Real-time Volumetric Deformable Models for Surgery Simulation using Finite Elements and Condensation

Morten Bro-Nielsen[1] and Stephane Cotin[2]

[1] Dept. of Mathematical Modelling
Technical University of Denmark, Bldg. 321
DK-2800, Lyngby, Denmark
e-mail: bro@imm.dtu.dk WWW http://www.imm.dtu.dk/~bro
[2] INRIA, Epidaure project, Sophia Antipolis, France

Abstract

This paper discusses the application of 3D solid volumetric Finite Element models to surgery simulation. In particular it presents three new approaches to the problem of achieving real-time performance for these models. The simulation system we have developed is described and we demonstrate real-time deformation using the methods developed in the paper.

Keywords: Virtual Surgery, Real-Time Deformation, Solid Volumetric Deformable Models, Virtual Reality, Finite Element Models.

1 Introduction

Speed is overriding concern in Surgery Simulation and it is only in the last few years that real-time surgery simulation has become practically possible.

The big problem in surgery simulation is modeling the deformation of solid volumetric objects, which often can have very complex forms, in real-time, ie. 15-20 frames/second. Since human organs and tissue have very complex elastic behaviour it has only been possible to model these using very simplistic models.

Almost all the attempts have used *surface* models as the basic modeling method [6] and realistic tissue models have only been applied in very specialized simulators such as the eye-surgery simulator of Sagar et al. [12].

The problem with surface models, besides the obvious non-solid behaviour, is the lack of an defined interior when surgical procedures are modeled. The surgeon cannot cut a virtual organ modeled using a surface model since there is nothing inside the surface. Some simple cuts can be modeled, such as cutting an artery or other thin structures, but general surgical incisions are impossible.

Attention is therefore turning towards solid *volumetric* models which model the complete 3D volumetric behaviour of the object [3, 5, 8] Unfortunately the complexity of the models rises dramaticly when volumetric models are used and real-time performance is difficult to achieve.

Another issue is the way elastic behaviour is simulated. Simulation can be performed using global parametric models [6, 16] or local mesh based models [3, 5, 8, 14, 15] (eg. Finite Element models). We believe the need to accommodate surgical incisions demand mesh based models. Only simplistic cuts can be achieved with global parametric models.

We would like to emphasize that we believe all the models mentioned above will find some use in surgery simulation, even in the same application.

This paper discusses real-time simulation of deformable objects using 3D solid volumetric Finite Element (FE) models which result in linear matrix systems. We present the theory behind 3D FE models of linear elasticity and then discuss various aspects of these models. In particular we present three new improvements over previous applications of FE.

The first improvement is called *Condensation* and in practice it allows us to compress the linear matrix system resulting from the *volumetric* FE model to a system with the same complexity as a FE *surface* model of the same object. Although it has the same complexity as the surface model it still models the volumetric behaviour of the object.

The second improvement concerns the way the linear matrix system is used for simulation. In contrast to the normal approach and against the advice of finite element people we explicitly invert the system matrix and use matrix vector multiplication with this matrix to achieve a very low calculation time.

The third improvement is what we call Selective Matrix Vector Multiplication where we exploit the sparse structure of the force vector.

In addition we describe a simulation system we have developed for surgery simulation. With this system we are able to simulate solid volumetric deformation of relatively large objects with video frame-rates.

2 Theory

In this section we describe the model which we use to simulate elastic deformation of a volumetric solid in real-time. To develop the model we formulate a number of requirements that the model should fulfil:

1. Speed is everything. Deformation should be calculated in the smallest amount of time possible.
2. We do not care about the time taken for one-time pre-calculation such as setting up equations, inverting matrices etc. If something takes 24 hours extra in the pre-calculation stage, but will save 0.01 second in the simulation stage, we should do it.
3. The elastic model should be *visually* convincing. The model may be physically incorrect if it looks right.
4. In the long run we want to be able to make cuts in the model to accommodate surgical procedures. This involves changing the topology of the model and most importantly requires models that have defined interiors, ie. volumetric models.

In particular the last requirement lead us to select mesh-based 3D Finite Element (FE) models. The alternative would be parametric models such as [6]. But these models does not provide the needed freedom to perform topology changes to allow cutting. Although some of the models can handle simple cuts we aim towards being able to make completely general cuts in the models. We are convinced that only mesh-based models will allow this.

To meet the first two requirements we choose the linear elastic deformation model which is also known as Hooke's law. Using linear elasticity as the basic model involves a number of assumptions regarding the physical material that is modeled. Most importantly linear elastic models are only valid for very small deformations and strains. They are typically correct for such rigid structures as metal beams, buildings etc. Although they are used extensively in modeling, the visual result of large deformation modeling using linear elasticity is seldom satisfying.

But when used with FE these models lead to linear matrix systems which are easy to solve and fast. There is, therefore, a trade-of between the speed of the system and the visual deformation result.

Linear elastic models are used here because modeling general elastic volumetric deformation using FE is only *just* possible with todays computers. With faster computers in the future we expect more realistic models, such as incompressible Mooney-Rivlin material models [4], to be used.

