# Modelling of Soft Tissue Deformation for Laparoscopic Surgery Simulation

G. Székely, Ch. Brechbühler, R. Hutter, A. Rhomberg and P. Schmid

Swiss Federal Institute of Technology, ETH Zentrum, CH-8092 Zürich, Switzerland

**Abstract.** Virtual reality based surgical simulator systems offer a very elegant solution to the development of endoscopic surgical trainers. While the graphical performance of commercial systems already makes PC-based simulators viable, the real-time simulation of soft tissue deformation is still the major obstacle in developing simulators for soft-tissue surgery. The goal of the present work is to develop a framework for the full-scale, real-time, Finite Element simulation of elastic tissue deformation in complex systems such as the human abdomen. The key for such a development is the proper formulation of the model, the development of scalable parallel solution algorithms, and special-purpose parallel hardware. The developed techniques will be used for the implementation of a gynaecologic laparoscopic VR-trainer system.

## 1 Introduction

Endoscopic operations have recently become a very popular technique for the diagnosis as well as treatment of many kinds of human diseases and injuries. The basic idea of endoscopic surgery is to minimise damage to the surrounding healthy tissue, normally caused in reaching the point of surgical intervention for the more inacessible of internal organs. The relatively large cuts in open surgery can be replaced by small perforation holes, serving as entry points for optical and surgical instruments. The small spatial extent of the tissue injury and the careful selection of the entry points result in a major gain in patient recovery after operation.

The price for these advantages is paid by the surgeon, who loses direct contact with the operation site. The necessary visual information is mediated by a specialised camera (the endoscope) and is presented on a screen. While preliminary systems experimenting with stereo optics are already available, today's surgery is usually performed under monoscopic conditions. Due to geometrical constraints posed by the external control of the surgical instruments through the trocar hull, the surgeon loses much of the manipulative freedom usually available in open surgery.

Performing operations under these conditions demands very special skills of the surgeon, which can only be gained with extensive training. The basic optical and manipulative skills can be learned today by using inexpensive, traditional training devices. These training units allow one to learn navigating under monoscopic visual feedback, as well as to acquire basic manipulative skills. In this way

the surgeon becomes accustomed to completing a particular task, but because the real-life effect is lost, gets only a limited training range in dexterity and problem solving. Additionally, the organs used by these units are generally made of foam, hence realistic surgical training is impossible. And while experiments on animals are sometimes used for testing new surgical techniques, practical as well as ethical reasons strongly restrict their use in everyday surgical training.

Virtual reality based surgical simulator systems offer a very elegant solution to this training problem. A wide range of VR simulator systems have been proposed and implemented in the past few years. Some of them are restricted to purely diagnostic endoscopical investigations [15, 13, 14], while others, for example, allow the training of surgical procedures for laparoscopic [12, 11, 16, 24, 25], arthroscopic [10, 18], or radiological [26] interventions.

While the graphical performance of commercial systems already renders PC-based simulators possible [15], the real-time simulation of soft tissue deformation is still the major obstacle while developing simulator systems for soft-tissue surgery. Different methods in use for deformation modelling include:

- Free-form deformation techniques of computer graphics [20, 22] use parametric interpolative models (as polynomial models, splines, or superquadrics, e.g.) for deformation estimation of solid primitives. While analogy to physical deformation processes is not always obvious, such techniques have become very popular in surgical simulators [24, 19] due to the resulting fast deformation calculation.
- Different more or less justified simple physical analogies have also been used for tissue deformation modelling. Most popular are mass-spring models [11, 25, 21] but other alternatives like space-filling spheres have also been implemented [27].
- Elastically deformable surface models introduced in computer graphics and computer vision [23] calculate surface deformations by solving the linear elasticity equations using different numerical techniques. These methods allow simulation of tissue deformation based on physical principles [12]. Full 3D extensions of these techniques [16, 17] already represent the first attempts for Finite Element based tissue deformation modelling.

The Finite Element Method (FEM) is a very common and accurate way to solve continuum-mechanical boundary-value problems [1, 8]. In the case of biological tissue, we have to deal with large deformations and also with anisotropic, inhomogeneous, and nonlinear materials. Furthermore, organs and surgical instruments interact, therein leading to numerical contact problems. Nonetheless, provided an adequate formulation is chosen, even in these cases the FEM is a very powerful tool.

