

## Soft Tissue Deformation and Optimized Data Structures for Mass Spring Methods

T. Halic<sup>1</sup>  
Mechanical,  
Aerospace and  
Nuclear  
Engineering  
RPI,  
[halict@rpi.edu](mailto:halict@rpi.edu)

S. Kockara<sup>1</sup>  
Computer  
Science, UCA,  
[skockara@uca.edu](mailto:skockara@uca.edu)

C. Bayrak  
Computer  
Science,  
UALR,  
[cbayrak@ualr.edu](mailto:cbayrak@ualr.edu)

R. Rowe  
Neurosurgery,  
UAMS,  
[RoweRichard@uams.edu](mailto:RoweRichard@uams.edu)

B. Chen  
Computer  
Science, UCA,  
[bchen@uca.edu](mailto:bchen@uca.edu)

### Abstract

*One of the essential components of a virtual reality surgical simulation is deformation. Deformations in computer graphics and surgical simulations are commonly modeled with three different approaches e.g. geometry based methods, Finite Element method (FEM), and Mass-spring Method (MSM). The geometry based methods are quite fast and visually appealing. Late two methods take the physics of deformation into consideration. Even though the FEM results in more physically realistic deformations, a significant drawback of this method is its expensive computation cost and vulnerability to surgical procedures such as incision. However, MSM is relatively computationally inexpensive. Because of the real-time physics based behavior of MSM, it is widely accepted by surgical simulation community. This study summarizes deformation simulation methods and their pros and cons for surgical simulations. Moreover, because of wide usage of the MSM, optimized data structures for MSM are provided and analyzed under different deformation settings.*

### 1. Introduction

Advances in use of information technologies in medicine and biology such as image processing and virtual environment based simulations are providing new opportunities for biologists and medical professionals. New researches in visualizations and virtual environments are opening the door to a new paradigm for practicing biology and medicine that promises to transform healthcare education system. Individualized medical training by virtual simulations and the use of advanced visualization

techniques in medicine will impact the way medicine is practiced.

Surgery simulations in virtual domain have promising potential to train physicians. The main goals in these simulations are reducing the costs and risks in surgical training. In general, such a simulation displays the model of human tissue on the computer monitor and simulates human spatial perception in virtual domain via simulating fluids, motions, contacts and responses, and deformations of the tissue. Nevertheless, in order to replace the traditional training with Virtual Reality (VR) technology, medical simulators should have acceptable realism that allows physicians to feel that they are truly in a surgery environment. One of the most challenging issues in the VR medical simulator is the deformation of computer-generated tissue geometry in real-time.

In traditional training, cadavers, animals, or mannequins are used. However, finding a cadaver is not easy, and is rather expensive. Also, working on and getting trained with dead tissues on a cadaver can provide a degree of structural understanding but lack in the functional behavior of real organs or tissues. Thus, without operating on a real patient, a medical student or surgeon is not only unaware of the fundamental characteristics but also the environment of the operating room. On the contrary, due to the real life emulation support of the VR, the deficiencies related to unrealistic details of the traditional approach can be replaced with the high-end state of the art graphic rendering hardware to display not only the geometry of the computer generated organs or tissues but also the real sensation through the use of haptic force-feedback devices.

Imitation of soft-tissue behaviors to an applied force is known as soft-tissue deformation simulation. That is one of the most important components towards a surgical simulation. In this work, in

<sup>1</sup> First two authors – Tansel Halic and Sinan Kockara- have made equal contributions to this study.

addition to fundamentals of a surgical simulation, soft tissue deformation simulation methods are summarized according to their advantages and disadvantages. Moreover, optimized data structures for one of the soft tissue deformation simulation methods –mass spring method- are scrutinized and analyzed in order to demonstrate soft-tissue behavior. The analysis hereby gives us an intuitive and quantitative assessment of the performance of our data structures for MSM. This analysis on developed data structures can help MSM users to select the most efficient data structure for their soft-tissue simulations.

