The Simulation of Elastic Tissues in Virtual Medicine Using Neuro-Fuzzy Systems

Arne Radetzky[a], Andreas Nürnberger[b] and Dietrich Peter Pretschner[a]

[a]Institute for Medical Informatics, Technical University of Braunschweig,
Fallersleber-Tor-Wall 22, D-38100 Braunschweig, Germany

[b]Institute for Knowledge and Language Processing, University of Magdeburg,
Universitaetsplatz 2, D-39106 Magdeburg, Germany

ABSTRACT

To improve the benefit of surgical simulators for education and research a visual convincing modeling of the operation scenario and the involved tissues is not sufficient. It is also necessary to simulate the deformation and resulting inner forces of tissue under influence of external forces caused by, for example, medical instruments or gravity. In this paper, we present a hybrid neuro-fuzzy system, which was designed for the description and simulation of tissues. The neuro-fuzzy system can be used to simulate the physical behavior like stiffness, viscosity and inertia of deformable or elastic tissues in surgical simulation. The parameters of a physical model or prior expert knowledge in the form of linguistic terms can be used to initialize the network parameters. Using a neural network structure, local changes to the system like cuts or ruptures can be performed during simulation. As an application example, some simulation results in the area of gynaecological laparoscopy are given.

Keywords: virtual reality, surgical simulation, deformable tissue, neuro-fuzzy system, neural network, fuzzy system

1. INTRODUCTION

A main goal of surgical simulators is to create a virtual training environment for prospective surgeons. Thus, the surgeon is enabled to rehearse the various steps of surgical procedures without any risks for the patient. Further advantages are the riskless trial of new operation methods and the possibility to reduce operation time, training costs and the number of animal experiments.

Nowadays computer assisted surgery is already used for procedure training and operation planning. Most of the utilized methods are based on static visualization techniques. Therefore, they can mainly be used for the visualization of the human anatomic.

To improve the benefit for surgical training a visual convincing of the operation scenario and the involved tissues is not sufficient, because a surgeon relies on his tactile senses during the operation to a great part. Especially for the interaction with medical devices, such as laparoscopic instruments, it is necessary to simulate the deformation of tissue under influence of collision forces. Additionally, the internal, compensating forces of the tissue must be calculated, so that the tissues can be 'felt' by the prospective surgeon. For this, the use of a force feedback model is of special importance.

The elasticity of structures can be described by use of differential equations and physical laws[2,4]. In many cases, a model of a physical system can not be obtained by conventional approaches. This can be caused by problems in constructing a system of differential equations and/or in finding the required parameters. Furthermore, the simulation of such a model description is often very time consuming and so could not be used in real time applications. A further problem is that even small or local changes to the model structure often require a rebuild of the whole system description.

Nevertheless, a mathematical exact computation is not necessary. The noticed haptic response can be seen as vague data. Thus, it seems possible to use fuzzy techniques for the description of these response signals.

For these reasons, we developed a model, which is motivated by a combination of a fuzzy system and an artificial neural network, a so-called hybrid neuro-fuzzy system. By using a fuzzy system, it is possible to integrate expert knowledge in form of medical terms to define the haptic and visual response of the tissue. In addition, with artificial neural network techniques it is possible to learn or adapt the parameters of the developed model. Measured data or physical models of the tissue can be used for learning.

In the following, we present parts of our actual work on this project. The underlying structure of the model is motivated and its implementation is presented. Furthermore, learning approaches are shown, and an example of a gynaecological laparoscopy in a virtual simulation environment is given.

2. THE USED MODEL STRUCTURE

As in traditional approaches the mass points of the object, in this case the tissue, are represented as nodes in a mesh[2]. Thus, the mass of the whole object is divided up between the mass points. We choose this approach because the mesh structure can be easily transferred into a network structure. Furthermore, it is easy to perform cuts in a mesh-based model and existing visualization procedures can be used (E.g. texture mapping, surface rendering and volume rendering which are already implemented in hardware.).

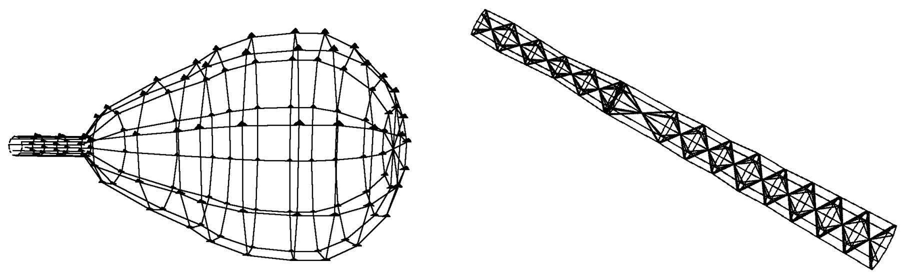
(a) Gall bladder *(b) Fallopian tube*

Figure 1. Mass points (nodes) connected via springs as surface meshes

Every node of the mesh can be connected elastically or viscously to its neighbors (see Figure 1 and Figure 2). The connections are represented as springs. The use of surface meshes to simulate organs is sufficient in most cases (e.g. vessels). The shape can be stabilized with additional springs at the inside (see Figure 1 *(b)*).