2.1 Linear elastic material model

We define the elastic solid Ω as the positions $\boldsymbol{x} = [x\ y\ z]^T$ where $\boldsymbol{x} \in \Omega$ (see figure 1). The displacement of particle $\boldsymbol{x}$ is defined as $\boldsymbol{u}(\boldsymbol{x}) = [u(\boldsymbol{x})\ v(\boldsymbol{x})\ w(\boldsymbol{x})]^T$ so that the particle $\boldsymbol{x}$ is moved by the deformation to

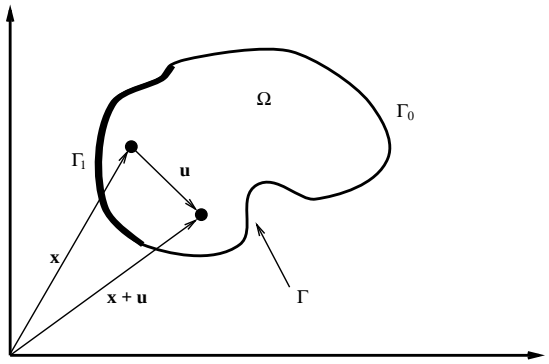

Figure 1: Solid elastic object.

$\boldsymbol{x} + \boldsymbol{u}$. The boundary $\Gamma = \Gamma_0 \cup \Gamma_1$, $\Gamma_0 \cap \Gamma_1 = \emptyset$ where Γ_0 has fixed displacements $\boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{u}_0(\boldsymbol{x})$ and forces $\boldsymbol{f}(\boldsymbol{x})$ are applied to Γ_1.

The *strain* energy of a linear elastic solid Ω is defined as [10]:

$$\mathrm{E}(\boldsymbol{u}) = \frac{1}{2} \int \int \int_{\Omega} \boldsymbol{\epsilon}^T \boldsymbol{\sigma} \, d\boldsymbol{x} \tag{1}$$

where the strain vector $\boldsymbol{\epsilon} = [\epsilon_x \ \epsilon_y \ \epsilon_z \ \gamma_{xy} \ \gamma_{xz} \ \gamma_{yz}]^T$ consists of:

$$\begin{aligned} \epsilon_x &= \frac{\delta u}{\delta x} \qquad \epsilon_y = \frac{\delta u}{\delta y} \qquad \epsilon_z = \frac{\delta u}{\delta z} \\ \gamma_{xy} &= \frac{\delta u}{\delta y} + \frac{\delta v}{\delta x} \qquad \gamma_{xz} = \frac{\delta u}{\delta z} + \frac{\delta w}{\delta x} \qquad \gamma_{yz} = \frac{\delta v}{\delta z} + \frac{\delta w}{\delta y} \end{aligned} \tag{2}$$

We can rewrite this as $\boldsymbol{\epsilon} = \boldsymbol{B}\boldsymbol{u}$ where

$$\boldsymbol{B} = \begin{bmatrix} \frac{\delta}{\delta x} & 0 & 0 \\ 0 & \frac{\delta}{\delta y} & 0 \\ 0 & 0 & \frac{\delta}{\delta z} \\ \frac{\delta}{\delta y} & \frac{\delta}{\delta x} & 0 \\ \frac{\delta}{\delta z} & 0 & \frac{\delta}{\delta x} \\ 0 & \frac{\delta}{\delta z} & \frac{\delta}{\delta y} \end{bmatrix} \tag{3}$$

The *stress* vector $\boldsymbol{\sigma}$ is related to the strain vector through Hooke's law by $\boldsymbol{\sigma} = \boldsymbol{D}\boldsymbol{\epsilon}$, where $\boldsymbol{D}$ is a symmetric 6×6 *material stiffness matrix*. For a *homogenous and isotropic* material this matrix is defined by the two lamé material constants λ and μ:

$$\boldsymbol{D} = \begin{bmatrix} \lambda + 2\mu & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda + 2\mu & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & \lambda + 2\mu & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{bmatrix} \tag{4}$$

Using these relations we can now rewrite the strain energy and add work done by external forces $\boldsymbol{f}$ to yield the potential energy function:

$$\mathrm{E}(\boldsymbol{u}) = \frac{1}{2} \int \int \int_{\Omega} \boldsymbol{u}^T \boldsymbol{B}^T \boldsymbol{D} \boldsymbol{B} \boldsymbol{u} \, d\boldsymbol{x} - \int \int_{\Gamma_1} \boldsymbol{f}^T \boldsymbol{u} \, d\boldsymbol{a} \tag{5}$$

where Γ_1 is the part of the surface $\Gamma = \Gamma_0 \cup \Gamma_1$ where external forces are applied. Fixed displacements $\boldsymbol{u}(x, y, z) = \boldsymbol{u}^0(x, y, z)$ are applied to Γ_0.

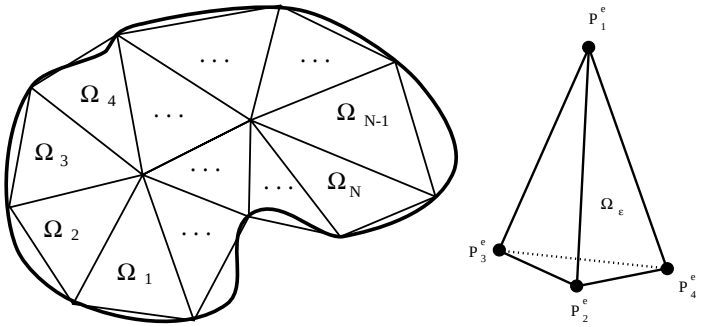

Figure 2: Left: Discretization of the domain into finite elements (2D illustration) Right: Tetrahedral finite element.