## 2. Background

Regarding medical training, deficiencies of current traditional education methods yield the VR simulators as subtle instruments in medical training. Impact of medical training is well understood when the report of National Library of Medicine (NLM) is taken into consideration that each year as many as 98,000 people in hospitals die due to medical errors some of which are results of surgical procedures or tests [26]. This report reveals that insufficiency of medical training and visualization could cost human lives. For that reason, National priority in healthcare delivery is to reduce errors and deaths. Hence, surgical proficiency is a must and the success of surgical training for proficiency can be provided/obtained by hands-on experience. Current classical training means are employed by practicing on cadavers, using laboratory animals, or operating on real-patients under the supervision of more experienced doctors. In fact, acquisition of cadaver is difficult and very costly; moreover, cadaver conditions generally don't match real tissues. Besides, it is not a feasible way to train and increase the rapid decision making of physicians for complex surgical operations. On the other hand, usage of animals gives great opportunity to comprehend how a real-surgery looks like; however, anatomical differences decrease usefulness of this method and make it impractical. Another training method is operating on a real patient with the supervision of an experienced professional, but this has a high-risk environment and it also may not be probable to train students for complex cases. Furthermore, when training surgeons for rarely seen or more complex scenarios under certain conditions are required, aforementioned traditional training methods may not be appropriate or sufficient solutions. Henceforth, the VR simulators remove most of these deficiencies of classical methods especially human risk factors. The VR simulators have potential to provide a safer, cost-effective, portable, and complementary method to the traditional education techniques, and a candidate of an upcoming training technology [27][28].

In VR simulators; however, soft tissue deformations and collision processing are the essential components towards the development of a simulator. In order to achieve the realism, displaying the physical behaviors and interactions of organs and tissues are indispensable. Besides, realistic simulation of tissue behaviors increases the invasiveness of medical professionals in virtual environment. In addition, supporting user interactivities by force feedback devices enhance plausibility and effectiveness of the simulator. In this regard, simulation of tissues realistically will mandate researchers to find physically and mathematically correct models for both deformation simulation and collision processing.

Soft tissues have complex mechanical nature. Unlike many materials, they exhibit nonlinear stress and strain relationships and may be deformed 100% which unlikely exist in common engineering materials. Soft tissues exhibit viscoelastic, anisotropic, and inhomogeneous behavior. Therefore, the differential equations of the physical phenomena turn out to have a non-linear nature. In addition to complexity of mathematical models, the general properties of tissue could not be derived since properties of tissues vary among even for the same person in different time intervals. The lack of experimental data of human tissue characteristics and stress-strain relationship makes the problem more challenging. Likewise, surgical operations (e.g. incision, tearing, suturing etc. [29]-[32]) that are common in a surgery will likely to change the physical property of a tissue so that tissue behavior will be altered [29].

There exists four common ways in tissue modeling and deformations: vertex based, spline based, particle based, and FEM based methods [1]. In this research, particle based MSM method is our primary concern because of its simplicity, plausibility, and wide usage. The latter two approaches (FEM and MSM) focus on physically based modeling and implementation. The advantages of geometry based (vertex and spline) approaches are: fast and smooth but the major drawback is in their accuracy [2]. On the contrary, FEM [3] based methods are very accurate and realistic but they are highly comprehensive, complicated, and computation intensive, that makes FEM implementation challenging. However, there is an ongoing research attempting to solve displacement vectors faster by removing unwanted degrees of freedom by performing condensation of stiffness matrix. That approach is known as Fast FEM [4]. In Fast FEM approach, nodes' displacements that are not in focus of interest, e.g. internal nodes that are not likely to be deformed, are removed from the solution set. As a result, computation time is reduced. Nonetheless, it is still a requirement to rebuild the stiffness matrix when geometric topology

changes. This happens during the surgical operations such as suturing, incision, and tearing that often occurs in every surgery [5]. These problems impose performance difficulties and lower the Frames Per Second (FPS) ratio of graphics scene and reduces the usability of FEM especially when the computation power is limited. When the graphics scene with high frame rates is taken into consideration, the time required for the calculation of deformation solution should not be higher than a certain threshold. All these advantages and disadvantages of FEM are summarized in Table 1.