Additional nodes in the inner structure could be necessary for the simulation of more complex objects. These volumetric meshes are needed especially if cuts must be performed or the volume must be preserved during deformation. Figure 2 shows three examples of volumetric meshes. On the left side (*(a)*) a straightforward approach of a volumetric mesh is depicted. The volume is divided in simple cubes. This mesh structure could only be used for the simulation of small deformations, because non-plausible shear effects could occur. Therefore, the mesh must be stabilized with diagonal springs (see Figure 2 *(b)*).

However, with the diagonal springs each internal node is connected by ten springs to its neighbors, so the model becomes unnecessarily complex. To simplify the model and thus increasing the computational performance for the simulation, the number of springs should be reduced. To minimize the number of springs in volumetric meshes without decreasing stability, crystal structures have been investigated.

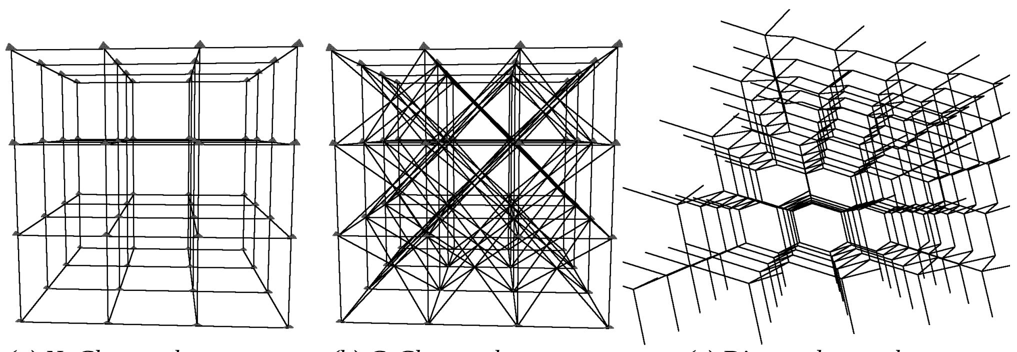
(a) NaCl crystal structure *(b) CsCl crystal structure* *(c) Diamond crystal structure*

Figure 2. Some examples of volumetric meshes

The volumetric mesh in Figure 2 *(a)* has the same structure as a NaCl crystal[10]. Of course, the rigid connections are replaced by springs. With the diagonal springs in Figure 2 *(b)*, the structure is nearly the same as in a CsCl crystal. In diamond structures (see Figure 2 *(c)*) each node is connected with its neighbors by only four springs. Nevertheless, the mesh structure of the diamond is at least as stable as the volumetric mesh in Figure 2 *(b)*.

To every node of the mesh, external forces can be applied (for example gravity or collision forces). The internal forces and the displacement of the nodes are computed correspondingly. Also, the displacement of a node can be given and the resulting forces are calculated.

3. THE NEURO-FUZZY SYSTEM

Fuzzy systems and neural networks are successfully used in the area of control theory, data analysis, and knowledge based systems[8,9,11]. Fuzzy systems can be used to derive parameters of dynamic systems, if only vague data about the system is available. Artificial recurrent neural networks can be used to simulate the dynamic of time-dependent systems. Furthermore, neural networks can be trained to simulate the behavior of real dynamic systems. Therefore, we choose a hybrid approach[1] for the description and simulation of elastic tissue in medicine.

The fuzzy-system is used for the derivation of the network parameters, which defines the behavior of the simulated tissue. The artificial recurrent neural network is used for the simulation process. Furthermore, it can be used to learn or adapt the parameters of the network, if measured data exist. So, the neural network and the fuzzy system can be described separately.

3.1. The network model

The presented network uses a problem specific structure. The structure of the neural network was designed to speed up the simulation and learning process for elastic solids. In contrast to common neural network models, this model uses vectors instead of single input and output signals. So the standard model of a neuron was vectorized to fit our needs.

The structure implements the system of differential equations, which defines the spring model. Two different network nodes are used for this purpose: the mass and the spring nodes.

3.1.1. Mass point dynamic

The neurons determining the mass dynamics (inertia) are divided in three 'sub-neurons' which calculate the position, velocity, and acceleration of the mass point (see Figure 3). An external force can be applied to the neuron, which calculates the actual acceleration. The velocity and position neurons are self-connected feedback nodes. These neurons are used as integrators (see, for example, Ref. 5 or 9).