Unfortunately, Finite Element calculations are notoriously slow, making them not very appealing for real-time applications like endoscopic surgery simulations. Accordingly, tissue deformation for surgical training systems is, up to now, only calculated by the FEM for fairly simple physical systems [9, 16, 17].

The goal of the presented work is to develop a framework for the full-scale real-time Finite Element simulation of elastic tissue deformation in complex

systems such as the human abdomen. The key for such a development is the proper formulation of the model (see Section 2), the development of scalable parallel solution algorithms (see Section 4) as well as dedicated parallel hardware (see Section 3). The developed techniques will be used for the implementation of a gynaecologic laparoscopic VR-trainer system.

## 2 Elastomechanical Modelling

To enable the virtual reality simulation of surgical operations, we must achieve the following:

- Calculation of realistic deformations and contact forces of the organs
- Performance of the calculations in real-time
- Stable calculations for the time range of the entire simulation

### 2.1 Explicit Finite Element Formulation

A continuum can be described mathematically with the following set of partial differential equations, satisfied for each material point:

$$\mathrm{div}\sigma + \mathbf{f} = \rho\ddot{\mathbf{u}} \quad \text{(momentum equations)} \tag{1}$$

$$\mathrm{div}(\rho\dot{\mathbf{u}}) + \dot{\rho} = 0 \quad \text{(continuity equation)} \tag{2}$$

$$\sigma = f_1(\epsilon) \quad \text{(constitutive law)} \tag{3}$$

$$\epsilon = f_2(\mathbf{u}) \quad \text{(strain formulation)} \tag{4}$$

where $\sigma$ is the Cauchy stress tensor, $\epsilon$ is the strain tensor, $\mathbf{f}$ is the vector of the volume forces, $\mathbf{u}$ stands for the displacements, and $\rho$ is the density. A superposed dot denotes a derivative with respect to time.

Within the FEM a body is subdivided by a finite number of well defined elements (hexahedrons, tetrahedrons, quadrilaterals, e.g.). Displacements and positions in the element are interpolated from discrete nodal values. A trilinearly interpolated hexahedron consists of eight nodes lying in the corners. Figure 2(left) shows a 2D set of elements. Neighbour elements share some nodes. For every element the equations (1)–(4) can be formulated resulting in the following discrete system of differential equations:

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{C}\dot{\mathbf{u}} + \mathbf{K}\delta\mathbf{u} = \mathbf{f} - \mathbf{r}\,, \tag{5}$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{C}$ is the damping matrix, $\mathbf{K}$ is the incremental stiffness matrix, $\mathbf{u}$ is the vector of the nodal displacements, $\mathbf{f}$ are the external node forces, and $\mathbf{r}$ are the internal node forces. All these matrices and vectors may be time dependent.

One possibility to solve equations in (5) is a quasi-static manner [3, 5]. In this case the dynamic part of the equations is neglected ($\ddot{\mathbf{u}} = \dot{\mathbf{u}} = 0$) and the solution is reached iteratively. Within every iteration a huge set of linear

algebraic equations has to be solved. In problems with large deformation and contact interactions the iteration hardly converges and a stable simulation over several minutes is nearly impossible.

Otherwise, the dynamic equations have to be integrated with respect to time [5]. The time integration of the equations can, in principle, be performed using implicit or explicit integration schemes. Using the implicit method, the solution has to be calculated iteratively at every discrete time step, like in the quasi-static problem. Contrary, the explicit time integration can be performed without iteration and without solving a system of linear algebraic equations. This integration scheme is only conditionally stable; that is, only very small time steps lead to a stable solution [4]. An estimation for the critical time step is made by $\Delta t = \frac{\Delta L}{c}$, where $c$ is the maximal wave propagation speed in the medium and $\Delta L$ is the smallest element length of the model. In the case of our model of an uterus and its adnexes, this equation leads to 10'000 time steps per second ($\Delta t = 100\,\mu s$).

