Intelligent Teleoperation: combining simulation and teleoperation

Paolo Fiorini , Debora Botturi
Department of Computer Science
University of Verona
Strada le Grazie, 15 -- 37134 Verona, Italy
Email: {fiorini,debora}@metropolis.sci.univr.it

Maarja Kruusmaa
Center for Biorobotics
Tallinn University of Technology
Akadeemia tee 15A-111, 12618 Tallinn, Estonia
Email: maarja.kruusmaa@biorobotics.ttu.ee

ABSTRACT

In this paper we summarize some of our research efforts in the area of high performance simulation to support advanced teleoperation. Simulation of rigid bodies has always been used in conjunction of teleoperation system, however simulation of deformable bodies is not available yet, since it presents a new set of challenges. In particular, we address the problems related to object modeling to achieve force feedback, and the computation infrastructure necessary to carry out the force computation at real time frequency. We describe the results achieved by combining mesh-based models with GPU-based computation. The result is a very realistic and performing simulator that combines rigid and deformable objects into a single computational framework. Furthermore, planning in a deformable environment must account for the dynamic properties of the object together with classical algorithms. The simulator and planner are embedded into our CORBA-based teleoperation system that allows an easily integration of different hardware and software functions. Examples showing the capabilities and performance of the complete system are given for the context of robot-assisted surgery, which presents a number of similarities to robots for space applications.

INTRODUCTION

In the next future, new manned exploration missions are planned, especially to the Moon, where a human outpost may be established. The work to build such an outpost will be probably carried out by various types of robotic devices. It is very unlikely that planetary construction robots can be endowed of full autonomy, because of the many uncertainties of the environment and the difficulties in autonomously controlling a complex task. Teleoperation will probably be the primary commanding mode of the robots. However, considering time delay and uncertainties, the operators will have a difficult time to precisely controlling the robot motions, and may run the risk of damaging some of the equipment. Thus the need of providing the operators with commanding help, in the form of visual and haptic aids, that could help them move safely the robots. Furthermore, some of the elements that teleoperation will have to deal with, may be made of flexible materials, such as inflatable structures, astronauts suites, and protective covers, whose position and properties are difficult to estimate and predict. In the past, simulations with “phantom robots”, i.e. moving without time delay, have been used as aids to teleoperation, but they did not include deformable parts. In fact, flexible materials and deformable bodies are rather difficult to model and, in particular, they need specific methods for the computation of contact forces that are required in haptic simulations.

Two main classes of deformable models have gained importance due to their good characteristics: Finite Element Models (FEM) and mass spring models (MSM). FEM’s are very easy to calibrate, since they represent the discretization of a function over a domain, and therefore it is possible to use the characteristic equation of the material to define their behavior. They can be computationally fast, but the velocity comes at the cost of the interaction types that can be simulated behavior, in fact they usually do not allow topology changes in the mesh. Although less precise and more difficult to calibrate, mass spring models allow changes in their topology at almost no cost, and they are suitable for the implementation on graphic processing unit. Given the topology, a mass spring model is univocally identified by the value of the mass associated to each node of the mesh and by the elastic coefficient assigned to each edge. However, a good model is not sufficient for a realistic simulation unless a suitable computation engine is used, especially when bilateral teleoperation is involved. Force feedback joysticks usually have six degrees of freedom, allowing a free and natural control over remote and/or virtual tools, but require the simulation to be updated at least 1000 times per second to allow the natural perception of forces acting on the tools. Since even high-end computers have inadequate performance with high complexity models, implementation usually takes advantage of preprocessing, client/server architectures or parallel vector processors. A recently growing research area is general purpose programming on graphics hardware, taking advantage of the computational power of Graphics Processing Units (GPU’s). Furthermore, a deformable environment presents planning problems that are not found in rigid environments and therefore not addressed in the literature. Humans have difficulties too in dealing with such environments, since motion is based on the continuous interaction with the environment and on the satisfaction of kino-dynamic constraints that depend on the nature of the deformable material. Thus, planning help is necessary to the human operator to perform motions that classical planners would rule out and still move the tool in a way that is safe for the environment.