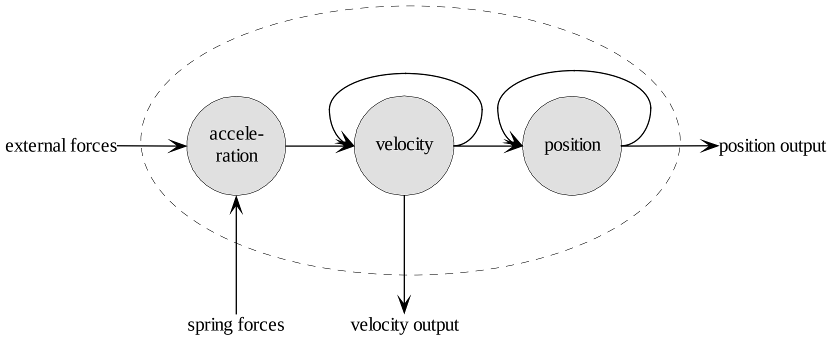

Figure 3. Neurons describing the mass point dynamic

Let t_c be the time constant used for simulation, s be the number of springs, and N_{S2} be the connection matrix with n_{i1} and n_{i2} be the node numbers connected to spring i, then the neurons can be defined by the following equations.

[1]Some further concepts of hybrid systems are described by Nauck et al. [11].

The network input, output and activation functions for the neurons are defined as:

$$net_j = \sum_i (w_{ij} \cdot o_i)$$

$$a_j = A_j(net_j) = net_j$$

$$\begin{array}{ll} \text{acceleration neuron:} & o_j = O_j(a_j) = a_j \\ \text{velocity and position neuron:} & o_j = O_j(a_j) = t_c \cdot a_j \end{array} \qquad (1)$$

Let *k* be the number of the node, then the weights are initialized with the following values:

$$\begin{array}{ll} \text{acceleration neuron:} & w_{ij} = \begin{cases} 1 & if \quad k = n_{i1}, \\ -1 & otherwise. \end{cases} \\ \text{velocity neuron:} & w_{ij} = \frac{1}{m}, \quad \text{where m is the mass of the node.} \\ \text{position neuron:} & w_{ij} = 1 \end{array} \qquad (2)$$

These nodes implement the following physical laws, which are used for the definition of the physical spring model:

$$\begin{array}{ll} \sum_i F_i = F = m \cdot a & \text{(acceleration and velocity neuron)} \\ \frac{dv}{dt} = a & \text{(velocity neuron)} \\ \frac{dp}{dt} = v & \text{(position neuron)} \end{array} \qquad (3)$$

3.1.2. Spring dynamic

The neurons determining the spring dynamics (see Figure 4) calculate the actual spring force. As in physical based models, the velocity vectors are used to simulate viscosity.

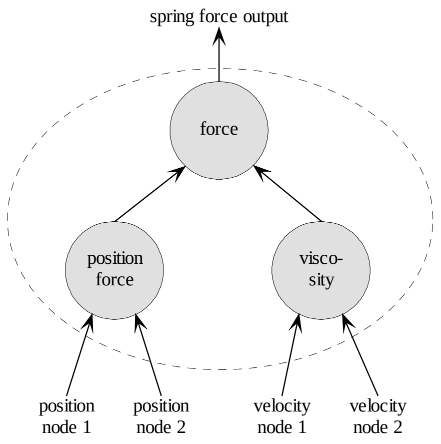

Figure 4. Neurons describing the spring dynamic

The net input and output function for all neurons are defined as:

$$\begin{aligned} net_j &= \sum_{i=1}^{2}(w_{ij} \cdot o_i) \\ o_j &= O_j(a_j) = a_j \end{aligned} \tag{4}$$

The activation function and the initial weights of the sub-neurons are defined as follows. The position force neuron is defined by:

$$\begin{aligned} & w_{1j} = 1,\ w_{2j} = -1 \\ & a_j = A_j(net_j) = SpringFunction(net_j), \end{aligned} \tag{5}$$

where *SpringFunction* defines the used spring function (for example, if a linear model is used a linear function which defines the spring forces). The velocity force neuron is defined by:

$$\begin{aligned} & w_{1j} = 1,\ w_{2j} = -1 \\ & a_j = A_j(net_j) = ViscosityFunction(net_j), \end{aligned} \tag{6}$$

where *ViscosityFunction* defines the used viscosity function. In case of linear models let *c* be the spring constant and *d* be the viscosity constant, otherwise let *c* = *d* = 1. Then the force neuron is defined by:

$$\begin{aligned} & w_{1j} = c,\ w_{2j} = d \\ & a_j = A_j(net_j) = 1 \cdot net_j \end{aligned} \tag{7}$$

These nodes implement the physical laws for spring forces. With equation (3) these forces can be defined as:

$$F = f(p,v) + d(p,v)\,, \tag{8}$$

where *f(p,v)* defines the spring force and *d(p,v)* the damping force or viscosity.

3.1.3. The network structure

A sample of a 2D-mesh is shown in Figure 5. As shown the network is structured by alternating spring and node layers. Of course, different structures can be used to model the objects (see also section 2). The system of differential equations defined by this structure can be derived by equations (3) and (8) (see, for example, Ref. 2 or 4).