These short time steps increase the computational effort but lead to a very stable contact formulation. The disadvantage of this method — that many steps have to be taken — is not so significant since each step is much less time consuming than in an implicit algorithm.

Because of the considerations mentioned above, and based on several numerical tests, we decided to solve the problems with an explicit Finite Element formulation. Unfortunately, this integration scheme is not as accurate as the implicit one and time discretisation errors will usually accumulate. In the next section we will show a way of avoiding this problem.

### 2.2 Element Formulation / Constitutive Equations

The most time consuming part in the explicit formulation is the computation of the internal element forces (see Section 2.3), including the calculation of stresses and strains. These state variables are related to well defined reference configurations. In the following we have to distinguish the updated and the total Lagrange formulation.

Within the updated Lagrange formulation a state is related to the last successfully calculated configuration. Here, strains are usually of an incremental form. This leads to an accumulation of discretisation errors and in consequence to the above mentioned lack of accuracy. Additionally, an incremental strain formulation (even in the case of elastic materials) usually leads to remaining deformations in a stress free state after a load cycle [6, 7]. These errors drastically influence the robustness of a simulation and have to be eliminated.

When a total Lagrange formulation is chosen every state is related to the initial configuration. In this case absolute strain formulations have to be considered (e.g., Green-Lagrange or Hencky). Even with the explicit time integration scheme this leads to an exact static solution, i.e., when the transient terms have been eliminated by damping effects. Contrary to incremental strain formulations, these strains lead to correct results after a load cycle and no error accumulation occurs.

These considerations suggested to use a hyperelastic material law. The Mooney-Rivlin law for nearly incompressible materials leads to an elegant Finite Element formulation and is frequently used in biological tissue modelling. The Mooney-Rivlin material law is a first attempt, and will probably be replaced by a neo-Hookean law, completed with an appropriate damping model.

### 2.3 Volume Integration

To obtain the internal forces of an element we have to solve the following integral:

$$f_i^I = \int_{V^0} F_{il} B_k^I S_{kl} dV^0 \,, \tag{6}$$

where repeated subscripts imply summation over the range of that subscript. $f_i^I$ are the internal forces in direction $i$ of node $I$ of the element, $F_{il}$ are the components of the deformation gradient, $B_k^I$ are the derivatives of the interpolation functions with respect to coordinate $k$, $S_{kl}$ are the components of the 2nd Piola-Kirchhoff stress tensor and $V^0$ is the initial volume of the element.

Commonly, this integral is evaluated by a numerical 8-point quadrature. That is, the integrand will be computed with respect to 8 different integration points, and the eight values are added with a well defined weighting factor. This computation is very time consuming.

The reduced volume integration is a method to circumvent this overhead [2]. Within this scheme the integrand has to be calculated just once for each element. Considering only the mean strains within an element, some deformation modes (hourglass modes) are not influenced by the resulting forces. These modes have to be controlled with additional stabilisation forces (hourglass control):

$$f_i^I = F_{il} B_k^I S_{kl} V^0 + {}^{stab} f_i^I \,. \tag{7}$$

A shortcoming of commonly used formulations for the stabilisation forces is their incremental character. Obviously, this leads to accumulated time integration errors and to incorrect results after load cycles. Consequently we formulate these forces with respect to the initial configuration. This method leads to very stable results even in cases of large deformations. Furthermore, no integration errors will be accumulated.

If Finite Elements are not distorted in the initial configuration absolute formulations have an additional advantage. Many multiplications can be saved, and even the hourglass control needs considerably fewer multiplications.

## 3 Design of a real-time FEM Computation Engine

The only way to provide the necessary computational power for the real-time solution of complex Finite Element systems is to build a parallel computer which supports fully parallel algorithms for the explicit time integration scheme. Algorithmic design as described in Section 4 below allows to scale the computation

to the necessary speed with the selection of an appropriate number of processor units or processing elements (PE). The section summarises the basic requirements and design principles for implementing special-purpose, parallel hardware to perform explicit Finite Element calculations.

### 3.1 Performance Requirements

**Computation** Once the necessary FE calculations and the corresponding parallel algorithms are determined, we have to analyse the performance requirements in detail to find out what kind of computer is needed to satisfy our demands.

