbol{\varepsilon}$  where the material matrix  $G$  is defined by:

$$G = \begin{pmatrix} \lambda + 2\mu & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda + 2\mu & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & \lambda + 2\mu & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{pmatrix} \quad (2)$$

and  $\lambda$  and  $\mu$  are the Lamé constants. Using either Galerkin's residue method or variational principles [13], we can obtain the following matrix equation for each element  $e_i$ :

$$K^{e_i} D^{e_i} = F^{e_i},$$

where  $K^{e_i}$  is the stiffness matrix ( $12 \times 12$  for a tetrahedron), and  $D^{e_i}$  and  $F^{e_i}$  are the vectors of the displacements and forces of the element's nodes. After assembling these individual forces at each node, the global equation can be written as

$$K[d_1 d_2 \dots d_{3n}]^T = [F_1 F_2 \dots F_{3n}]^T \quad (3)$$

where  $F_i$  are external forces,  $n$  is the number of the non-fixed nodes and  $K$  is the assembled global stiffness matrix.

There are many direct or iterative methods that can be used to solve (3), but here we use the direct LU decomposition to obtain  $K^{-1}$  in a preprocessing stage. The LU method involves  $O(n^3)$  operations and practically any method with

$O(n^2)$  or higher orders of complexity is unlikely to achieve a real time speed except for very small  $n$ . Therefore the preprocessing step may take from several minutes to hours to complete the LU decomposition.

## 2.2. Simulation of the Incision Behavior

When a scalpel cuts into an anatomical structure, one of the most important changes is the local continuity that is fundamental to the finite element method. As a result, when the scalpel progresses, the global deformation behavior can be quite different from that generated from the original continuous mesh. In order to realistically model this process, each cutting step should be small and the topological structure of the anatomy must be updated to reflect the loss of the continuity. There are several re-meshing mechanisms, such as volume subdividing or face adjustment [8;9] available to achieve a better visual reality. However these dynamic updating schedules usually have disadvantages such as increasing mesh size and reducing mesh quality, which generally increase the computational time and reduce the system's stability.

In this paper, we adopt the method employed in [14] by removing the tetrahedron most recently touched by the scalpel. Consequently the contributions from this tetrahedron to each vertex must be removed from the corresponding entries in the stiffness matrix. For example, if two vertices of a tetrahedron share only this tetrahedron, they will lose any link and the two corresponding stiffness entries will be null. Unfortunately updating an entry in  $K$  requires the modifications of all the  $n^2$  entries in  $K^{-1}$ , so the expense of this update is at least  $O(n^2)$ . The adaptive approach introduced below is based on the sparse property of  $K$  and  $F$  and it only requires  $O(n)$  operations.

## 2.3. Adaptive Algorithm

When a single tetrahedron is removed from a mesh, there are  $12 \times 12$  entries in the stiffness matrix that must be updated. The new matrix may no longer be regular or it may include null columns and rows. In the later case, these columns or rows must be removed from the matrix if we try to find its inverse. The Sherman-Morrison-Woodbury formula [6] is especially useful in updating the inverse matrix when a few entries in the matrix are modified. This formula may be written as

$$(A + B \cdot C^T)^{-1} = A^{-1} - \{A^{-1} \cdot B \cdot (I + C^T \cdot A^{-1} \cdot B) \cdot C^T \cdot A^{-1}\}. \quad (4)$$

To apply this formula to an incision simulation, let  $A$  be the  $3n \times 3n$  global stiffness matrix,  $B$  and  $C$  the  $3n \times 12$  matrices, expanded from the  $12 \times 12$  identity matrix and the corresponding updated  $12 \times 12$  individual stiffness matrix, respectively. Then the numbers of the non-zero entries in  $B$  and  $C$  are 12 and 144, respectively, and their positions are determined by the indices of four vertices of this tetrahedron.