**Table 1 Pros and Cons of FEM**

FINITE ELEMENT METHOD	
Advantages	Disadvantages
Accurate and Valid	Slow
Visually realistic	Depend on the mesh structure. Well structured mesh is crucial
Easy to experiment different material properties	Needs more memory
Open to improvement such as (i.e. condensation, meshless methods)	Pre-computation takes time
	hard to change mesh structure in real-time
	hard to apply cutting and tearing operations

The surgical invasiveness can be increased by integration of the haptic tools which complement the interactions. However, tactile force feedback tools promote synchronization problem; synchronization of graphics scene with haptic threads. This problem occurs since haptic rendering is much faster than the scene rendering. In a typical surgical simulation scene mainly includes deformations, collisions, and fluid dynamics concurrently. These compute intensive tasks need to be calculated simultaneously. Thus, even minute performance gain is very important for the simulation's plausibility since human eye is more perceptive to flaw than physical correctness of the simulation. Moreover, the realistic tissue or organ models for visualization generally consist of huge number of polygons which accommodates the delicate details. Thus, the tissue models impose trade-off between the realism and accuracy of methods. In conclusion, developed algorithms for both deformation simulation and collision detection should satisfy the required features of VR simulators such as realism and speed. Therefore, the overall system compels developer to devise ways to find optimized but also acceptable algorithms for tissue deformations and collision detections.

## 2.1. Collision Detection

One of the main bottlenecks in the real-time rendering loop is conventional collision detection. Therefore, improvements in this area are of a great interest and plenty of researches have been done in the field. Main concern of the collision detection algorithms is how to get the best return from the speed, accuracy, and collision response trade-offs. To overcome these problems many collision detection algorithms developed and mainly classified as feature-oriented algorithms [9], simplex-oriented algorithms [10], and bounding volume hierarchies [11]-[17] that are summarized in surveys [18] and [19].

The recent developments of programmable graphics hardware i.e. the introduction of floating point textures, increased programmability of vertex and pixel shaders; has generated a lot of interest in using graphical processing unit's (GPU) excessive capacity for more general tasks rather than rendering. Since the GPUs are allowing rendering tasks to be parallelized across many deeply pipelined computational units, this parallelism gives an overall speed advantage to the GPUs. By using this advantage of the graphics hardware for other computation purposes, it is possible to move collision detection operations from CPU to the GPU and to reduce CPU's computation load for being able to use by other computations.

There are several attempts to use graphics hardware for collision detection operations [20]-[23]. All these image-spaced-based techniques rely on rasterizing the objects of a collision query into color, depth, or stencil buffers and determines whether the objects in the virtual scene are colliding or not. In this research, collision detection algorithm Cinder [23] based on image-spaced-base technique - handling both convex and non-convex geometries- is used. The tests for collisions are performed in image space. The algorithm does not require any pre-processing or special data structures. We use frame buffer operations for every pixel to implement a virtual ray casting algorithm that detects interferences among objects. The edges of the objects are written to the depth buffer and the objects that penetrated each other are detected by using virtual ray-casting. Virtual ray-casting is a technique that locates a point relative to an object by casting a ray from the point. The number of polygons that the ray passes through is counted in such a way that if summation result for one ray is even then that means the point is outside the object. In contrast, if the summation result is un-even then the point is inside the object and there is collision. This is illustrated in the Figure 1.

We use a stencil buffer for counting the number of

front and back facing polygons that the rays pass through. The values in the stencil buffer are increased for front-facing polygons and decreased for back-facing polygons. If at the end there is non-zero value in the stencil buffer then edge in the specific pixel is inside an object (meaning more front-facing polygons than back-facing ones). Colliding objects' identifications' numbers kept in color-buffer. The algorithm's running time is linear in the number of objects and the number of polygons existing in the objects. With this algorithm collisions that are about to happen or have already occurred will not be detected. This occurs when objects' very thin parts pass through each other in one frame interval. To overcome this problem, thickness of objects can be increased.