The model structure can be easily modified during simulation to be able to realize modifications of the simulated object. Performing cuts, for example, can be done by removing springs. Besides, it is possible to simulate ruptures if the spring forces are greater than a threshold value.

3.1.4. Propagation

In a first approach[15] we choose a algorithm similar to the propagation in Hopfield networks[6]. This propagation algorithm was separated in two steps. During the first step, the network was propagated until a local energy minimum was reached. The second step was used to calculate the activities a_i of the sub-neurons for the next time step.

However, with rigid tissues stability problems could occur during the propagation process. If the parameters of the network are badly defined (e.g. small spring constants, zero viscosity) the network tends to an unstable (chaotic) behavior. For example, in case of great force values and large spring constants, the calculated displacement of the nodes during one time step t_c could be much to far. Thus, the propagation process could start to oscillate or even diverge. The problem can be solved choosing a very low time constant. But then, the propagation process can not be done in real time.

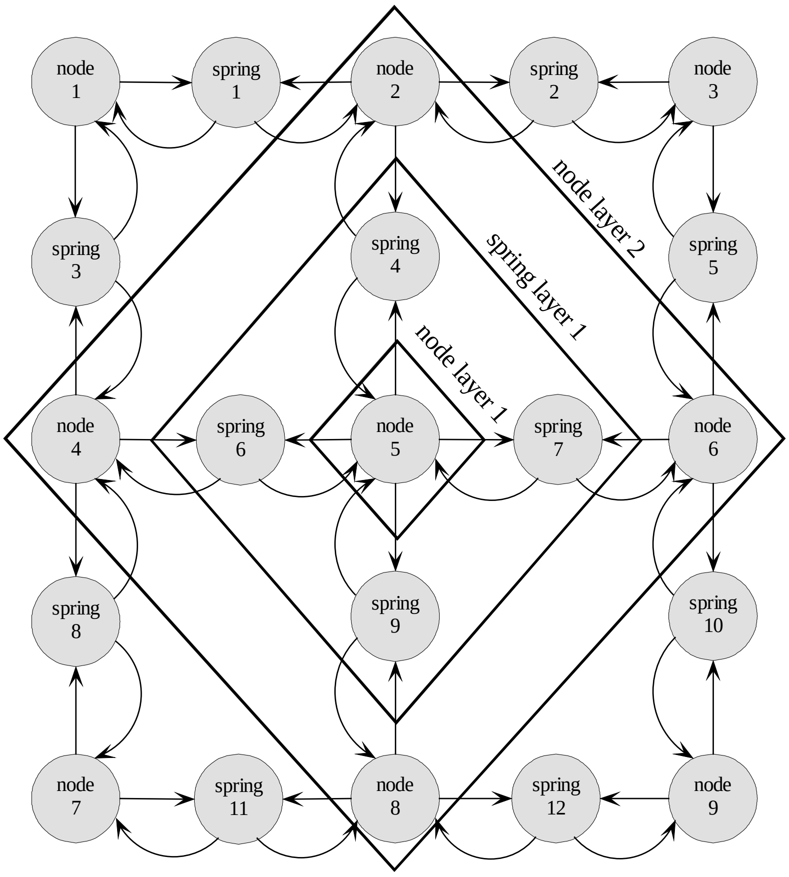

Figure 5. Network representation of a 2D-mesh

However, experiments have shown that unstable behaviors only occur during very short time intervals, so that a variable time step can be used. The propagation algorithm is shown in Table 1. Initially t_c is equal to a predefined global simulation refresh rate t_Δ (e.g. $t_\Delta = 0.05$).

The elasto-dynamic structure could not produce greater internal forces then the external forces, which are formerly applied to the structure. Therefore, the greatest sum of the actual internal force vector values should not be greater then the greatest sum of former force vector values. If that happens then t_c is reduced and the propagation starts over. Otherwise, t_c can be risen slowly.

Table 1. The propagation algorithm

```
$t_{start} := t$;
$t_c := t_\Delta$;
repeat
        propagate all springs at time $t$;
        propagate all nodes at time $t$;
        if |greatest sum of force vector values| > |greatest sum of former force vector values| then
                $t_c := t_c \cdot 0.5$;
        else
                $t := t + t_c$;
                $t_c := t_c \cdot 2$;
        end if
until $t \geq t_{start} + t_\Delta$;
```

At the end of this propagation step, the activity of the node sub-neurons defines the position, velocity and acceleration of the mass points at time $t_0 + n \cdot t_\Delta$, with n be the number of the actual propagation step. During propagation, all spring layers are calculated. The inputs of the spring layers are the position and velocity vectors of the nodes connected with the springs. The output of each spring is a force vector in the direction of the spring. If all springs are updated the propagation of the node layer is started.