2.2 Discretization using FE model

We assume that the domain Ω of the volumetric solid has been discretized into a number of finite elements Ω^e in the form of tetrahedrons and nodes P_q defined by $\boldsymbol{x}_q = [x_q \; y_q \; z_q]^T$ (see figure 2). The deformation at each node is specified by the deformation vector $\boldsymbol{u}_q = [u_q \; v_q \; w_q]^T$. In addition we also stack these vectors into two compound vectors:

$$\underline{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{x}_1^T & \boldsymbol{x}_2^T & \dots & \boldsymbol{x}_n^T \end{bmatrix}^T \qquad \underline{\boldsymbol{u}} = \begin{bmatrix} \boldsymbol{u}_1^T & \boldsymbol{u}_2^T & \dots & \boldsymbol{u}_n^T \end{bmatrix}^T \tag{6}$$

The nodes of each finite element Ω^e are denoted P_i^e, where i is the local number of the node which is unrelated to the global numbering of the nodes.

As finite elements we use four-node tetrahedrons with linear interpolation of the displacement field between the nodes:

$$\boldsymbol{u}(\boldsymbol{x}) = \sum_{i=1}^{4} N_i^e(\boldsymbol{x}) u_i^e \tag{7}$$

The basis functions $N_i^e(\boldsymbol{x})$ are defined as the natural coordinates L_i of the tetrahedron:

$$N_i^e(\boldsymbol{x}) = L_i = \frac{1}{6V^e}(a_i^e + b_i^e x + c_i^e y + d_i^e z), i = 1,2,3,4 \tag{8}$$

The natural coordinates, the volume V^e and the coefficients $a_i^e, b_i^e, c_i^e, d_i^e$ are defined in appendix A.

We find the solution to the deformation problem when the potential energy of the system assumes its minimum value. This happens when the first variation of the functional E vanishes, ie. when $\delta \mathrm{E}(\boldsymbol{u}) = 0$.

Using the fact that:

$$\frac{\delta \boldsymbol{u}}{\delta x} = \sum_{i=1}^{4} \frac{\delta N_i^e(\boldsymbol{x})}{\delta x} \boldsymbol{u}_i^e \qquad \frac{\delta \boldsymbol{u}}{\delta y} = \sum_{i=1}^{4} \frac{\delta N_i^e(\boldsymbol{x})}{\delta y} \boldsymbol{u}_i^e \qquad \frac{\delta \boldsymbol{u}}{\delta z} = \sum_{i=1}^{4} \frac{\delta N_i^e(\boldsymbol{x})}{\delta z} \boldsymbol{u}_i^e \tag{9}$$

where

$$\delta N_i^e(\boldsymbol{x})/\delta x = b_i^e \qquad \delta N_i^e(\boldsymbol{x})/\delta y = c_i^e \qquad \delta N_i^e(\boldsymbol{x})/\delta z = d_i^e \tag{10}$$

we can rewrite the equilibrium equation for each element as

$$\boldsymbol{0} = \int\int\int_{\Omega^e} \boldsymbol{B}^{eT} \boldsymbol{D} \boldsymbol{B}^e \underline{\boldsymbol{u}}^e d\boldsymbol{x} - \underline{\boldsymbol{f}}^e \tag{11}$$

where $\underline{\boldsymbol{f}}^e$ is a discretized force vector for the element and $\boldsymbol{B}^e$ is a constant matrix given in appendix A [1].

[1] $\underline{\boldsymbol{u}}^e$ is 12×1, $\underline{\boldsymbol{f}}^e$ is 12×1 and $\boldsymbol{B}^e$ is 6×12

Because everything inside the integration sign is constant the equilibrium equation for the finite element becomes a linear matrix equation $\boldsymbol{K}^e \underline{\boldsymbol{u}}^e = \underline{\boldsymbol{f}}^e$, where $\boldsymbol{K}^e = \boldsymbol{B}^{eT} \boldsymbol{D} \boldsymbol{B}^e V^e$ is called the *stiffness* matrix [2] and V^e is the volume of the tetrahedron (see appendix A).

The only remaining step is assembly of the global stiffness matrix from the element stiffness matrices:

$$\boldsymbol{K} = \sum_e t(\boldsymbol{K}^e) \tag{12}$$

where $t()$ is a transfer function from element node numbers to global node numbers. The result is a large sparse linear system $\boldsymbol{K}\underline{\boldsymbol{u}} = \underline{\boldsymbol{f}}$.

2.3 Condensation

The linear matrix system $\boldsymbol{K}\underline{\boldsymbol{u}} = \underline{\boldsymbol{f}}$ models the behaviour of the *solid* object. This includes both surface nodes as well as the internal nodes of the model. But for simulation purposes we are usually only interested in the behaviour of the surface nodes since these are the only *visible* nodes. We, therefore, use condensation [11] to remove the internal nodes from the matrix equation.

The matrix equation for the condensed problem has the same size as would result from a FE *surface* model. But, it is important to understand that it will show *exactly* the same behaviour for the surface nodes as the original *solid volumetric* system.