Analysis of optimised explicit Finite Element algorithms shows that approximately 700 floating point operations per element are needed in each time step. Additional computation time has to be reserved for collision detection and handling. This leads to 10 MFLOPS (Million Floating Point Operations Per Second) per element for the chosen time step of 100 $\mu$s. For 2000 elements, a total of 20 GFLOPS sustained are needed.

This amount of computational power is much too high for a state of the art workstation. Implicit FE calculation can be implemented well on vector-supercomputers, since the main task is to solve huge sparse systems of linear equations [28, 29]. However, the time steps and the vectors of the explicit method are too short, therefore a high performance parallel machine is needed.

The latest processors are able to deliver >300 MFLOPS with optimised programming, so 64 of these will meet our demands. As Finite Element calculations have a tendency to become unstable, accurate computation using double precision floating point operations is necessary. During the time steps, data has to be sent and received with minimal processor involvement, requiring that very light weight protocols be used.

**Communication** There are three different types of communication: exchange of forces, collision detection and handling, and data transfer to the graphics engine. For a FE model with 2000 elements on a $4 \times 4 \times 4$ processor machine, 5600 force vectors have to be exchanged each time step. This results in a communication bandwidth of 1.35 GByte/s. This is a huge number, but the fact that data only has to be transferred between neighbours eases our requirements.

In collision communication, we note that only part of the surface nodes are potentially colliding with other parts of the FE model, and only about 300 vectors have to be exchanged for a 2000 element model (72 MByte/s).

Finally, the information for all the surface nodes must be sent to the graphics engine. This takes place once every 40ms, but to avoid adding too much latency to the system, data should be sent in a shorter time (20 time steps) . On this basis, 18 MByte/s have to be sent to the graphics engine interface.

### 3.2 Parallel Processing Architecture

According to the previous algorithmic considerations, a three-dimensional mesh of processors can be optimally used, where every processing element (PE) is

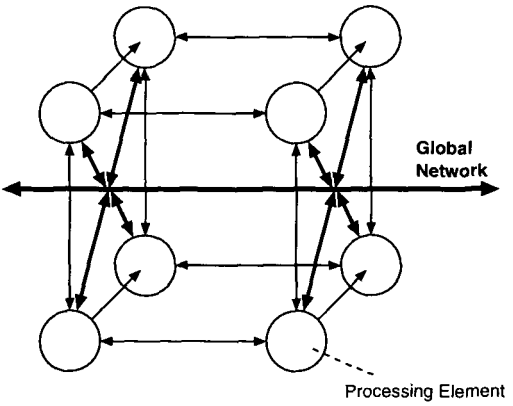


**Fig. 1.** 3D Communication Architecture

connected to its six neighbours (Figure 1). Every PE calculates a cube of the whole FE model, so it has to exchange data from the faces of its cube with PEs that share the same faces. In addition there are also edges and corners, where data has to be shared among processors that are not connected by a direct channel. Such data has to be routed through other PEs. This is actually no drawback, since in most cases the intermediate PEs have also data to transmit to the same target PE, so they just add their values before transmitting the received data block.

**Local Communication** To achieve the necessary low latency, we have to exploit the specialties of our communication pattern. If data has to be exchanged between processors, it must be communicated in every time step. Fortunately it is known at programming time which data values have to be sent, in what order, and to which destination. Secondly, when the elements are assigned to the processors, only those processors with elements sharing nodes on the common face of the cubes have to exchange data (collision detection is an exception to this). Bearing this in mind, the best way to provide sufficient communication bandwidth with low latency and minimal software overhead is to use point to point communication and to transmit pure data without additional address information. There is no need for routing information when point to point channels are used, and sending only data values minimises software overhead on the sender side. But the data still has to be recognised by the receiver. The address information, however, of the data is implicitly given by the order within the time step it is sent. It is known at compile time which values have to be exchanged, so the tool that partitions the elements also sets up the order in which data values are sent and received.

**Global Communication** Communication takes place mostly between neighbours, although data for collision detection has to be exchanged between two

processors that may be far away within the mesh. Part of these data values has to be broadcast to all other processors. Additionally, several processors can send and receive data at the same time.