As net input of each *node* the weighted sum of the external and internal force vectors is used (see Figure 3). The internal force vectors are received from the connected spring nodes. Under consideration of the node's mass, the acceleration-neuron computes the node's acceleration, which is used as input of the velocity neuron. The self-feedback at the velocity and position neurons performs a time-dependent integration. The dynamic behavior is realized with the additional feedback over the *springs* (see Figure 4).

The computation can be done parallel with all nodes and springs. To compute this network it is effective to group several nodes and springs to greater units and solve each unit on a single processor.

The order of computation of the node and spring layers during propagation is important to minimize the required steps. Therefore, a heuristic is used to determine the order of computation. During the first propagation, the node with the greatest force vector is identified. Then the next propagation starts with this node and continuously goes on layer by layer until all nodes are computed. During the propagation, the node with the greatest force difference between the old force and the new computed force is identified. If the force difference is greater than a threshold value, the propagation starts over.

During propagation, the network is updated using the time constant t_Δ. In real time applications, this constant can be used to define the graphic refresh rate. Thus, it is possible to synchronize the propagation process with a real time environment.

3.1.5. The learning / initialization methods

We are currently working on learning methods, which are based on backpropagation learning methods for recurrent neural networks (see, for example, Refs. 5, 9, 12, 17, 18; an overview of some learning approaches – also for time dependent and real-time recurrent learning - is given in Refs. 13 and 14).

The learning algorithms use measured data or data generated by an exact physical model of the tissue for learning. The learning algorithms use the position of every node at discrete time-steps as input. The learning algorithm tries to minimize the error (total cost) function of the network, which is defined as:

$$E = \sum_{t=0}^{T} E(t) = \sum_{t=0}^{T} \left[\frac{1}{2} \sum_{k} (E_k(t))^2 \right]$$

Let $t = 0, 1, ..., T$ be the time-step, $y_k(t)$ the position of node k and $p_k(t)$ the position of the node k in the exact physical model at the time-step t, then $E_k(t)$ is defined as

$$E_k(t) = \begin{cases} p_k(t) - y_k(t) & \text{if node k has a desired output } p_k \text{ at time t,} \\ 0 & \text{otherwise.} \end{cases}$$

The parameters (*weights*) of the network are adapted after each time-step by a gradient descent method. The learning process is finished if E is sufficient small. Thus, even local differences in the solid structure can be learned. During learning only the weights of the velocity sub-neurons (the masses m_j) and the spring neurons are modified to ensure the interpretability of the network. Although, we are still at the beginning of our work concerning these learning approaches, the first results are very promising.

Another approach to initialize the parameters of the network is to use the constants of the real physical model to the set the parameters in equations (2), (5), and (6). The initialization by use of a fuzzy system[8,11] is presented in the following section. The fuzzy system can be used to describe the relations between existing (vague) expert knowledge of the solid behavior (for example 'very hard', 'soft', 'elastic') and the network parameters.

3.2. The fuzzy system

To be able to use a-priori knowledge for the initialization of the network parameters we use a fuzzy system. A fuzzy system approximates an unknown function based on vague samples, which are described by linguistic rules, so-called fuzzy rules[7,8]. The fuzzy rules make use of linguistic terms defined by fuzzy sets to describe vague data.

A fuzzy system consists of r parallel rules. Let $x_1, ..., x_n$ be n input values, $y_1, ..., y_m$ the m output values and $\mu_r^{(1)}, ..., \mu_r^{(n)}$ and $\nu_r^{(1)}, ..., \nu_r^{(m)}$ fuzzy sets of the rule r, then the fuzzy rules of a fuzzy system are defined as:

$$\textit{if } x_1 \textit{ is } \mu_r^{(1)} \textit{ and ... and } x_n \textit{ is } \mu_r^{(n)} \textit{ then } y_1 \textit{ is } \nu_r^{(1)} \textit{ and ... and } y_m \textit{ is } \nu_r^{(m)} .$$

To calculate the output of this system, the outputs of every rule are computed first. Then, all outputs are combined into a single system output (usually, fuzzy sets). A crisp (exact) output value is derived by a defuzzification procedure[7], e.g. the center of gravity of the resulting fuzzy set. To construct a fuzzy system, the fuzzy rules as well as the membership functions describing the fuzzy sets have to be defined.

The constructed fuzzy system describes the relations between the vague knowledge of the tissue (e.g. ‘very hard’, ‘soft’, ‘elastic’) and the network parameters. As input values (domains) we currently use hardness, elasticity, weight and consistency. The fuzzy system can be easily extended to different domains.

As output values the physical parameters of the network model are determined, those are spring constant, viscosity, mass and fraction force. Every domain was partitioned by fuzzy sets representing the linguistic terms used. An example is given in Figure 6.