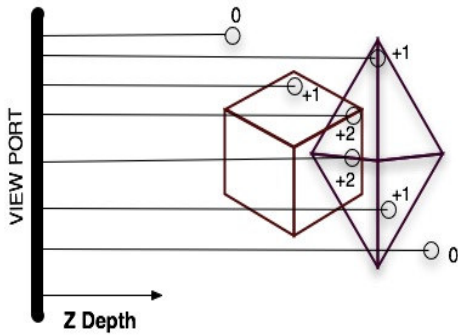


Figure 1. Virtual semi infinite ray casting for collision detection

### 3. Mass Spring Method

Even though for MSM, it is difficult to extract parameters from experiments for the thousands of individual springs and masses or dampers, because of its nice properties (e.g. simplicity, being physics-based, and wide usage) MSM is used in this research. To have both real-time performance and an acceptable deformation output, MSM is common choice for deformation simulations. Properties of MSM are summuned in Table 2.

In this model, body is assumed to be an object that is composed of discrete mass particles. Masses are generally assigned to vertices and are connected to other masses with springs as depicted in Figure 2. In MSM model, the geometry is discretized into masses which are connected to each other with the governing equation called Lagrange equation of motion [6].

$$M_i \ddot{x} + \delta_i P \dot{x} + K_i x_i = F_{extern} \quad (1)$$

Table 2 Pros and Cons of MSM

MASS SPRING MODELS	
Advantages	Disadvantages

Fast and easy to implement	Weak in preserving mesh volume
Initialization is not required	Stability is dependent on time step
Change in the mesh topology can be handled easily	Hard to integrate tissue properties into the system.
Suitable for parallel computation.	Adding constraints to the system is troublesome. For instance; modeling the incompressible volumetric objects
	Hard to validate the deformation

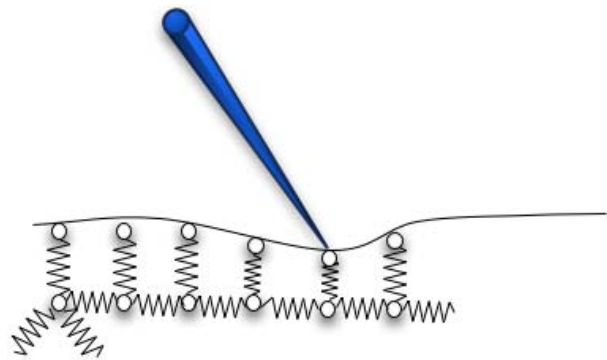


Figure 2. Vertices are connected to each other with spring elements.

One of the fundamental problems with MSM is finding the corresponding stiffness parameters for a particular tissue [7]. Besides, in MSM; discretization does not completely represent or approximate the full body. In order to overcome this drawback we increase the nodes connectivity by applying restoring forces and adding damping for energy dissipation [8].

MSM allows simulating the tissue physics and behaviors at a high frequency update rates, and allows medical simulators to have real-time interactivity. In this model, the soft tissue with triangular mesh structure is thought of as discrete point masses, which generally are located at the vertices' positions, are connected to each other with linear springs. Therefore, employing force to a particle mass within the spring mesh will move the particular node and propagate the forces across its neighbours. Formulation of the model is given with the following formula.

In this formulation  $a_i^t$  denotes the acceleration of the particle at time t.  $N(i)$  is the list of neighbours of  $i$ ,  $k_{ij}$  is the stiffness of the spring that connects  $i$  to  $j$ .  $d_{ij}$  is the unit vector between the particle  $i$  and  $j$ .  $l_{ij}^t$  is the length of the spring at time t between particles

$$m_i a_i^t = \sum_{j \in N(i)} k_{ij} d_{ij}^t (l_{ij}^t - l_{ij}^0) - \gamma_i v_i^t + f_{ext}^t$$

$i$  and  $j$ .  $l_{ij}^0$  indicates initial length of the spring  $ij$ .  $\gamma_i$  represents damping coefficient,  $v_i^t$  is the velocity of particle  $i$  at time  $t$  and external force is  $f_{ext}^t$ . In the equation,  $\sum_{j \in N(i)} k_{ij} d_{ij}^t (l_{ij}^t - l_{ij}^0)$  term denotes the spring force applied on node  $i$ . The same force with an opposite direction is applied to node  $j$  during the iteration.