Updating the inverse matrix using the above formula still involves  $O(n^2)$  multiplications and is unlikely to achieve real time performance if  $n$  is larger than a few hundred. So for each cutting operation we need to directly compute the displacements, instead of updating the inverse matrix. For the  $i$ -th cutting step, suppose the vector  $F_i$  represent the external forces imposed on the object. Then the displacements  $D_{i+1}$  of all the nodes could be generated as follows

$$D_{i+1} = A_{i+1}^{-1} \cdot F_i = A_i^{-1} \cdot F_i - M_i \cdot C_i^T \cdot A_i^{-1} \cdot F_i \quad (5)$$

where  $A_i^{-1} \cdot F_i$  and  $M_i$  can be simultaneously calculated from the following iterative equations:

$$A_{i-k+1}^{-1} \cdot F_i = A_{i-k}^{-1} \cdot F_i - M_{i-k} \cdot C_{i-k}^T \cdot A_{i-k}^{-1} \cdot F_i, \quad k = 1, \dots, i, \quad (6)$$

$$M_i = A_i^{-1} \cdot B_i \cdot (I + C_i^T \cdot A_i^{-1} \cdot B_i)^{-1} \quad (7)$$

$$A_{i-k+1}^{-1} \cdot B_i = A_{i-k}^{-1} \cdot B_i - M_{i-k} \cdot C_{i-k}^T \cdot A_{i-k}^{-1} \cdot B_i, \quad k = 1, \dots, i, \quad (8)$$

Now assume that we have already pre-calculated the inverse  $A^{-1}$  of the global stiffness matrix  $A$ . To calculate the displacement vectors  $D_{i+1}$  in (5), we need to compute its two main components  $A_i^{-1}F_i$  and  $M_i$ . We know that  $F_i$  contains only a few non-zero entries, so  $A_i^{-1}F_i$  involves  $O(n)$  operations. For  $M_i$ , we need to compute two terms  $A_i^{-1}B_i$  and  $(I - C_i^T A_i^{-1} \cdot B_i)^{-1}$ . Since  $B_i$  and  $C_i$  are sparse, and since  $(I - C_i^T A_i^{-1} \cdot B_i)^{-1}$  is only a 12x12 matrix, updating each  $M_i$  also requires in the order of  $O(n)$  operations. The following diagram shows the updating scheme for each cutting operation:

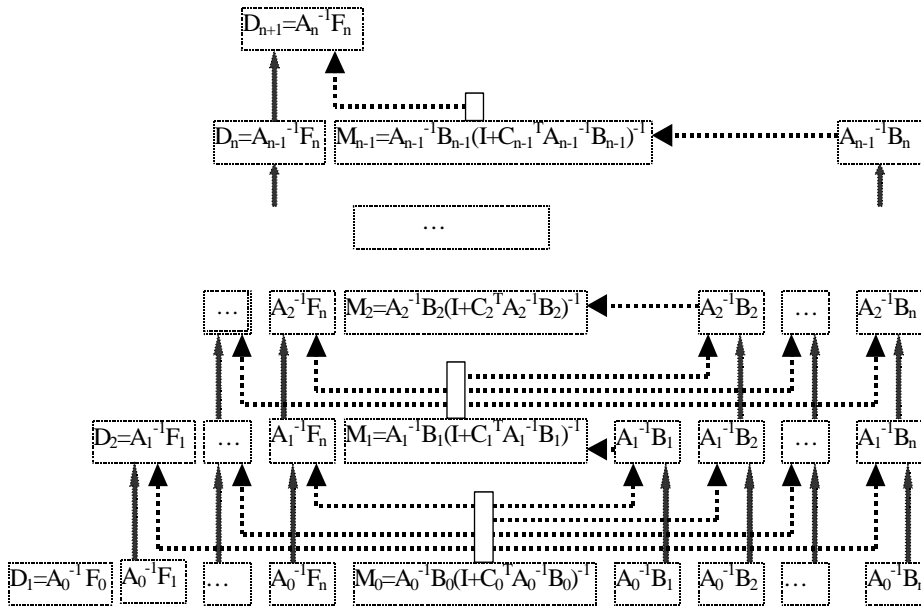


Diagram 1. Flowchart for computing the  $n$ -th cutting deformation with the basic updating procedures.