In this paper we describe the advances made in simulation of deformable objects due to the combined use of precise model and fast computation hardware, and their application to high fidelity bilateral teleoperation. Furthermore, we also present our first steps into planning in deformable environments, for safe motions without damages to the environment. These new features in fact enable fast design and testing of difficult procedures due to the combination of planning and haptic feedback. The availability of a modular and easily configurable teleoperation system, based on our Penelope integration framework, allows easy integration of the features described above in a real set up. The paper summarizes some of the experiments carried out to verify our system, and describes our future plans to develop simulation-assisted teleoperation.

PRIOR WORK ON TELEOPERATION IN DEFORMABLE ENVIRONMENTS

The main classes of deformable models used in simulation are: Finite Element Models (FEM) [1] and Mass Spring Models (MSM) [2]. Because of the need to account for changes in object topology, i.e. cuts and ruptures, FEM models cannot be used in real time simulations, and MSM are normally used, provided that their physical properties can be estimated correctly. Two categories of methods are available for the estimation of MSM parameters that guarantee a realistic behavior. The first category is composed by methods that use some known property of the tissue to model [3]. The second class is composed by methods that use a minimization procedure to find the model that shows the closest behavior to that of the tissue to be modeled. To the first category, belong papers such as [4] describing a method to assign a value to each mass point; [5] proposing a 2D method to instantiate a triangular mass spring model; [6] using the linear system associated with a FEM to identify the tensor related to each node and to each edge of the model; and [7] deriving a formula to calculate the damping coefficient for different mesh resolution. All these methods can be very fast, but they are not very suitable to identify parameters for personalized models. The use of the second class of methods for MSM calibration leads to a minimization problem solved in [4] with simulated annealing technique, in [8] with neural networks, and in [9] and [10] with genetic algorithms. A few simulators that use deformable models to give realistic visual and haptic feedback have been developed. The best-known simulators are the AISIM project at INRIA [11], the Touch Lab by MIT [12], and the commercial product VEST System One [13].

From the implementation point of view, simulators have been developed using graphical cards and by allocating masses on a 2D texture and springs on a stack of 2D textures where each element stores a spring connected to a mass stored on the same position [14]. Since spring valence is not constant over the model, it is necessary to store invalid elements that leave the result buffers unaltered, thus reducing the computational efficiency of the approach. In [15], the physical simulation is not applied to the set of masses, but to a cubic grid of points interconnected to the 26 nearest ones. Since some of these points are external of the body's volume, the efficiency of the method is somewhat reduced. Many different implementations of deformable object simulations use FEM [16]. Recent methods take advantage of the computer GPU for the necessary matrix multiplications. However, computational time does not reach the minimal frequency of 1 KHz for haptics rendering. Some of the best-known general methods are GJK [17], CInDeR [18] and CULLIDE [19]. In [20] and [21] collision detection is implemented on the GPU using depth-buffers and the OpenGL SELECTION rendering mode.

Motion planning among deformable objects is still an unexplored research area. One main difficulty is to model deformations that accurately represent the object physical properties, while preserving the efficiency of the planner. In fact, a planner that uses a physically correct deformation model can be very slow [22] and a planner that uses only geometric deformations can compute unnatural motions [23]. Workspaces studied in most standard planning algorithms, generally include only static and rigid obstacles. Dynamic workspaces have been addressed, for example, in [24], [25], but in these cases, environment dynamics is meant to represent velocities and accelerations of rigid bodies, and not their deformation. Framework such as Probabilistic Roadmap Planner and Rapidly-Exploring Random Tree (RRT), [23], [26], describing path computation in totally dynamic and deformable environments, or [22], [27], for planning with physically correct deformations models, are not suitable as general approaches, because their main goal is avoiding the deformable objects, whereas better solutions may involve deforming the object to get access to the desired goal. Moreover the expensive computations needed for solving mechanical models and generating collision detection data structures are too time consuming for a general environment, and in fact they are applied only to simple objects, such as metal sheets or pipe-like robots. The objective of our work is to compute a safe trajectory that deforms the obstacles, when it is not feasible to avoid all the obstacles. The environment of our studies is a workspace filled with deformable object where we have to move rigid instruments, to simulate for example the scenario of the Minimally Invasive Robotic Surgery (MIRS). The trajectories needed are in general quite simple (almost linear), but they have to find room between colliding objects, therefore collision free trajectories are unfeasible and it is not possible to use classical techniques.