Without loss of generality, let us assume that the nodes of the FE model have been ordered with the surface nodes first, followed by the internal nodes. Using this ordering we can rewrite the linear system as a block matrix system (*s*urface / *i*nternal):

$$\begin{bmatrix} \boldsymbol{K}_{ss} & \boldsymbol{K}_{si} \\ \boldsymbol{K}_{is} & \boldsymbol{K}_{ii} \end{bmatrix} \begin{bmatrix} \underline{\boldsymbol{u}}_s \\ \underline{\boldsymbol{u}}_i \end{bmatrix} = \begin{bmatrix} \underline{\boldsymbol{f}}_s \\ \underline{\boldsymbol{f}}_i \end{bmatrix} \tag{13}$$

From this block matrix system we can create a new linear matrix system $\boldsymbol{K}^*_{ss}\underline{\boldsymbol{u}}_s = \underline{\boldsymbol{f}}^*_s$ which only involves the variables of the *surface* nodes:

$$\boldsymbol{K}^*_{ss} = \boldsymbol{K}_{ss} - \boldsymbol{K}_{si}\boldsymbol{K}^{-1}_{ii}\boldsymbol{K}_{is} \qquad \underline{\boldsymbol{f}}^*_s = \underline{\boldsymbol{f}}_s - \boldsymbol{K}_{si}\boldsymbol{K}^{-1}_{ii}\underline{\boldsymbol{f}}_i \tag{14}$$

The displacement of the internal nodes can still be calculated using $\underline{\boldsymbol{u}}_i = \boldsymbol{K}^{-1}_{ii}(\underline{\boldsymbol{f}}_i - \boldsymbol{K}_{is}\underline{\boldsymbol{u}}_s)$ Notice, that if no forces are applied to internal nodes, $\underline{\boldsymbol{f}}^*_s = \underline{\boldsymbol{f}}_s$.

Generally the new stiffness matrix will be dense compared to the sparse structure of the original system. But, since we intend to solve the system by inverting the stiffness matrix in the pre-calculation stage, this is not important.

2.4 Simulation

In this section we will discuss the different simulation methods available to us. We have two linear matrix equations. One with all the nodes of the FE model and a sparse stiffness matrix. And a reduced version with only the surface nodes and a dense stiffness matrix.

2.4.1 Solving linear matrix system

Implicit solution of a linear system is often performed using iterated algorithms such as the Conjugate Gradient (CG) algorithm [1]. In principle this algorithm performs a sparse matrix vector multiplication, three vector updates and two inner products in an iterative loop. The complexity can therefore roughly be seen as $n \times t_{Mv+3v+2vv}$ where $t_{Mv+3v+2vv}$ is the time required for the operations in one iteration and n is the

[2] $\boldsymbol{K}^e$ is 12×12

number of iterations. n is seldom less than 5-10 and cannot in practice be predicted. Especially for a system where the response must come at specific frame rates, unpredictable solution time is very unfortunate.

The alternative that we use here is explicitly inverting the stiffness matrix. Normally this is never done when linear systems resulting from FE models are solved. The precision of the result suffers from numerical errors and the amount of storage needed to store a dense inverted stiffness matrix is huge compared to the sparse stiffness matrix itself. But, as we stated in the beginning, we are not concerned about precision or memory size, only speed.

Although the time for inversion is considerable, the solution time is very small since it only involves a dense matrix vector multiplication ($\underline{\boldsymbol{u}} = \boldsymbol{K}^{-1}\underline{\boldsymbol{f}}$). We have performed numerical tests using the Meschach library [13] to solve a linear system generated by a FE model. These experiments included both explicit inversion, CG with and without preconditioner, Gauss elimination and several factorization techniques such as QR and Cholesky. When the pre-calculation time was ignored, solution by matrix vector multiplication with the inverted stiffness matrix was at least 10 times faster than any other method. We have not stated the actual numerical results here, since the implementations of the different algorithms in the Mescach library have not been properly optimized. The timings could therefore be different for other implementations although the general result would be the same.

2.4.2 Dynamic system

In order to use a physically correct model of the solid we add mass and damping to the model. We do this by formulating the Lagrangian equation of motion for the object:

$$\boldsymbol{M}\ddot{\underline{\boldsymbol{u}}} + \boldsymbol{C}\dot{\underline{\boldsymbol{u}}} + \boldsymbol{K}\underline{\boldsymbol{u}} = \underline{\boldsymbol{f}} \tag{15}$$

where $\boldsymbol{M}$ is the mass, $\boldsymbol{C}$ is the damping and $\boldsymbol{K}$ is the stiffness matrix of the system. $\boldsymbol{K}$ is calculated as shown above. Assuming lumped masses at the nodes we use diagonal damping and mass matrices:

$$\boldsymbol{M}^e_{ii} = \frac{1}{3}\rho V^e \qquad \boldsymbol{C}^e_{ii} = \alpha \boldsymbol{M}^e_{ii} \tag{16}$$

where ρ is the mass-density and α is a scaling factor.