From the diagram above we see that the computation of  $A_n^{-1}F_n$  involves the computation of all  $A_{n-i}^{-1} \cdot F_n$ ,  $A_{n-i}^{-1} \cdot B_n$  as well as  $M_{n-i}$  for  $i=1, \dots, n$ . Since  $F_n, B_n$  are related only to the  $n$ -th cutting step, the first two terms must be calculated at each cutting simulation. However except for  $M_{n-1}$ , all other  $M_{n-i}$  ( $i=2, \dots, n$ ) have already appeared in the previous cutting calculation, so they are available for the next step.

## 2.4. Implementation of the Algorithm

Based on the above discussion, we perform the following steps to realize the  $i$ -th updating procedure:

$$T_0^i = A_0^{-1} \cdot F_i, \quad T_k^i = T_{k-1}^i - M_{k-1} \cdot C_{k-1}^T \cdot T_{k-1}^i, \quad k = 1, \dots, i \quad (9)$$

$$S_0^i = A_0^{-1} \cdot B_i, \quad S_k^i = S_{k-1}^i - M_{k-1} \cdot C_{k-1}^T \cdot S_{k-1}^i, \quad k = 1, \dots, i \quad (10)$$

$$D_{i+1} = T_i^i, \quad M_i = S_i^i \cdot (I + C_i^T \cdot S_i^i)^{-1} \quad (11)$$

In the formulae (9), (10) and (11), we implement the multiplications of  $A^{-1} \cdot F$ ,  $A^{-1} \cdot B$  and  $C^T \cdot S$ , based only on non-zero entries in  $F$ ,  $B$ ,  $C$ , and the inverse of the  $12 \times 12$  matrix  $(I - C_i^T A_i^{-1} \cdot B_i)$  is computed through LU decomposition. From the above formulation we see that each cutting step not only involves the current force  $F_i$  and contacted node information  $C_i$ , but it also relates to the previous computed fundamental components  $M_i$  and  $C_i^T$ , so they must be saved to lists for the later updating step. While this may increase the requirements for more computer memory, compared to the  $n^2$  entries of  $K^{-1}$ , saving all the  $12n$  entries of  $M_i$  is not a significant burden. With the output  $D_{i+1}$  we need to update the displacements of all the nodes instead of just those at the surface, since we don't know which node might be on the surface in the later cutting step.

This project has been developed in C++ with VTK's polydata structure for visualization while its GUI was implemented with Qt. The effect of this cutting simulation is visualized below through a simulated liver section that is meshed into a data set consisting of 1097 nodes and 4780 tetrahedra (see Figure 1 and 2).