The paper is organized as following. In the next Section, we first briefly describe the modeling algorithm used for deformable bodies, and we focus on the calibration aspects of the model. Then, in the following Section, we address the difficulties of generating haptic feedback in real time, and we describe our current solution, based on the computer Graphic Processing Unit. Finally, the last Section describes a planning aid that can compute an instrument trajectory by deforming the obstacle body, provided that a given safety threshold is not passed. Finally a brief summary and our future plans conclude the paper.

MODELING DEFORMABLE OBJECTS

The modeling process of a deformable object requires the definition of a suitable internal topology of masses and springs to support the computation of realistic deformations and contact forces. From the surface model we can build a volumetric mesh by populating its interior with points and establish the connections between them. The points with their edges will form the tetrahedral elements of the mesh. We developed an algorithm that produces a smooth distribution of mesh points in order to obtain later a well-constrained mass-spring system [28]. Once the element distribution has been defined, the calibration of spring and masses has to be carried out [29].

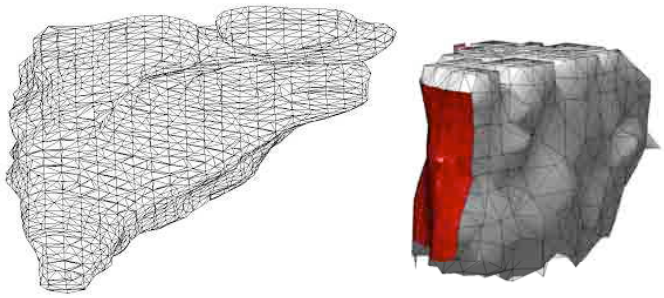

Figure 1: Examples of mesh generation for a deformable object (human liver).

Mass Calibration

As discussed in [28], the mesh formation process starts with the acquisition of the object surface, either by scanning or by known models. The model can consist of meshes with different level of details by simply specifying the max number of elements wanted. We decided to use meshes of 32000, 16000, 11000, 6000 and 1500 tetrahedral. In a biological model, i.e. the liver shown in Figure 1, the 32000 tetrahedral mesh (left) is very detailed the 1500 elements model (right) is significantly coarser; nevertheless all the tests carried out gave results in accordance to data found in medical literature. For the human liver it is known that its weight ranges between 1.2 and 1.5 kilograms and the density should lie in the interval [1040,1060] Kg/m^3 , with a mean value of 1050 Kg/m^3. During various test of model generation, the total mass of the reconstructed liver was within the correct interval, whereas the local density of the tissue dropped at times below the minimum value. The simple interpolation functions used in our method guarantee a reduced computational time. If the method is associated with a mesh generation procedure that refines the mesh where the discontinuities in the material are higher, it also allows a correct instantiation of the model masses.

Spring Calibration

The spring mesh forming the object model is calibrated using a genetic algorithm. Genetic algorithms are adaptive methods that can be used to solve search and minimization problems. If they are properly set up they offer good characteristics of exploration and exploitation of the search space. The measure to be minimized is the difference in the behavior of the simulated model and the desired one. The desired behavior is defined through a set of data that relate forces applied to some point of the model and the corresponding deformations. To test the correctness of the spring calibration method we created a reference model to generate the deformation measures, and then recomputed them from its topology, mass values and displacement measures using the developed spring calibration algorithm. In these tests we chose to use a mesh with a reduced number of tetrahedral due to the time requirements of the algorithm. The model used for the test is a cube with 1 cm edge made of 120 nodes, 377 tetrahedra and 590 springs. The mass values were instantiated considering a homogeneous density of 1040 Kg/m^3 and the spring constants were calculated in accordance to Equations 6 and 7 with a homogeneous Young modulus of 3.6 P a. The bottom face of the cube was considered attached to the ground and a force of 0.001 N was sequentially applied to the central node of each face for 0.2 seconds to obtain 200 sample points. Since the deformations used in the calibration should be representative of the deformations that should be simulated by the model, we restrict our test to punctual compression. Snapshots of the deformations are shown in Figure 2. We suppose to know only the position and the speed of the contact point and to completely ignore

any other information about the deformation since eventually we would like to use in vivo experimental data.