Routing such data through the local channels is very inefficient. Bandwidth is used that is needed for local communication. The routing may consume processor time which can be better utilised for FE calculation and routing through several PEs increases latency to an unacceptable level. Therefore, such messages are better sent over a global network over which any PE can exchange data with all others. Data must be identified, thus packets of data must be sent with information concerning the receiver as well as an identification.

## 4 Partitioning the Model for Parallel Computation

Explicit FE computation suggests element-wise parallelism, decomposing the spatial domain. Each element is mapped to exactly one processor, and one processor computes the internal forces of several elements (see Figure 2 left). Whenever elements residing on different processors share common nodes, these


**Fig. 2.** *Left:* A small 2D model serves for illustrating the concepts on paper. The six different shades of gray illustrate an arbitrary assignment of elements to $2 \times 3$ processors. *Middle:* The coordinates of the processors to which parts of the model (groups of elements) are assigned. Dashed lines show $x$ borders, dotted lines $y$ borders. *Right:* The model is partitioned in submodels. Grey shades indicate batches for recovery, arrows mark nodes on the border

nodes must be represented on all involved processors. The resulting force acting on such a distributed node emerges from the communication among the contributing processors, which treat the foreign forces like external forces, adding them to their own.

### 4.1 Parallel Computation

Due to the inherently 3D nature of the problem, 3D processor networks appear to be the optimal topology for a multiprocessor architecture. PEs are are labelled with "coordinates", which generally do not correspond to geometric coordinates (see Figure 2 middle).

The mapping of elements to processors should balance the computational load while minimising the demand for communication bandwidth.

A force acting on a node must be applied exactly once. This is inherently given for inner forces (from recovery). Inertial and gravity forces are applied only on one PE, where the node is circled black in Figure 2(right). Other nodes (circled gray) initialize forces to 0.

The mapping defines the sequence of computation and communication which have to run on the parallel computer. To make effective use of the parallel architecture, the load should be balanced among the single processors, and waiting for data from other processors should be minimised. The procedure sketched below aims at this goal. The term "recovery" means determining the interior reaction forces within one element (eqn. (6)).

The summation over all elements is split into four phases with communication interlaced between them. Nodes on the $x$ border are marked with a horizontal arrow. Before sending in direction $x$, all elements with such a node must be computed; they are in batch $x$, and the figure shows them shaded in a darker gray. While the forces are being transferred, elements in the $y$ batch (lighter gray) are recovered. (No $z$ batch exists in the 2D illustration.) The remaining elements are local to the PE; they belong to the "inner" batch (white).

| |
|---|
| recover elements in batch $x$ |
| send $x$ border nodes in direction $x$ |
| recover elements in batch $y$ |
| add foreign forces from direction $x$ |
| send $y$ border nodes in direction $y$ |
| recover elements in batch $z$ |
| add foreign forces from direction $y$ |
| send $z$ border nodes in direction $z$ |
| recover elements in "inner" batch |
| add foreign forces from direction $z$ |

When possible, the recovery of some "own" elements is computed between sending partially assembled node forces in some direction and using the foreign forces from said direction.

### 4.2 Collision Detection

The next part is collision detection, which uses the global network for data transfer. We adopt an approximation to collision detection that is commonly used with the FEM. This method only determines which nodes of one contact partner, the slave, penetrate any face of the other contact partner, the master. Flipping the roles of master and slave and averaging will balance this obvious asymmetry. To save time on collision detection, the dynamic behaviour of the model is taken into account. Elements do not move faster then 10 cm/s, so if two elements are 2cm apart, the earliest time they can touch is 100 ms or 1000 time

steps away. A collision between these elements cannot take place within the next 1000 time steps and therefore needs no attention. Secondly, the system is built in a hierarchical manner. Every PE first checks bounding boxes with every other PE, before checking for collisions on an element level.