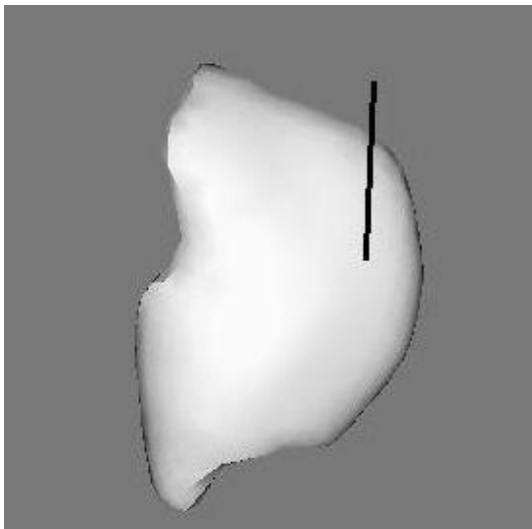


Figure 1: A liver model with 1097 nodes and 4780 tetrahedra

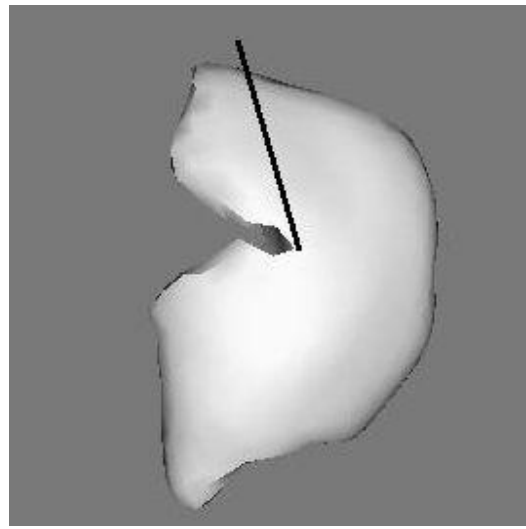


Figure 2: Cutting results using our adaptive algorithm

### 3. NUMERIC ANALYSIS AND RESULTS

Before evaluating the proposed algorithm, we first describe how much computational expense the localized conjugate gradient method [8] requires. The pseudo code of the CG method [10] can be expressed as

1. for  $i=1, 2, \dots$
2.     solve  $Mz^{(i-1)}=r^{(i-1)}$
3.      $\rho_{i-1}=(r^{(i-1)})^T z^{(i-1)}$
4.     if  $i=1$   $p^{(1)}=z^{(0)}$
5.     else  $\beta_{i-1}=\rho_{i-1}/\rho_{i-2}$ ;
6.      $p^{(i)}=z^{(i-1)}+\beta_{i-1}p^{(i-1)}$

7. endif
8.  $\mathbf{q}^{(i)} = \mathbf{A}\mathbf{p}^{(i)}$
9.  $\alpha_i = \mathbf{p}_{i-1} / (\mathbf{p}^{(i)})^T \mathbf{q}^{(i)}$
10.  $\mathbf{x}^{(i)} = \mathbf{x}^{(i-1)} + \alpha_i \mathbf{p}^{(i)}$
11.  $\mathbf{r}^{(i)} = \mathbf{r}^{(i-1)} - \alpha_i \mathbf{q}^{(i)}$
12. evaluate  $\mathbf{r}^{(i)}$  to check the convergence
13. end

From the above specification on the CG method, it is easy to see that the dominant computation time in each loop is on the line 8 which generally requires  $n^2$  multiplications. However, since the matrix  $\mathbf{A}$  is sparse and on each row or column only those entries which represent adjacent nodes are non-zero, if a node  $p_i$  has  $n_i$  neighbor nodes, the number of all the multiplications in the line 8 actually is the upper bound  $3n \times \max(n_i)$ . As claimed in [8], a uniform mesh is important to reduce the computation time because it may minimize  $\max(n_i)$ . On average,  $n_i$  is about 20. So the total number of multiplications is about  $60n\nu$  where the convergent loop number  $\nu$  is about 100 for a tetrahedral mesh with 2886 free nodes.

From section 2.3, we see that it is possible to directly compute the inverse of the updated matrix by using the Woodbury formula, which requires  $O(n^2)$  operations and gives a speed of about 1 frame/sec in our implementation for a smaller size of mesh (1097 nodes). Since the speed of the CG method depends on the iterate number as well as the number of neighbors each node may have, the efficiency of directly using the Woodbury formula could be better than or comparable to the CG method for a system of less than 500 degrees of freedom. For a large system, however, we need to update each component in the Woodbury formula instead of updating the global inverse, to avoid the  $O(n^2)$  operations.

Let us examine formulae (9), (10) and (11) in section 2.4 for the cost of the  $i$ -th cutting simulation. Since  $B_i$  only has 12 non-zero entries for a tetrahedron element, and  $F_i$  has up to 9 non-zero entries when only one face of a tetrahedron is intersected at a time, computing  $S_0^i$  and  $T_0^i$  needs a total of  $21n$  multiplications.  $C_i^T$  has 144 non-zero entries, and therefore  $C_{i-k}^T \cdot T_{i-k}^i$  and  $C_{i-k}^T \cdot S_{i-k}^i$  need  $12 \times 12$  and  $144 \times 12$  multiplications, respectively. The dominant components in the iteration are  $M_{i-k} \cdot C_{i-k}^T \cdot S_{i-k}^i$  and  $M_{i-k} \cdot C_{i-k}^T \cdot T_{i-k}^i$  which require totally  $156n$  operations. To compute  $M_i$ , inverting the  $12 \times 12$  matrix may require  $12^3$  multiplications while another  $144n$  are needed when multiplying the inverse with  $S_i$ . The total number of multiplications for computing the  $i$ -th operation is  $(C_1 + 165n) + (C_2 + 156n)i$ . The two constants  $C_1$ ,  $C_2$  are independent of the node number  $n$ , but are related to the element type. For tetrahedra their values are  $\sim 2000$ .