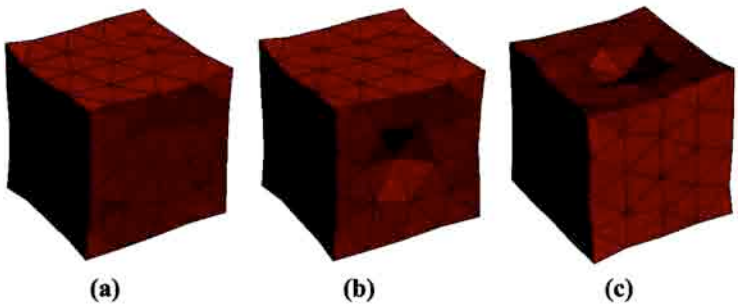

Figure 2: Examples of test for spring value calibration.

The output of the genetic algorithm is a sequence of real numbers that should be interpreted as the Young moduli of each point of the model. To have a first estimate of the goodness of the method we compared the Young moduli of the model defined as a reference and the reconstructed one. The maximum difference obtained is of 9.49% that corresponds to a difference of 8% in the spring's values, where the mean of the two end value is used. It is reasonable to think that moving away from the points used for the calibration makes the errors raise as the information derived by the measures decreases. We identify two important results. The first is that different models can have similar behaviors, at least for the class of deformation used in calibration. The second is that the calibration procedures can correctly identify a model that reproduces the measured behavior. It is important to identify and use, in the calibration of the springs, a set of deformations that completely represents the behavior of the object that we want to reproduce.

GPU-BASED MODEL COMPUTATION

Mass-Spring systems are commonly used in interactive dynamic simulation thanks to their much lower computational cost compared to FEM-based methods. Since the model is based upon a discretization of the object volume it is, in principle, vey suitable for parallel computation, such as the one carried out using a computer GPU. The GPU is the vector processor of current video cards. It is composed of a small set of parallel computational units and can be programmed with a language explicitly thought for 3D graphic, such as OpenGL [30] and OpenGL Shading Language [31]. The computational model is based on a kernel function, called fragment shader, evaluated independently for each rectangular element in the rendering region, defined by a primitive type and a set of vertices. In [34] we have described a novel algorithm for the fast computation of the model described above using a GPU.

All traditional CPU algorithms have to be adapted to be implemented on GPU: this operation is often complex due to the complexity of the correct specification of all rendering parameters, the issues of video driver implementations and limited debug support. For each mass we store a position vector for two contiguous time instants, the force vector F and a set of constant values, such as mass or a "is on" surface flag. Since each buffer element can store a maximum of 4 scalars, we need to use auxiliary buffers. Moreover, the current hardware limitations impose to use 2D arrays to store up to 4096 elements. The representation of springs requires a texture stack. In each element at position (u, v) we store a spring connected to a mass allocated at the same position. Each spring is processed and stored twice, since there are two connected masses. The spring data structure requires all the 4 allocable scalars in fact; it is composed by the rest length, the elastic coefficient, the damping coefficient and a scalar representing the density of the object. To store the entire system, we need as many textures as the maximum spring valence. However the analysis of complex geometric models evidences that the valence usually varies from a minimum of 3 to a maximum of 18. Consequently, a lot of elements in the texture stack are useless and need to be filled with null values. One of the biggest issues using mass-spring systems is the inability to model some material volumetric properties, for example incomprimibility. This limitation can be overcome introducing a volumetric entity, tetrahedra, and considering additional force contributions, as suggested in [32]. The representation of tetrahedra is similar to the spring one: we use a texture stack where each element at position (u, v) stores a tetrahedron connected to a mass allocated at the same position. In this case, the memory requirements are higher: each tetrahedron is stored and processed 4 times. The tetrahedron data structure is composed by 3 indices, which defines the triangle opposed to the processed mass, and the rest volume.