The global element matrices are assembled into global matrices. Since the mass and damping matrices are diagonal they are also block diagonal. The Lagrangian equation of motion for the reduced system, therefore, simply becomes:

$$\boldsymbol{M}_{ss}\ddot{\underline{\boldsymbol{u}}}_s + \boldsymbol{C}_{ss}\dot{\underline{\boldsymbol{u}}}_s + \boldsymbol{K}^*_{ss}\underline{\boldsymbol{u}}_s = \underline{\boldsymbol{f}}^*_{ss} \tag{17}$$

Using finite difference estimates in time we can write the Lagrangian equation for the full system as:

$$\frac{\boldsymbol{M}}{\Delta t^2}(\underline{\boldsymbol{u}}_{t+\Delta t} - 2\underline{\boldsymbol{u}}_t + \underline{\boldsymbol{u}}_{t-\Delta t}) + \frac{\boldsymbol{C}}{2\Delta t}(\underline{\boldsymbol{u}}_{t+\Delta t} - \underline{\boldsymbol{u}}_{t-\Delta t}) + \boldsymbol{K}\underline{\boldsymbol{u}}_{t+\Delta t} = \underline{\boldsymbol{f}}_{t+\Delta t} \tag{18}$$

or $\tilde{\boldsymbol{K}}\underline{\boldsymbol{u}}_{t+\Delta t} = \underline{\tilde{\boldsymbol{f}}}_{t+\Delta t}$ where:

$$\begin{aligned} \tilde{\boldsymbol{K}} &= \frac{\boldsymbol{M}}{\Delta t^2} + \frac{\boldsymbol{C}}{2\Delta t} + \boldsymbol{K} \\ \underline{\tilde{\boldsymbol{f}}}_{t+\Delta t} &= \frac{\boldsymbol{M}}{\Delta t^2}\underline{\boldsymbol{u}}_t - (\frac{\boldsymbol{M}}{\Delta t^2} - \frac{\boldsymbol{C}}{2\Delta t})\underline{\boldsymbol{u}}_{t-\Delta t} + \underline{\boldsymbol{f}}_{t+\Delta t} \end{aligned} \tag{19}$$

We leave it to the reader to formulate the same equations for the condensed system.

2.4.3 Static system and Selective Matrix Vector Multiplication

Generally the $\underline{\tilde{\boldsymbol{f}}}_{t+\Delta t}$ vector is a full vector because of the contribution from the previous displacement vectors $\underline{\boldsymbol{u}}_t$ and $\underline{\boldsymbol{u}}_{t-\Delta t}$. In contrast the original force vector $\underline{\boldsymbol{f}}_{t+\Delta t}$ can be a sparse vector in cases where

Figure 3: Voxel data from the visible human data set.

forces only are applied to a few surface nodes. Since this is often the case in simulation we were inspired to develop an alternative simulation method which for sparse force vectors is considerably faster.

The cost of this simulation method is the loss of dynamics. The idea is to use the original static linear system $\boldsymbol{K}\underline{\boldsymbol{u}} = \underline{\boldsymbol{f}}$ (or the condensed version) instead of the Lagrangian dynamic system above and exploit the sparse structure of the force vector.

Formally, solving the system using the inverted stiffness matrix is performed using $\underline{\boldsymbol{u}} = \boldsymbol{K}^{-1}\underline{\boldsymbol{f}}$. If only a few positions of the force vector are non-zero, clearly standard matrix vector multiplication would involve a large number of superfluous multiplications. We note that

$$\underline{\boldsymbol{u}} = \boldsymbol{K}^{-1}\underline{\boldsymbol{f}} = \sum_i \boldsymbol{K}^{-1}_{*i}\underline{\boldsymbol{f}}_i \tag{20}$$

where $\boldsymbol{K}^{-1}_{*i}$ is the i'th column vector of $\boldsymbol{K}^{-1}$ and $\underline{\boldsymbol{f}}_i$ the i'th element of $\underline{\boldsymbol{f}}$. Since the majority of the $\underline{\boldsymbol{f}}_i$ are zero, we restrict i to run through only the positions of $\underline{\boldsymbol{f}}$ for which $\underline{\boldsymbol{f}}_i \neq 0$. If n of the N positions in $\underline{\boldsymbol{f}}$ are non-zero this will reduce the complexity to $o(n/N)$ times the time of a normal matrix vector multiplication. We call this approach *Selective Matrix Vector Multiplication* (SMVM).

The SMVM method and the dynamic model converge for slow changes in forces applied to the solid body.

3 Simulation system

We first summarize the theory of the last section and then present the simulation system that has been developed for surgery simulation. In addition we also describe how we generate the FE mesh model of the physical organ, limb etc.

3.1 Summary of simulation methods

In the previous theory section we have presented a number of ideas related to simulation using FE models. In general two criterias separate the possible algorithms: Full FE model contra condensed FE model, and dynamic simulation contra selective matrix vector multiplication (SMVM).

	Dynamic simulation	SMVM
Full FE model	FD	FS
Condensed FE model	CD	CS

The choice of algorithm depends on the requirements of the application. Whether one chooses dynamic simulation or SMVM is a question of the speed requirement and size of the problem. If the problem can be processed fast enough using dynamic simulation it would be the best choice.

The choice between the full FE model versus the condensed model is based on whether it is necessary to modify the FE model during simulation or not, eg. to model a surgical cut in the model. If modification is necessary, the standard full system would probably be easier to modify, rather than the condensed system which is one step further in refinement. Without the need for modification the condensed model should clearly be used.

We have implemented the CS, CD and FS methods in our simulation system.

3.2 Mesh generation using Nuages and Mvox

In addition to a range of simple box-like structures we have used data from the Visible Human project [17] to make a model of the lower leg.

Since the Visible Human data set is voxel-based (see figure 3) it was necessary to generate a mesh model of it. To do this, we first used the Mvox software [2] to manually draw contours on the boundary of the skin and bone in the voxel data (see figure 4). We then applied the Nuages software [7] to create a 3D tetrahedral mesh model of the leg. The result was the FE mesh model shown in figure 4. This model was used in the simulation system described in the next section.