To evaluate the efficiency of this algorithm, we use an MRI image of a brain phantom to generate a larger mesh consisting of 11741 tetrahedra and 3921 nodes among which 1035 nodes in the base of the phantom are fixed. During the incision test we delete one tetrahedron each time and a total of 100 tetrahedra are removed at the end of the incision. The simulation requires 3.2G bytes of memory, and its computing speed, as a function of number of tetrahedrons removed, on a 1 GHz PC is illustrated in Figure 3. Figure 4 shows the results of a simulated incision experiment.

#### 4. DISCUSSION

Finite element approaches are limited in simulating soft tissue incisions. For example, CG methods barely achieve real-time performance, and the previous PFE methods are not applicable to incision simulation. In contrast, the adaptive algorithm outlined in this paper can be applied in such situations with a satisfactory computational efficiency.

Compared to the localized CG method described in [8], our method demonstrates a speed increase of up to 5 times in the first 12 or so incision operations, and unlike the iterative CG method, its updating computation can be completed in a predictable time. The advantages of this approach also include its lack of run-time convergence requirements and the fact that all the subsequent cutting steps are based on the original inverse matrix. Therefore there is no need to regenerate the mesh or reassemble the global stiffness matrix, which may not only be more costly computational, but could also reduce system stabilities. In practice, we only need to update the rendering structure to remove the corresponding tetrahedron.

Our method is based on a linear elastic model, so it is not suitable to characterize a large deformation or rotation. In practice, we tested our method on two mesh data sets, one a liver model fixed at six nodes, and the other a brain phantom fixed on its entire base. During a surgical incision animation, the brain has no obvious rotation, but the liver has larger deformation and rotation. For this reason, the effect of linear elastic model may not be satisfactory, so it will be worthwhile in the future to study nonlinear elastic models with real time cutting algorithms.

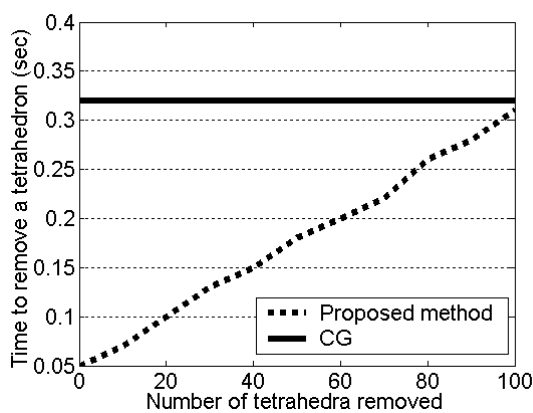


Figure 3: Incision simulation speed on a mesh with 2886 free nodes.

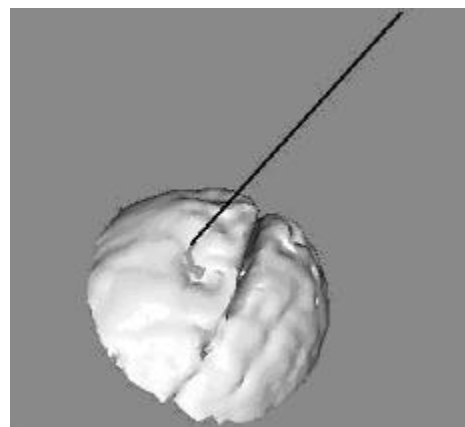


Figure 4: Incision simulation with the proposed algorithm on a mesh of a brain phantom.