A probing action is shown in Figure 3 (left). The implementation of this action is realized in two distinct phases. In the first phase, we mark all masses colliding with the instrument. These masses will be affected by the additional external force applied by the operator and will contribute to the final haptic forces. To reduce computational time, we approximate the tool geometry with an ellipsoid and the body volume with the particle set. This may cause the lack of

collision detections if the body volume is not discretized with a high density mass set. The colliding masses are marked and stored in an additional texture and will be used in haptic force computation. In the second phase we compute the collision response. This computation requires loading the previously stored marks, to discard unmarked masses and translate the remaining ones to the approximated tool surface.

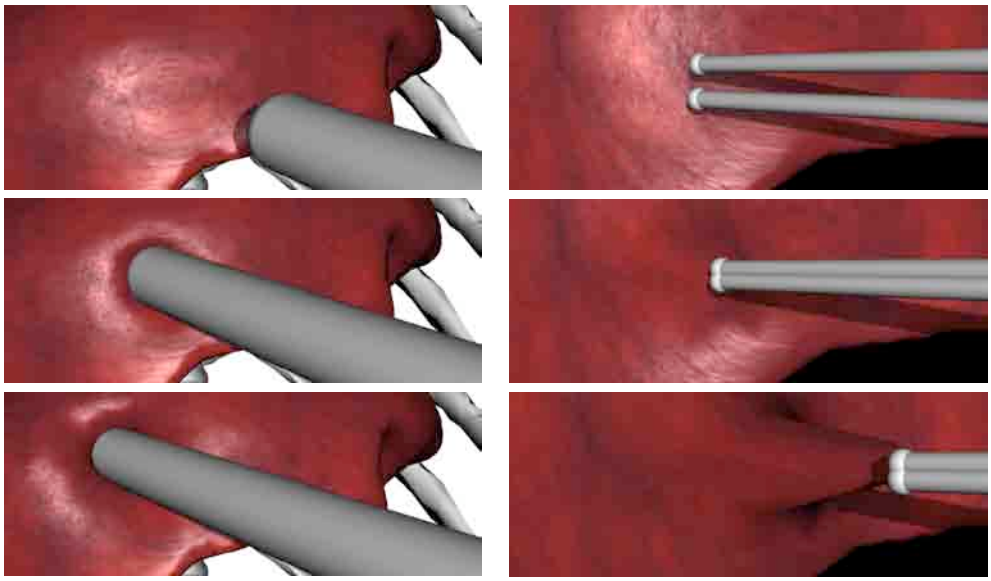

Figure 3: Examples of probing and grabbing actions.

The grabbing gesture shown in Figure 5 (right) allows the operator to manipulate object parts with tweezers. Also this implementation consists of two different steps. The first step aims at marking grabbed masses. The simplest method requires checking if positions are inside a small sphere centered in the closure point. This process has to be realized only at tweezers' closing instant, so we can use a more complex methods if GPU has enough computational power: our method marks masses if one of the connected springs intersects with the test sphere. The second step realizes the grabbing action by translating the marked masses according to tweezers' motion. This method has a higher complexity than the simple translation but allows avoiding degeneration of springs and tetrahedra. Since this step is required only if at least one mass is grabbed, we use a query to count the number of marked masses at closing instant but we start using its results only after one simulation step to avoid synchronization issues due to transfer of counter data.

Test	QuadroFX3450	GoForce7900	GeForce8800
Physical Simulation	1006 Hz	XXX Hz	8182 Hz
Marking of masses	940 Hz	XXX Hz	7112 Hz
Haptic feedback	524 Hz	XXX Hz	2632 Hz

TABLE I

SIMULATION RATES WITHOUT VOLUME PRESERVATION.