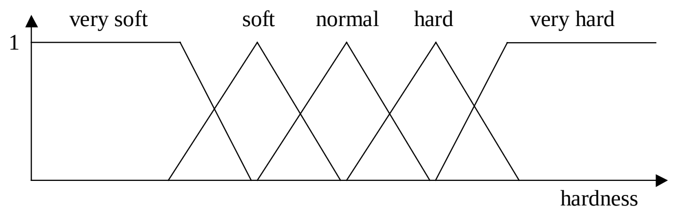

Figure 6: Partitioning of the domain hardness

Some sample fuzzy rules are shown in Table 2. The fuzzy rules were found out by inquiring of experts and they are optimized manually. Currently we are working on neuro-fuzzy methods to optimize the derived rule base.

Table 2. Some fuzzy rules for the description of tissue.

if hardness *is* soft *and* elasticity *is* big	*then*	spring constant *is* low
if shiftability *is* high	*then*	spring constant *is* very low *and* viscosity *is* high
if consistency *is* chapped	*then*	fraction force *is* low *and* spring constant *is* very high
if consistency *is* slightly hard	*then*	spring constant *is* slightly high *and* viscosity *is* low

The fuzzy system defined in this way makes it possible to define the parameters of the network by linguistic terms or - in a graphical implementation of the network generation utility - by sliders. Of course, specific parts of the network can be defined separately, to make the simulation of tissue consisting of different layers possible.

4. APPLICATION EXAMPLE

Although the simulation of deformable tissues can be used in a wide field of surgical simulation[1], the main field of application is the virtual laparoscopy also known as keyhole surgery. On the one hand, force feedback devices for virtual laparoscopy already exists and on the other hand, keyhole surgery requires much more training as conventional surgery.

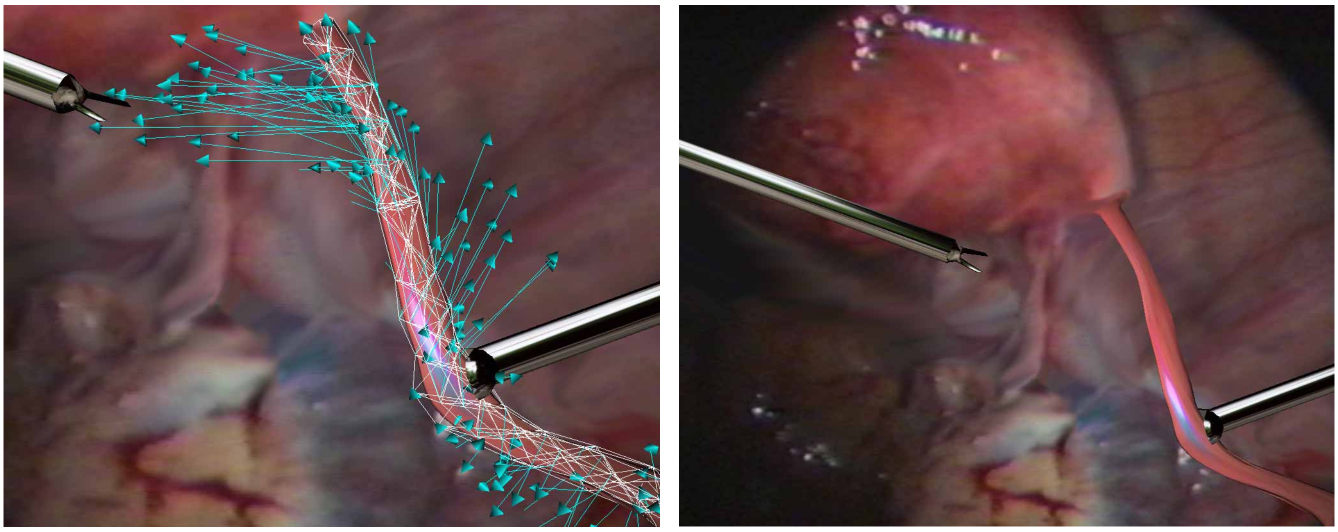
Figure 7: Example of virtual laparoscopy: On the left side the actual force vectors at the mesh nodes are shown. On the right side, the resulting deformation in the virtual environment is presented.