3.3 SGI Performer parallel pipe-lining system

The simulation system has been implemented on an Silicon Graphics ONYX with four Mips R4400 processors using the SGI Performer graphics library. SGI Performer allows the programmer to create parallel pipe-lining software quite easily by providing the basic tools for communication, shared memory etc.

Currently our system can run with 3 processes: The Application, Culling and Drawing processes. The *Application* process handles the actual simulation of the deformable solid, ie. calculates displacements etc. The *Culling* process analyzes the scene that the simulation process provides, and determines which parts are visible in the current window. It then pipes the visible parts on to the *Drawing* process which finally renders the scene.

Notice, that although the entire system is a parallel system the actual deformation simulation system still runs on a single processor. We use the parallel features only to separate rendering from simulation, although we plan to implement a parallel version of the simulation process in the future.

Figure 5 shows a screen dump with the Virtual Operating room environment and the leg lying on the operating table. Figure 6 shows the surface of the FE mesh shown in the simulator.

4 Conclusion

In this paper we have described four methods for real-time simulation of elastic deformation of a volumetric solid based on linear elastic finite elements. We have also described the simulation system we have developed for surgery simulation using these methods.

Performance of the dynamic simulation methods are determined solely by the size of the linear system. We have achieved 20 frames/second for models with up to 250 nodes in the system equation. For the full linear system this includes all nodes, both internally and on the surface. But for the condensed system it only includes the surface nodes. The number of internal nodes of the model, therefore, does not matter for the condensed system since they have been removed from the system equation.

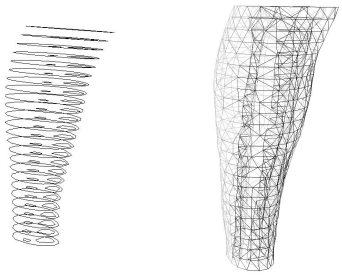

Figure 4: Contours created using Mvox (left) and FE mesh created from contours using Nuages (right).

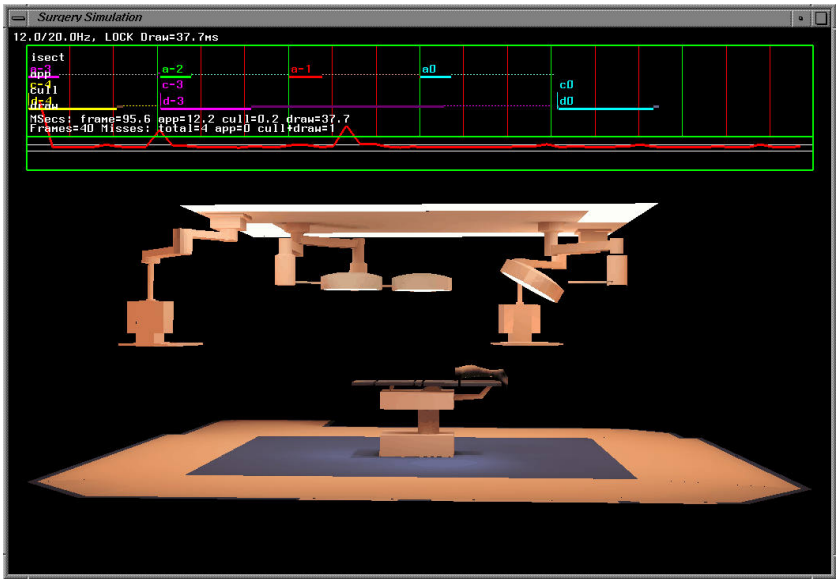

Figure 5: Simulation system implemented using SGI Performer.

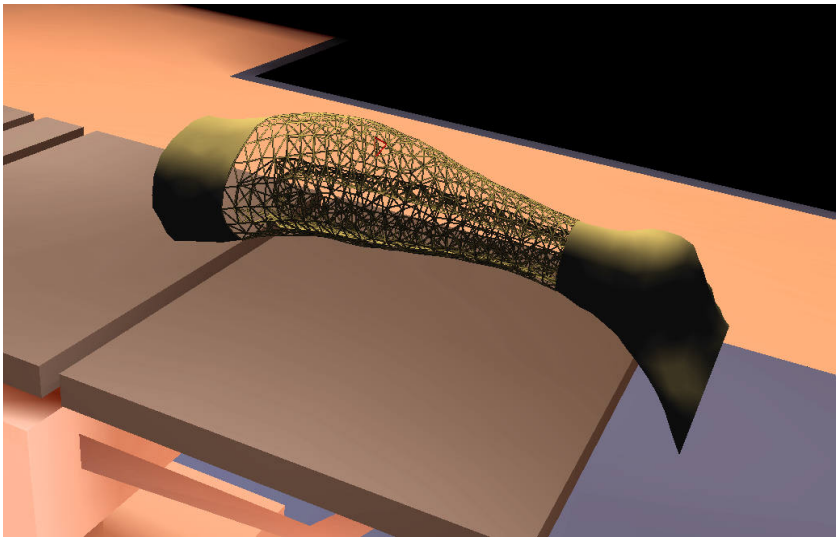

Figure 6: Wireframe model of lower leg in simulator.