Test	QuadroFX3450	GoForce7900	GeForce8800
Physical Simulation	278 Hz	XXX Hz	2338 Hz
Marking of masses	268 Hz	XXX Hz	2184 Hz
Haptic feedback	238 Hz	XXX Hz	1730 Hz

TABLE II

SIMULATION RATES WITH VOLUME PRESERVATION.

Table I and II highlight the simulation rates of our implementation using the model of the liver shown in Figure 1 (7750 masses, 38077 tetrahedra and 48254 springs). The second table refers to the simulation in which volume preservation is taken into account. As can be easily seen, we reach a rate of at least 1 KHz on current GPU. This is sufficient to provide realistic haptic feedback to an operator.

PLANNING WITHIN A DEFORMABLE ENVIRONMENT

Before addressing the planning problem, it is necessary to choose the environment representation. We use the point-based representation of objects: it allows reconstructing the scene in a more automatic and straightforward way, to do fast model resampling, and to easily compute collisions and penetration depth. Following the method described in [33] point clouds can be easily generated from different 3D representation (CAD models, triangular surfaces, segmented medical images, implicit functions) thus our method can be generalized to many scenarios represented by real data sets. We compute a penetration depth at each model point, and then we minimize the penetration function by choosing the most suitable trajectory for the instrument tip until it reaches a configuration near the final goal. We also compute the force direction, pointing outside the object, to identify the next possible point of the trajectory. Thus, each state of the problem space consists of the position of the instrument and the penetration function at that point. Such a path is feasible when a collision does not damage the obstacle and provided that the obstacle deforms of an appropriate amount. Key element of the algorithm is collision detection, since besides detecting a collision it should also compute the penetration depth (PD) between two objects, where with PD we mean the minimum displacement to be applied to one object to remove the colliding state.

The object meshless representation has the advantage of describing both surface and volume of an object by a discrete set of points without using edges to model the connections. The model is composed of the set of surface points called phyxels. The probe is also described by a meshless model (we use a sphere but it could be easily generalized). We formulate the problem of computing the minimum penetration trajectory among deformable obstacles as a minimization problem, where the optimization variables are the trajectory parameters, and the performance index is a measure of how much the probe collides with the objects on its path. As the trajectory function, we used polynomials of degree 3 in order to assure smoothness but also the capability to overcome more than one obstacle. To see how the algorithm scales with increasing complexity, we ran it with environments with a varying number of objects, ranging from 3 to 10. Then, we repeated the tests with polynomials of degree 5 to compare performance, both in execution time and in the quality of the trajectories generated. In every scenario, the objects used are irregular objects, each of about 1300 phyxels, whereas for the probe we use a small spherical object. The function describing the object penetration is null outside the objects and because of this, the algorithm pushes the computed trajectory to the border of the obstacles, but not farther. For the calculation we used an inflated version of the probe model, with the result of pushing the probe outside the objects.

An example scenario is shown in Figure 4, with two different values of stiffness for the various objects. In this case, the trajectories were forced on a plane passing through all the object, and had to collide with them.

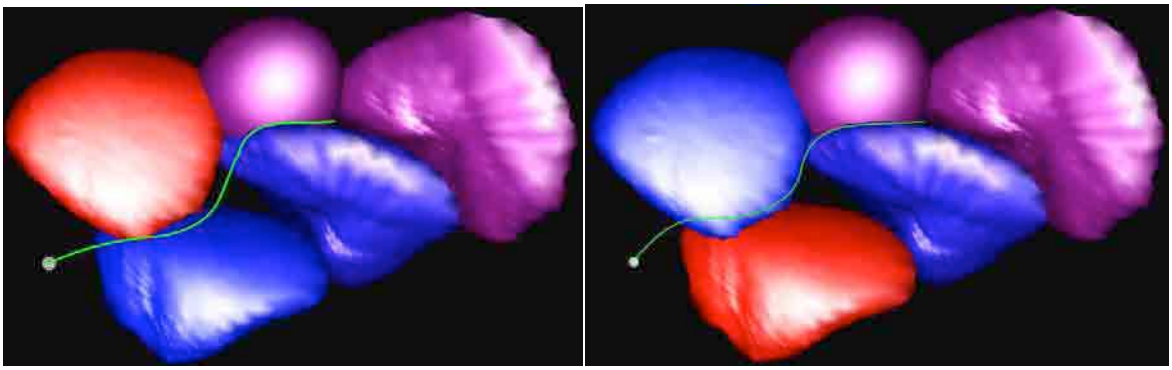

Figure 4: Different stiffness factors lead to different trajectories. Red objects are the most stiff.