In each rendering frame, solution of the differential equation should be obtained and the new displacements of each particle should be passed to the graphics engine. Solution to the differential equation is obtained with the Verlet Integration. In the Verlet Integration, the new position of a particle for next time step is determined using the previous position of the particle and its acceleration. So, the equation turn into the following.

$$x(t+h) = x(t) + (x(t) - x(t-h)) * damping + a * h * h$$

In this equation  $x(t+h)$  is the vertex position that we are interested.  $x(t-h)$  denotes the previous position ( $h$  indicating time step). Damping coefficient is for energy dissipation which arises due to friction or any other forces. Damping serves to increase both stability and realism of the system

The Verlet Integration is chosen in our model on account of its efficiency in storage. In the simulation, only the previous positions of each particle are stored. Acceleration is computed in every frame. So, the previous position for each particle is not allocated, that saves memory.

## 4. Optimized Data Structures for MSM

In the simulation, for storing the vertex connections, three different data structures are employed and compared with each other. These are efficient storage, fast link, and dynamic bulk allocation structures.

### 4.1. Efficient Storage Structure

In this structure, every spring connection among the nodes is allocated dynamically. For this purpose, a linked list structure is developed. A new node is inserted in the linked list when a new connection will be created. The total amount of memory spent is the same as the number of springs of the total mesh. This structure is very efficient for the system that has memory limitations. Also, it provides flexibility for the cases where springs are frequently removed and inserted. The major drawback of this approach is the memory access overhead that dramatically lowers its performance. However, the use of multi-threaded approaches and buffering scheme would reduce the

performance bottlenecks. The schematic representation of this structure is given in Figure 3.

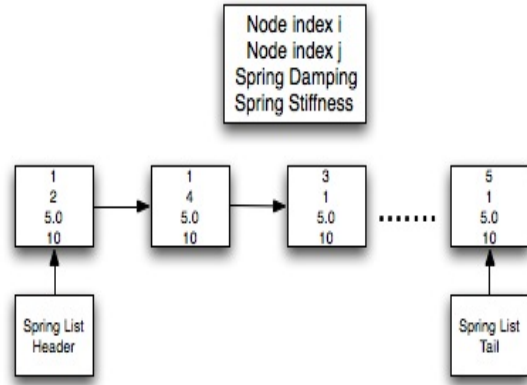


Figure 3. Efficient Storage Structure

### 4.2. Fast Link Structure

This structure is the fastest approach among all. However, it trades off the flexibility and memory efficiency. In this structure, average number of springs per vertex is defined before beginning of the simulation. The total size of an array, therefore, is equal to  $N * AVGNumberofSprings$ . The number of springs required is dynamically allocated from the memory. This fixed-length array limits the total number of springs exists in the simulation. The major drawback is the implementation problems of adaptive approaches such as refining the mesh structure, or creating large implicit connections that increases the number of springs during the simulation. The schematic representation is given in Figure 4.

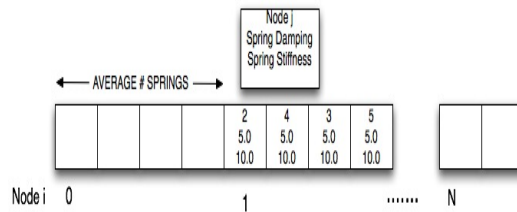


Figure 4. Fast Link Structure

Each node in the array (see Figure 4) has the spring connections of itself. Accessing a particular node's springs becomes straightforward without iterating over the pointer lists as in the previous scheme. Thus, a vertex's springs are started at  $(i-1) * AverageNumberofSprings$  in the array.

### 4.3. Dynamic Bulk Allocation

This structure combines aforementioned two data structures. Pointer array with size N (vertical array denotes this array in Figure 5) is defined for storing the beginning of the spring arrays. During the simulation if the capacity runs out, the new bulk array is added to the end of the array list of the vertex. This provides a dynamic extension without copy operations that cause excessive memory accesses. In this approach, performance is worse than the fast link structure but better than the efficient storage structure. However, it saves the unused memory. At the worst case, this approach uses  $N*(AVG\text{NumberofSprings}-1)*\text{SizeOfOneArray}$  bytes.