## 5. CONCLUSION

Precomputed finite element methods are generally considered not to be suitable for surgical cutting simulation due to the fact that they are restricted by the mesh, boundary constraints and contact locations predefined in a preprocessing stage. In this paper we described an adaptive pre-computed algorithm for real time finite element computation, that differs from the previous PFE methods in that it records updated components so that all the previous operations may be tracked. For the brain and liver examples shown here, this approach performs with a satisfactory speed (up to 15 and 35 updates per second, respectively) on a 1 GHz PC. Compared to a conjugate gradient method, this algorithm is faster, independent of the convergence behavior and mesh quality, and thus more stable in a real time simulation.

Like all the pre-computed FE models, this approach needs large memory and storage for the pre-computed inverse stiffness matrix, as well as for all the basic components at each cutting. Also this approach is essentially based on the input force, and it is not easy to specify the exact cutting depth with a given force. The simulation of soft tissue incision still is a new area of research and it involves many factors such as friction, mass loss, cutting models or fracture parameters. We hope to further address these topics in the near future.

## ACKNOWLEDGEMENTS

The authors gratefully acknowledge the financial support provided by the Canadian Institutes for Health Research MOP 14735, the Ontario Research and Development Challenge Fund, OCITS, and SHARCNet.

### Bibliography

- [1] Gibson, S. F. F. 3D chainmail: a fast algorithm for deforming volumetric objects, In 1997 Symposium on Interactive 3D Graphics, 149-154, Providence, Rhode Island, 1997
- [2] Miller, G. The motion dynamics of snake and worms, Computer Graphics (SIGGRAPH 88 Conference Proceedings), 169-173, In Dill, J. editor, Atlanta, Georgia, 1988
- [3] Meseure, P. and Chaillou, C. Deformable Body Simulation with Adaptive Subdivision and Cuttings, Proc. WSCG'97, 361-370, In Thalmann, N. M. Skala V. (Eds.), Campus Bory, Plzen - Bory, Czech Republic, 1997
- [4] Bro-Nielsen, M., "Finite element modeling in medical VR", *Proc.IEEE*, vol. 86, no. 3, pp. 490-503, 1998.
- [5] James, D. L. and Pai, D. K. Accurate real time deformable objects, Computer Graphics (SIGGRAPH 99 Conference Proceedings), 65-72, Los Angeles, California, 1999
- [6] Hager, W. W., "Updating the Inverse of a Matrix", *SIAM Review*, vol. 31 pp. 221-239, 1989.
- [7] Cotin, S., Delingette, H., and Ayache, N., "Real-time elastic deformations of soft tissues for surgery simulation", *IEEE Trans.Visua.Compu.Graph*, vol. 5 pp. 62-73, 1999.
- [8] Nienhuys, H. W. and van der Stappen, A. F. Combining finite element deformation with cutting for surgery simulation, Proc.Eurographics 2000, 143-152, In Sousa, A. de Torres J. C (Eds.), 2000.
- [9] Nienhuys, H. W. and van der Stappen, A. F. A surgery simulation supporting cuts and finite element deformation, MICCAI2001, LNCS 2208 , 145-152, In Niessen, W. Viergever M. (Eds.), Springer-Verlag, 2001
- [10] Barrett, R., Berry, M., Chan, T., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., and van der Vorst, H., *Templates for the solution of linear systems: building blocks for iterative methods*, 2 ed. Philadelphia, PA: SIAM, 1994.
- [11] Berkley, J., Weghorst, S., Gladstone, H., Raugi, G., Berg, D., and Ganter, M., "Banded matrix approach to finite element modeling for soft tissue simulation", *Virtual Reality*, vol. 4 pp. 203-212, 1999.
- [12] Easley, J. G., *Mechanics of elastic structures* Prentice-Hall, 1989.
- [13] Zienkiewicz, O. C. and Taylor, R. L., *The finite element method, Volume 1: The Basis* Barcelona, Spain: Butterworth Heinemann, 2000.
- [14] Cotin, S., Delingette, H., and Ayache, N., "A Hybrid Elastic Model allowing Real-Time Cutting, Deformations and Force-Feedback for Surgery Training and Simulation", *Visua Compu*, vol. 16 pp. 437-452, 2000.