We compared different solutions and rank them based the minimal cost function and the computation time. In the experiments we performed, the algorithm is usually successful but the final result depends on the initial choice of the parameters. In particular, in the presence of non-convex object, it could converge to local minima. This also makes comparisons among experiments with different scenarios particularly hard. There are however a few considerations that can be made. The most notable is that the algorithm bends the trajectory more with an increase of the iterations. In particular it gives more relevance to the coefficients of the higher degree terms. Probably as a consequence of this, the five degree trajectory is not as successful as the third degree one and, as expected, the computation takes more time.

CONCLUSIONS

In this paper we have briefly summarized some of the issues related to developing operator aids for teleoperating rigid instruments among deformable objects. This situation may arise during space activities involving maneuvers with inflatable elements, for which some deformation is acceptable, but within given force limits. To address this problem we first developed and tested a few algorithms for modeling deformable objects, in particular we focused on tetrahedral mesh representation, which ensures good calibration features and haptic rendering compatible with real time requirements. To address the planning problem instead, we used meshless models for which collision detection and penetration depth can be computed in a simpler way than with mesh-based models. Furthermore, we envision planning as an off-line activity, thus reducing the need of real time performance. In the paper we have given simple examples of the three algorithms results. The tetrahedral representation can be calibrated, either using data from the literature or taken from direct measurements, with good approximation, yielding model behavior within a few percentage points of the true object behavior. The GPU based model rendering has demonstrated a rendering cycle in excess of 1KHz, and thus can be used in high performance simulations to compensate communication time delay, or in training activities. Finally, a planner that takes advantage of the object deformation capability has shown to compute feasible trajectories in environments where classical planners fail to provide a solution.
Clearly, the next objective of this research is to develop an integrated framework that can combine force reflection teleoperation with high performance simulation and planning, to give operators the flexibility to test procedures on a simulator, enhanced with planning aids, and to integrate real video with simulated graphics to compensate image delay.