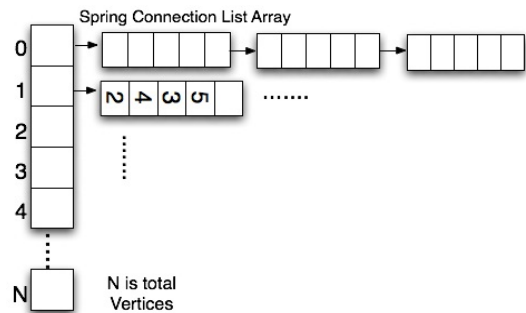


Figure 5. Dynamic Bulk Allocation

Figure 6 shows deformed version of a cloth with the use of dynamic bulk allocation data structures. The tip of the instrument has yellow sphere that illustrates contact region which is detected by image-spaced-based algorithm. Force applied to the cloth causes sinking effect on the cloth.

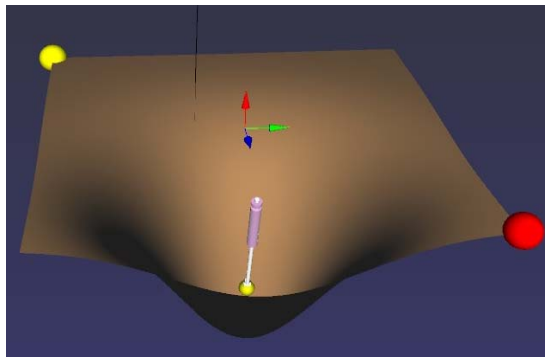


Figure 6. Cloth deformation

## 5. Results

Three data structures' performances are tested and compared with different number of nodes on a laptop PC with 1GB ram and Intel Centrino Duo 2 1,83 MHz CPU. In the performance tests, fast link structure outperforms the other two structures in computation times. From a memory allocation point of view, efficient storage structure outperforms the

other two structures. The tests point out that the performance difference is more noticeable when the number of nodes reaches one million nodes.

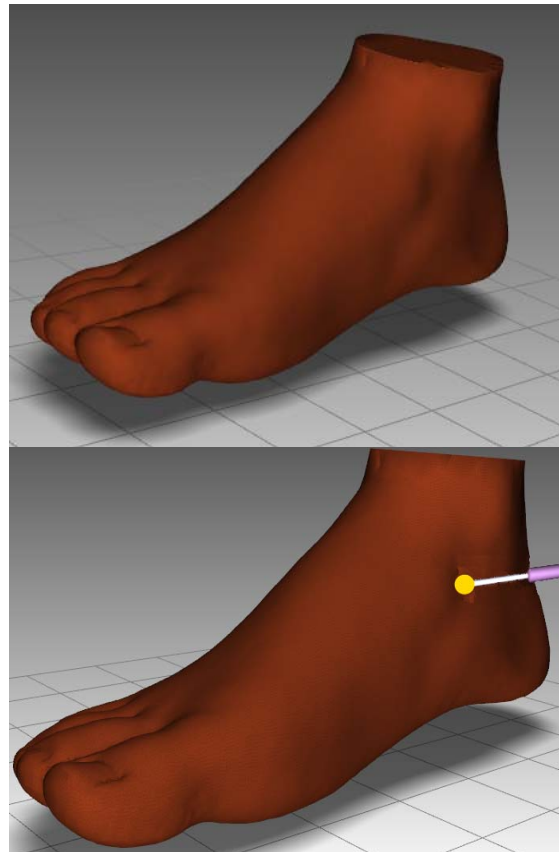


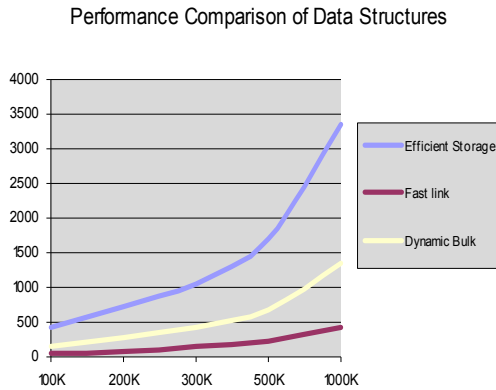
Figure 7. Non-deformed and deformed soft tissue