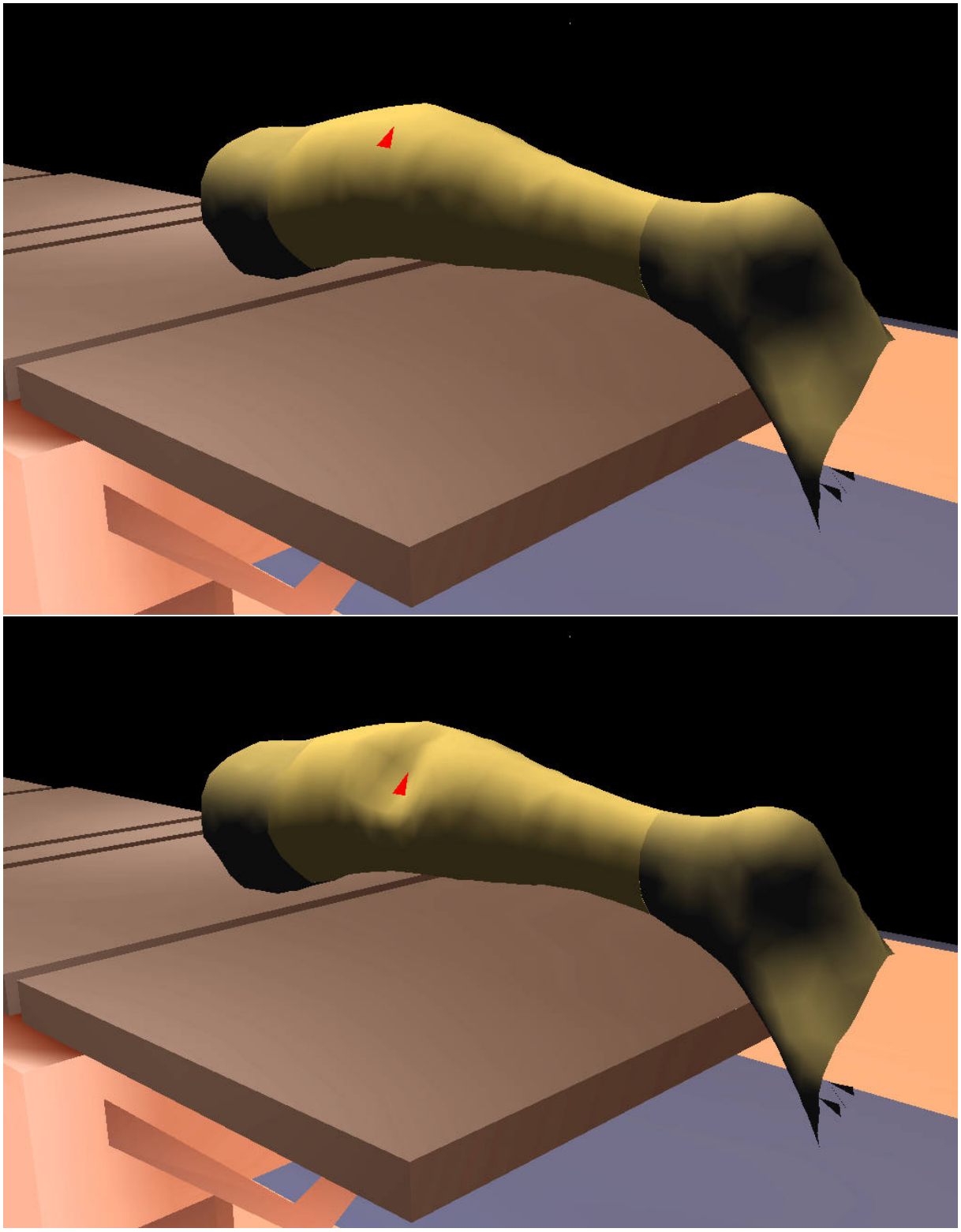

Figure 7: Simulation of pushing on a the lower leg. Top: Default shape. Bottom: Deformation of leg when a push is applied to the black triangle.

It is more difficult to predict the performance of the methods using Selective Matrix Vector Multiplication (SMVM). The above comments concerning the full contra condensed systems apply here also. But in addition the number of nodes that have forces applied to them is very important also.

The example using a leg from the Visible Human data set with 700 system nodes (condensed system with only surface nodes) ran comfortably using only 1/3 of a frame (20 frames/second) when forces were applied to 3 nodes. This included calculation of the deformation and also basic processing. So although both more nodes and more surface nodes with forces applied would increase the time requirement, we believe bigger models could be accommodated using the SMVM method.

Although we have shown that real-time simulation of solid volumetric deformable models is possible there is still much work to be done before realistic surgery simulation can be performed. Most importantly we are currently working on the implementation of cutting in a FE mesh.

In addition to more realistic tissue models we also need detailed segmentation of the organs, limbs etc. to allow different material properties and models to be used. The current parallel research in digital atlasses such as the VoxelMan atlas [9] is a significant step in this direction.