Thus collision detection takes the following steps:

- Every 1000 time steps all bounding boxes are compared.
- If close enough bounding boxes are detected, the corresponding PEs establish a communication channel to exchange the displacements of nodes that may be involved in collisions.
- During the next 1000 time steps, the displacements of these nodes are sent to and received from the other PEs. Collisions are detected and each PE determines the consequences for its nodes. Because these computations are symmetrical, all force vectors generated by collisions sum to zero.

Since the communication channels are only redefined every 1000 time steps, software overhead is minimised.

## 5 Conclusion

Explicit Finite Element analysis leads to very stable simulations even in the case of large deformations and contact problems. The use of a total Lagrange formulation increases the stability and eliminates integration errors in the static solution. Reduced integration decreases the calculation costs by a factor of 5–7 and nevertheless leads to very stable and sufficiently accurate results when the stabilisation forces are calculated with respect to the initial configuration.

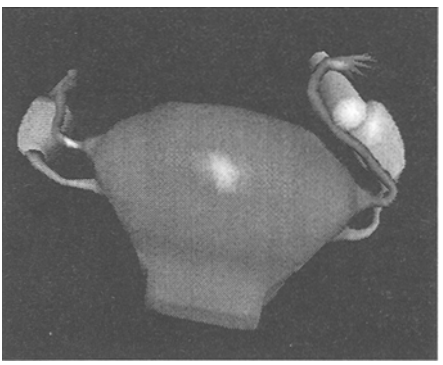

**Fig. 3.** Deformation of the fallopian tubes with a blunt probe.

Appropriately designed parallel algorithms enable implementation of the explicit integration method on a large, 3-dimensional network of high-performance processor units, allowing the subsequent usage of these techniques even in real-time surgery simulator systems. The usefulness of the developed methods have

been demonstrated with the simulation of deformations of the uterus' fallopian tubes during diagnostic gynaecological laparoscopy (see Figure 3). The implementation of the complete laparoscopic simulator system including the construction of the specialised FEM computational engine is in progress.

## References


1. K.J. Bathe: *Finite Element Procedures*, Prentice Hall, Englewood Cliffs, New Jersey 1996
2. T. Belytschko and S. Ong: *Hourglass Control in Linear and Nonlinear Problems*, Computer Methods in Applied Mechanics and Engineering, **43**:251–276, 1984
3. M.A. Crisfield: *Non-linear Finite Element Analysis of Solid and Structures - Volume 1: Essentials*, John Wiley & Sons, Chichester 1991
4. D.P. Flanagan and T. Belytschko: *Eigenvalues and Stable Time Steps for the Uniform Strain Hexahedron and quadrilateral*, Journal of Applied Mechanics, **51**:35–40, March 1984
5. E. Hinton: *NAFEMS - Introduction to nonlinear Finite Element Analysis*, Bell and Bain Ltd., Glasgow 1992
6. M. Kojic and K.J. Bathe: *Studies of Finite Element Procedures-Stress Solution of a Closed Elastic Strain Path with Stretching and Shearing Using the Updated Lagrangian Jaumann Formulation*, Computers & Structures, **26**(1/2):175–179, 1987
7. S. Roy et al.: *On the Use of Polar Decomposition in the Integration of Hypoelastic Constitutive Laws*, International Journal of Engineering Science, **30**(2):119–133, 1992
8. O.C. Zienkiewicz and R.L. Taylor: *The Finite Element Method*, McGraw-Hill Book Company, London 1994
9. M.A. Sagar, D. Bullivant, G.D. Mallinson, P.J. Hunter and I.W. Hunter: *A Virtual Environment and Model of the Eye for Surgical Simulation*, Comp. Graphics **28**:205–212, 1994
10. R. Ziegler, W. Mueller, G. Fischer and M. Goebel: *A Virtual Reality Medical Training System*, Proc. $1^{st}$ In. Conf. on Comp. Vision, Virtual Reality and Robotics in Medicine, CVRMed'95, Nice, Lecture Notes in Comp. Sci., **905**:282–286, Springer-Verlag, 1995
11. U.G. Kühnapfel, H.G. Krumm, C. Kuhn, M. Hübner and B. Neisius: *Endosurgery Simulations with KISMET: A flexible tool for Surgical Instrument Design, Operation Room Planning and VR Technology based Abdominal Surgery Training*, Proc. Virtual reality World'95, Stuttgart, 165–171, 1995
12. S.A. Cover, N.F. Ezquerra and J.F. O'Brian: *Interactively Deformable Models for Surgery Simulation*, IEEE Comp. Graphics and Appl. **13**:68–75, 1993
13. D.J. Vining: *Virtual Endoscopy: Is It Really?*, Radiology, **200**:30–31, 1996
14. : R. Satava: *Virtual Endoscopy: Diagnosis using 3-D Visualisation and Virtual Representation*, Surgical Endoscopy, **10**:173–174, 1996
15. A.M. Alyassin, W.E. Lorensen: *Virtual Endoscopy Software Application on a PC*, Proc. MMVR'98: 84–89, IOS Press, 1998
16. S. Cotin, H. Delingette, J.M. Clément, M. Bro-Nielsen, N. Ayache and J. Marescaux: *Geometrical and Physical Representations for a Simulator of Hepatic Surgery*, Proc. MMVR'96:139–151, IOS Press, 1996
17. M. Bro-Nielsen and S. Cotin: *Real-time volumetric deformable models for surgery simulation using Finite Elements and condensation*, Comp. Graphics Forum **15**(3):57–66, 1996