From the given Table 1 and Figure 8 that indicate computation times in milliseconds per number of nodes, fast link data structure apparently behaves linearly during the performance test. Regardless of total number of elements in the fast link array structure, the total time for accessing the nodes would be the same. After increasing the number of nodes more than 500K fast link structure indicates a non-linear increase due to the sudden increase in the paging within virtual memory management system of operating system. On the contrary, from the performance point of view, dynamic bulk storage scheme has performance combination of both structures. It is non-linear in a large number of nodes and linear in fewer nodes.

Table 3 Comparisons by Milliseconds Per Number of Nodes

#of nodes	Efficient Storage	Fast link	Dynamic Bulk
100K	422	47	141

200K	735	78	281
300K	1047	141	437
500K	1703	219	687
1000K	3344	422	1344
2000K	89625	859	30796



**Figure 8. Comparisons of data structures, number of vertices vs computation times (ms)**

## Acknowledgement

We would like to thank NVIDIA, especially Dr. David Luebke from NVIDIA, for their partnership in continuous support.

## 6. References

- [1] Al-khalifah, D Roberts “Survey of Modelling Approaches for Medical Simulators”, Proceedings 5<sup>th</sup> Intl Conf. Disability, Virtual Reality & Assoc. Tech., Oxford, UK, 2004.
- [2] H. Delingette “Towards Realistic Soft Tissue Modeling in Medical Simulation”, Proceedings of the IEEE Special issue on Surgery Simulation, pp. 512-523, April 1998.
- [3] M. Bro-Nielsen and S. Cotin. “Real-time Volumetric Deformable Models for Surgery Simulation using Finite Element Methods and Condensation” In Eurographics, pages 21-30, 1996.
- [4] M. Bro-Nielsen, Fast Finite Elements for Surgery Simulation *Studies in Health Technological Information*, vol. 39, pp. 395-400, 1997
- [5] Basdogan, C., “Real-time Simulation of Dynamically Deformable Finite Element Models Using Modal Analysis and Spectral Lanczos Decomposition Methods ”, Proceedings of the Medicine Meets Virtual Reality (MMVR’2001) Conference, pp. 46-52, Irvine, CA, 2001.
- [6] S Gibson and B Mirtich, “A Survey of Deformable Modelling in Computer Graphics”, TR-97-19, Mitsubishi Electric Research Laboratory, Cambridge, MA, USA., 1997.
- [7] Y. Bhasin and A. Liu, “Bounds for Damping that Guarantee Stability in Mass-Spring Systems”, MMVR, 2006
- [8] Colorado University “Real-Time Visually and Haptically Accurate Surgical Simulation”, Centre for Human Simulation, Electronic version: <http://www.uchsc.edu/sm/chs/research/mmvr.html>, 2004.
- [9] M. C. Lin and J. F. Canny, “A Fast Algorithm for Incremental Distance Calculation”, IEEE Conference on Robotics and Automation, pp. 1008-1014, 1991.
- [10] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, “A Fast procedure for Computing the Distance Between Complex Objects in Three-Dimensional Space”, IEEE Transactions on Robotics and Automation, vol. 4, no. 2, pp. 193-203, April 1988.
- [11] S. Quinlan, “Efficient Distance Computation between Non-convex Objects”. In *Proc. IEEE Int. Conf. On Robotics and Automation*, pp. 3324–3329, 1994.
- [12] P. M. Hubbard, “Approximating Polyhedra with Spheres for Time-critical Collision Detection”. *ACM Transactions on Graphics (TOG)* 15, 3, pp. 179–210, July 1996.
- [13] Van Den Bergen, “Efficient Collision Detection of Complex Deformable Models using AABB Trees”. *Journal of Graphics Tools*, 2, 4, pp. 1–14, 1997.
- [14] R. Webb R., M. Gigante, “Using Dynamic Bounding Volume Hierarchies to Improve Efficiency of Rigid Body Simulations”. In *Proceedings of the 10th International Conference of the Computer Graphics Society on Visual computing : integrating computer graphics with computer vision*, Springer-Verlag New York, Inc., pp. 825–842, 1992.
- [15] S. Gottschalk, M. C. Lin, D. Manocha, “OBBTree: A Hierarchical Structure for Rapid Interference Detection”. In *Computer Graphics (SIGGRAPH '96 Proceedings)* pp. 171–180, 1996.
- [16] S. Redon, A. Kheddar, S. Coquillart, “Fast Continuous Collision Detection between Rigid Bodies”. *Computer Graphics Forum* 21, 3, pp. 279–288, September 2002.
- [17] G. Ziegmann, “Rapid Collision Detection by Dynamically Aligned DOP-trees”. In *Proceedings of the IEEE Virtual Reality Annual International Symposium*, pp. 90–97, March 1998.
- [18] M. C. Lin and Gottschalk, “Collision Detection Between Geometric Models: A Survey”, 1998.
- [19] P. Jimenez, F. Thomas, and C. Torras, “3D Collision Detection: A Survey”, *Computers and Graphics*, 25(2):pp. 269-285, 2001.