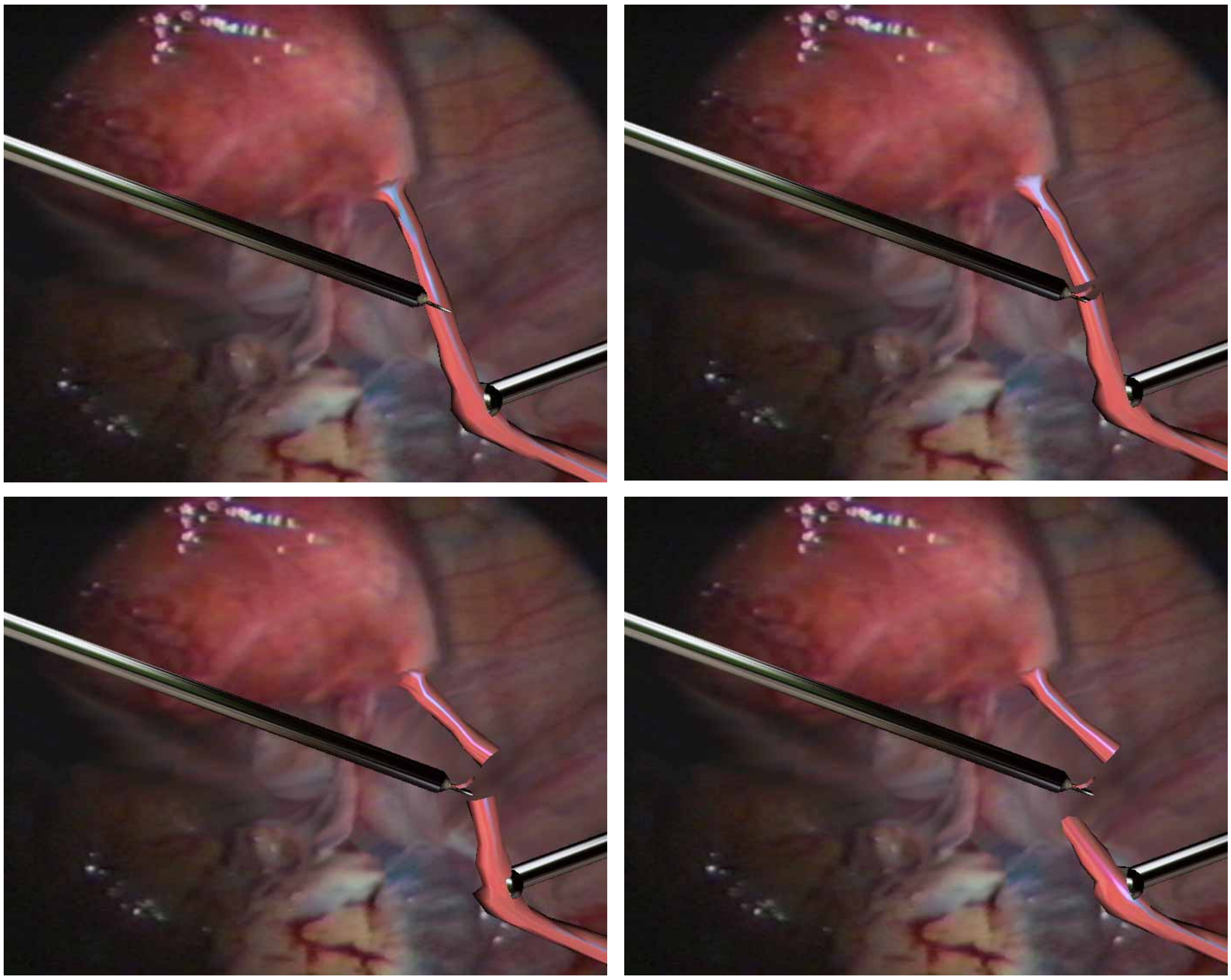
Figure 8. A virtual sterilization: The Fallopian tube is cut with scissors.

We choose the gynaecological laparoscopy for a first experimental simulator. The simulator is written in C++ and is executable under Windows NT and Unix. The graphical interface is based on Open Inventor and uses many features to improve the graphic level of reality. We are currently using a Silicon Graphics Onyx2 Infinite Reality for high-end visualization. A simpler version can be executed on a standard PC.

Figure 7 shows a virtual simulation environment in gynaecological laparoscopy with the uterus at the upper left side of the right picture and the Fallopian tube at the lower right side. Two endoscopic instruments are simulated: scissors and a forceps. For a sterilization, only the Fallopian tube must be simulated with an elastic mesh, because it is the only manipulated organ.

The Fallopian tube is build by use of 160 nodes and 610 springs (see Figure 1 *(b)*). The elasto-dynamic properties of the Fallopian tube are described with the terms “soft hardness”, “big elasticity” and “low mass”. For the presented application, we used NURBS (Non-Uniform Rational B-Spline) interpolation to obtain a smooth surface (for details see, for example, Ref. 3). The other organs are modeled conventionally.

On an Onyx2 the simulation can be done with 40 frames per second using only one processor, although a SGI O2 is sufficient for the simulation without NURBS. The left part of Figure 7 represents the internal force vectors for every node of the mesh. The force vectors are resulting from the change of the mesh node’s positions caused by the forces of the laparoscopic devices. In the right part of Figure 7 an endoscopic view at the operating scenery is presented.

Figure 8 shows the virtual sterilization. This time, the Fallopian tube is drawn without NURBS. The Fallopian tube is cut and the loose endings are retracted because of a small prestressing. To complete the operation procedure the loose endings must be tied together with a sling. Currently we are working on simulating the slings and building knots.