18. S. Gibson, J. Samosky, A. Mor, C. Fyock, E. Grimson, T. Kanade, R. Kikinis, H. Lauer, N. McKenzie, S. Nakajima, H. Ohkami, R. Osborne and A. Sawada: *Simulating Arthroscopic Knee Surgery using Volumetric Object Representations, Real-Time Volume Rendering and Haptic Feedback*, Proc. CVRMed'97:369–378, Springer-Verlag, 1997
19. C. Basdogan, Ch-H. Ho, M.A. Srinivasan, S.D. Small, S.L. Dawson: *Force interactions in Laparoscopic Simulations: Haptic Rendering of Soft Tissues*, Proc. MMVR'98:385–391, IOS Press, 1998
20. A.H. Barr: *Global and Local Deformations of Solid Primitives*, Computer Graphics **18**(3):21–30, 1984
21. M. Bro-Nielsen, D. Helfrick, B. Glass, X. Zeng, H. Connacher: *VR Simulation of Abdominal Trauma Surgery*, Proc. MMVR'98:117–123, IOS Press, 1998
22. T.W. Sederberg and S.R. Parry: *Free-Form Deformation of Solid Geometric Models*, Computer Graphics **20**(4):151–160, 1986
23. D. Terzopoulos, J. Platt, A. Barr and K. Fleischer: *Elastically Deformable Models*, Computer Graphics **21**(4):205–214, 1987
24. Ch. Baur, D. Guzzoni and O. Georg: *Virgy: A Virtual Reality and Force Feedback Based Endoscopy Surgery Simulator*, Proc. MMVR'98:110–116, IOS Press, 1998
25. M. Downes, M.C. Cavusoglu, W. Gantert, L.W. Way and F. Tendick: *Virtual Environments for Training Critical Skills in Laparoscopic Surgery*, Proc. MMVR'98:316–322, IOS Press, 1998
26. J.K. Hahn, R. Kaufman, A.B. Winick, Th. Carleton, Y. Park, R. Lindeman, K-M Oh, N. Al-Ghreimil, R.J. Walsh, M. Loew, J. Gerber and S. Sankar: *Training Environment for Inferior Vena Caval Filter Placement*, Proc. MMVR'98:291–297, IOS Press, 1998
27. N. Suzuki, A. Hattori, T. Ezumi, A. Uchiyama, T. Kumano, A. Ikamoto, Y. Adachi, A. Takatsu: *Simulator for virtual surgery using deformable organ models and force feedback system*, Proc. MMVR'98:227–233, IOS Press, 1998
28. V.E. Taylor: *Application-Specific Architectures for Large Finite-Element Applications*, PhD Thesis, Dept. El. Eng. and Comp. Sci., Univ. of California, Berkeley, CA, 1991
29. C. Pommerell: *Solution of large unsymmetric systems of linear equations*, PhD Thesis, Department of Electrical Engineering, ETH Zürich, 1992