- [20] N. K. Govindaraju, M. C. Lin, D. Manocha, "Quick-Cullide: Fast Inter- and Intra-object Collision Culling Using Graphics Hardware". In *IEEE VR*, 2005.
- [21] M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M. P. Cani, F. Faure., N. Magnenat-Thalmann, W. Strasser, P. Volino, "Collision Detection for Deformable Objects". *Computer Graphics Forum 24*, pp. 61–81, 2005.
- [22] N. K. Govindaraju, S. Redon., M. C. Lin., D. Manocha, "CULLIDE: Interactive Collision Detection between Complex Models in Large Environments using Graphics Hardware". In *Graphics Hardware 2003*, Eurographics Association, pp. 25–32, 2003.
- [23] D. Knott, D. Pai, "Cinder: Collision and Interference Detection in Real-time Using Graphics Hardware". In *Proc. of Graphics Interface '03* 2003.
- [24] S. Cotin, H. Delingette, and N. Ayache, Real-time Elastic Deformations of Soft Tissues for Surgery Simulation, *IEEE Transactions On Visualization and Computer Graphics*, 5, 1, pp.62-73, 1999.
- [25] S. Gibson and B. Mirtich. "A Survey of Deformable Modeling in Computer Graphics", Tech. Report No. TR-97-19, Mitsubishi Electric Research Lab., Cambridge, MA, November 1997.
- [26] National Institute of Medicine, "To Err is Human", November 1999.
- [27] Scerbo, MW, "Medical virtual reality simulation. Enhancing safety through practicing medicine without patients", *Biomedical instrumentation & Technology*, 38, pp: 225-228, June 2004.
- [28] Scerbo, M.W, "Medical Virtual Reality Simulators Have we missed an opportunity", *Human Factors and Ergonomics Society*, vol. 48, no. 5, May 2005.
- [29] Chanthasopeephan, T., Desai, J.P., Lau, A. C. W., "Study of Soft Tissue Cutting Forces and Cutting Speeds", In *Medicine Meets Virtual Reality Conference (MMVR12)*, Newport Beach California, IOS Press pp: 56–62, January 2004.
- [30] Liu, A., Kaufmann, C., Daigo T., "An architecture for Simulating Needle-Based Surgical Procedures," *Proceedings of the 4th International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp: 1137-1144, October 2001.
- [31] DiMaio, S. P., Salcudean, S. E., "Needle insertion Modelling For the Interactive Simulation of Percutaneous Procedures", In *Medical Image Computing and Computer-Assisted Intervention, MICCAI 2002* no. 2489 in *lecture Notes in Computer Science*, Tokyo Japan, pp:253–260, September 2002.
- [32] Ursino, M., Tasto, P. D. J. L., Nguyen, B. H., Cunningham, R., Merrill, G. L., "CathSim: An InTravascular Catheterization Simulator on a PC", *Medicine Meets Virtual Reality, Convergence of Physical and Informational Technologies: Options for a New Era in Healthcare*, pp:360-366, 1999.