A Linear tetrahedral finite element

We assume that the nodes of the tetrahedron have been numbered as illustrated in figure 2. The *natural coordinates* L_1, L_2, L_3 and L_4 of the tetrahedron are related to the global coordinates x, y and z by (we ignore element superscripts):

$$\begin{bmatrix} 1 \\ x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \end{bmatrix} \begin{bmatrix} L_1 \\ L_2 \\ L_3 \\ L_4 \end{bmatrix} \tag{21}$$

This equation can be inverted to give

$$\begin{bmatrix} L_1 \\ L_2 \\ L_3 \\ L_4 \end{bmatrix} = \frac{1}{6V} \begin{bmatrix} a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ a_3 & b_3 & c_3 & d_3 \\ a_4 & b_4 & c_4 & d_4 \end{bmatrix} \begin{bmatrix} 1 \\ x \\ y \\ z \end{bmatrix} \tag{22}$$

where

$$\begin{aligned} 6V &= \begin{vmatrix} 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \end{vmatrix} \qquad a_1 = \begin{vmatrix} x_2 & x_3 & x_4 \\ y_2 & y_3 & y_4 \\ z_2 & z_3 & z_4 \end{vmatrix} \\ b_1 &= -\begin{vmatrix} 1 & 1 & 1 \\ y_2 & y_3 & y_4 \\ z_2 & z_3 & z_4 \end{vmatrix} \qquad c_1 = \begin{vmatrix} 1 & 1 & 1 \\ x_2 & x_3 & x_4 \\ z_2 & z_3 & z_4 \end{vmatrix} \qquad d_1 = -\begin{vmatrix} 1 & 1 & 1 \\ x_2 & x_3 & x_4 \\ y_2 & y_3 & y_4 \end{vmatrix} \end{aligned} \tag{23}$$

The other coefficients are found by cyclic interchange of the indices.

The $\boldsymbol{B}^e$ matrix becomes:

$$\boldsymbol{B}^e = \frac{1}{6V} \begin{bmatrix} b_1 & 0 & 0 & b_2 & 0 & 0 & b_3 & 0 & 0 & b_4 & 0 & 0 \\ 0 & c_1 & 0 & 0 & c_2 & 0 & 0 & c_3 & 0 & 0 & c_4 & 0 \\ 0 & 0 & d_1 & 0 & 0 & d_2 & 0 & 0 & d_3 & 0 & 0 & d_4 \\ c_1 & b_1 & 0 & c_2 & b_2 & 0 & c_3 & b_3 & 0 & c_4 & b_4 & 0 \\ 0 & d_1 & c_1 & 0 & d_2 & c_2 & 0 & d_3 & c_3 & 0 & d_4 & c_4 \\ d_1 & 0 & b_1 & d_2 & 0 & b_2 & d_3 & 0 & b_3 & d_4 & 0 & b_4 \end{bmatrix} \tag{24}$$

References

[1] R. Barret et al., *Templates for the solution of linear systems: Building blocks for iterative methods*, WWW http://www.netlib.org/templates/templates.ps

[2] M. Bro-Nielsen, *Mvox: Interactive 2-4D medical image and graphics visualization software*, submitted to CAR'96, 1996

[3] M. Bro-Nielsen, *Modelling elasticity in solids using Active Cubes - Application to simulated operations*, Proc. Computer Vision, Virtual Reality and Robotics in Medicine (CVRMed'95), pp. 535-541, 1995

[4] P.G. Ciarlet, *Mathematical elasticity, vol. I: Three-dimensional elasticity*, North-Holland, ISBM 0-444-70529-8, 1988

[5] S. Cotin, H. Delingette, M. Bro-Nielsen, N. Ayache, J.M. Clément, V. Tassetti and J. Marescaux, *Geometric and Physical representations for a simulator of hepatic surgery*, Proc. Medicine Meets Virtual Reality, 1996

[6] S.A. Cover, N.F. Ezquerra and J.F. O'Brien, R. Rowe, T. Gadacz and E. Palm, *Interactively deformable models for surgery simulation*, IEEE Computer Graphics & Applications, pp. 68-75, Nov. 1993

[7] B. Geiger: *Three-dimensional modeling of human organs and its application to diagnosis and surgical planning*, INRIA Tech. Rep. 2105, Dec. 1993

[8] J-P. Gourret, N.M. Thalmann and D. Thalmann, *Simulation of object and human skin deformations in a grasping task*, 23(3):21-30, 1989

[9] K.H. Höhne, M. Bomans, M. Riemer,R. Schubert, U. Tiede and W. Lierse, *A 3D anatomical atlas based on a volume model*, IEEE Computer Graphics Applications, 12(4):72-78, 1992

[10] K.H. Huebner, *The finite element method for engineers*, John Wiley & Sons, ISBN 0-471-41950-8, 1975

[11] H. Kardestuncer, *Finite element handbook*, McGraw-Hill, ISBN 0-07-033305-X, 1987

[12] M.A. Sagar, D. Bullivant, G.D. Mallinson, P.J. Hunter and I. Hunter, *A virtual environment and model of the eye for surgical simulation*, Proc. SIGGRAPH'94, pp. 205-212, 1994

[13] D.E. Stewart, *Meschach: Matrix computations in C*, WWW ftp://ftpmaths.anu.edu.au/pub/meschach/meschach.html, 1992

[14] D. Terzopoulos and K. Fleischer, *Deformable models*, The Visual Computer, 4:306-331, 1988

[15] D. Terzopoulos and K. Waters, *Analysis and synthesis of facial image sequences using physical and anatomical models*, IEEE Trans. on Pattern Analysis and Machine Intelligence, 15(6):569-579, 1993

[16] D. Terzopoulos and D. Metaxas, *Dynamic 3D models with local and global deformations: Deformable superqudraics*, IEEE Trans. on Pattern Analysis and Machine Intelligence, 13(7):703-714, 1991

[17] *The Visible Human Project*, WWW http://www.nlm.nih.gov/extramural_research.dir/visible_human.html