5. CONCLUSIONS AND FUTURE WORK

In section 4 we have shown that the developed model can be used successfully for the simulation of deformable objects e.g. tissues in surgical simulation. The simulation of small organs like the Fallopian tube in section 4 can be done even on low-cost hardware (for example Pentium PC’s). The elastic simulation is faster as the graphical computation so only for an enhanced graphic representation high-end hardware is needed.

Since the presented propagation algorithm can be modified to be executed on multiprocessor systems, even larger objects can be simulated on standard hardware. Currently we are working on a connection of volumetric meshes and volume rendering to be able to simulate cuts more realistic by use of solid organs.

By use of the presented model, it is easy to apply external forces to any node, for example, gravity forces or collision forces caused by medical tools (see Figure 7). Furthermore, changes of the object’s structure can be done during simulation, for example changes caused by cuts, ruptures or fractions. The weights of the network can be initialized by real mass and spring parameters. Besides, these parameters can be adapted or learned by use of a physical model or measured data of real objects.

Nevertheless, the propagation procedure has to be analyzed in more detail, to be able to evaluate the quality and stability of the propagation process. We are currently studying some improvements for the propagation procedure.

Besides, we are working on improvements for the learning procedure. One of the objectives of our project is to develop a complete method, which enables us to measure the elasto-dynamic behavior of real tissues and to generate a simulation model of this tissue, by use of the measured data. At present, we are only able to learn the behavior of simple deformable objects.

Deformable tissues improve the quality of surgical simulators. However, without being able to simulate whole surgical procedures, the presented application can not be used for training purposes. For example, the simulation of slings, knots, and irrigations is necessary, because these procedures require the most training for prospective surgeons. We will include these procedures in our experimental simulator in the near future, to test it for surgical training with medical professions.

Up-to-date information concerning this project can be obtained via the Internet from http://www.umi.cs.tu-bs.de/full/ger_index_arne.html or http://fuzzy.cs.uni-magdeburg.de/~nuernb.

REFERENCES

1. C. Basdogan, P. Loan, J. M. Rosen et al, “An interactive model of the human thigh for simulating surgical procedures in virtual environments,” in *Proc. of the Winter Annual Meeting of ASME (BED-Vol. 33)*, pp.439-440, Atlanta, 1996
2. M. Bro-Nielsen, “Surgery Simulation Using Fast Finite Elements,” In *Visualization in Biomedical Computing (Vol. 1131 of Lecture Notes in Computer Science)*, pp. 529-534, Springer, Berlin, Heidelberg et al., 1996
3. G. E. Farin, *NURB Curves and Surfaces*, A. K. Peters Ltd., Wellesley, Massachusetts, 1995
4. P. L. Gould, *Introduction to Linear Elasticity*, Springer, New York, 1994
5. S. Hayken, *Neural Networks*, Prentice-Hall Inc., New Jersey, London, et al., 1994
6. J. J. Hopfield, “Neural networks and physical systems with emergent collective computational abilities,” in *Proc. Nat. Acad. Sci. (Vol. 79)*, pp. 2554-2558, USA, 1982
7. G. J. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic: Theory and Applications*, Prentice-Hall Inc., New Jersey, London et al., 1995
8. R. Kruse, J. Gebhardt and F. Klawonn, *Foundations of Fuzzy Systems*. John Wiley & Sons, Inc., New York, Chichester, et al., 1994
9. C. T. Lin and C. S. G. Lee, *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems.* Prentice-Hall Inc., New Jersey, London, et al., 1996
10. T. C. W. Mak and G.-D. Zhou, *Crystallography in modern chemistry : a resource book of crystal structures*. John Wiley & Sons, Inc., New York, Chichester, et al., 1992
11. D. Nauck, F. Klawonn and R. Kruse, *Foundations of Neuro-Fuzzy Systems*, John Wiley & Sons, Inc., New York, Chichester, et al., 1997
12. F. J. Pineda, “Generalization of back-propagation to recurrent neural networks,” *Physical Review Letters*, **59(19)**, pp. 2229-2232, 1987
13. F. J. Pineda, “Dynamics and Architecture for Neural Computation,” *J. Complexity*, **4**, pp. 216-245, 1988
14. F. J. Pineda, “Recurrent back-propagation and the dynamical approach to adaptive neural computation,” *Neural Computation*, **1**, pp. 161-172, 1989
15. A. Radetzky, A. Nürnberger and D. P. Pretschner, “A Neuro-Fuzzy Approach for the Description and Simulation of Elastic Tissues,” in *Proc. of the European Workshop on Multimedia Technology in Medical Training*, pp. 30-40, Aachen, Germany, 1997
16. Z. Wesolowski (ed), *Nonlinear Dynamics of Elastic Bodies*, Springer, Wien, 1978
17. R. J. Williams and D. Zipser, “A learning algorithm for continually running fully recurrent neural networks,” *Neural Computation*, **1**, pp. 270-280, 1989
18. R. J. Williams and D. Zipser, “Experimental analysis of the real time recurrent learning algorithm,” *Connection Science*, **1**, pp. 87-111, 1989