REFERENCES

[1] M. Bro-Nielsen, “Surgery simulation using fast finite elements.” in VBC, 1996, pp. 529–534.
[2] S. Gibson, C. Fyock, E. Grimson, T. Kanade, R. Kikinis, H. Lauer, N. McKenzie, A. Mor, S. Nakajima, H. Ohkami, R. Osborne, J. Samosky, and A. Sawada, “Volumetric object modeling for surgical simulation,” Medical Image Analysis, vol. 2, no. 2, pp. 121–132, 1998.
[3] Y. C. Fung, Biomechanics - Mechanical Properties of Living Tissues,2nd ed. Springer-Verlag, 1993.
[4] O. Deussen, L. Kobbelt, and P. Tucke, “Using simulated annealing to obtain good nodal approximations of deformable objects,” in Computer Animation and Simulation ’95, D. Terzopoulos and D. Thalmann, Eds. Springer-Verlag, 1995, pp. 30–43.
[5] A. V. Gelder, “Approximate simulation of elastic membranes by triangulated spring meshes,” J. Graph. Tools, vol. 3, no. 2, pp. 21–42, 1998.
[6] G. Picinbono, J. Lombardo, H. Delingette, and N. Ayache, “Anisotropic elasticity and force extrapolation to improve realism of surgery simulation,” 2000.
[7] C. Paloc, F. Bello, R. Kitney, and A. Darzi, “Online multiresolution volumetric mass spring model for real time soft tissue deformation,” in MICCAI ’02: Proceedings of the 5th International Conference on Medical Image Computing and Computer-Assisted Intervention-PartII. London, UK: Springer-Verlag, 2002, pp. 219–226.
[8] A. Nürnberger, A. Radetzky, and R. Kruse, “A problem specific recurrent neural network for the description and simulation of dynamic spring models.”
[9] G. Bianchi, B. Solenthaler, G. Székely, and M. Harders, “Simultaneous topology and stiffness identification for mass-spring models based on fem reference deformations.” in MICCAI (2), 2004, pp. 293–301.
[10] J. Louchet, X. Provot, and D. Crochemore, “Evolutionary identification of cloth animation models,” in Computer Animation and Simulation ’95, D. Terzopoulos and D. Thalmann, Eds. Springer-Verlag, 1995, pp. 44–54.
[11] “Aisim, laparoscopic surgery simulation group, inria:
http://www.sop.inria,fr/epidaure/formercollaborations/aisim/index-gb.html.”
[12] “Touch lab, laboratory for human and machine haptics, mit: http://touchlab.mit.edu/index.html.”
[13] “Vest system one, select-it vest systems ag: http://www.selectit.de/index.php.”
[14] J. Georgii, F. Echtler, and R. Westermann, “Interactive simulation of deformable bodies on GPUs,” in Simulation und isualisierung 2005 (SimVis 2005), 3-4 Marz 2005, Magdeburg (T. Schulze, G. Horton, B. Preim, and S. Schlechtweg, eds.), pp. 247–258, SCS Publishing House e.V, 2005.
[15] T. Soresen and J. Mosegaard, “Haptic feedback for the GPU-based surgical simulator,” Medicine Meets Virtual Reality 14, pp. 523–528, 2006.
[16] W. Wu and P.-A. Heng, “An improved scheme of an interactive finite element model for 3D soft-tissue cutting and deformation,” The Visual Computer, vol. 21, no. 8-10, pp. 707–716, 2005.
[17] E. Gilbert, D. Jonhson, and S. Keerthi, “A fast procedure for computing the distance between complex objects in three-dimensional space,” IEEE Journal of Robotics and Automation, vol. 4(2), pp. 193–203, Apr. 1988.
[18] D. Knott and D. K. Pai, “CIndeR: Collision and interference detection in real-time using graphics hardware,” in Graphics Interface, pp. 73– 80, 2003.
[19] N. K. Govindaraju, S. Redon, M. C. Lin, and D. Manocha, “CULLIDE: Interactive collision detection between complex models in large environments using graphics hardware,” in Proceedings of the 2003 Annual ACM

SIGGRAPH/Eurographics Conference on Graphics Hardware (EGGH-03) (A. S. W. Mark, ed.), (Aire-la-ville, Switzerland), pp. 25–32, Eurographics Association, July 26–27 2003.
[20] J.-C. Lombardo, M.-P. Cani, and F. Neyret, "Real-time collision detection for virtual surgery," in Computer Animation, p. 82, 1999.
[21] J. R. Navarro, M. Sainz, and A. Susin, "GPU based cloth simulation with moving humanoids," tech. rep., Nvidia Corporation, Universitat Politecnica de Catalunya, 2005.
[22] E. Anshelevich, S. Owens, F. Lamiraux, and L. Kavraki, "Deformable volumes in path planning applications," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2000, pp. 2290–2295.
[23] O. B. Bayazit, J.-M. Lien, and N. M. Amato, "Probabilistic roadmap motion planning for deformable objects," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2002.
[24] D. Hsu, R. Kindel, J. C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," in *Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, 2000.
[25] R. Kindel, D. Hsu, J. Latombe, and S. Rock, "Kinodynamic motion planning amidst moving obstacles," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2000, pp. 537–542.
[26] S. Rodriguez, J.-M. Lien, and N. M. Amato, "Planning motion in completely deformable environments," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2006.
[27] F. Lamiraux and L. Kavraki, "Planning paths for elastic objects under manipulation constraints," *The International Journal of Robotics Research*, vol. 20, no. 3, pp. 188–208, 2001.[18] M. Levoy and T. Whitted, "The use of points as a display primitive," University of North Carolina, Tech. Rep. 85-022, 1985.
[28] L. Orsatti, D. Botturi, and P. Fiorini, "*Deformable models of body organs for surgery simulation*," in CARS2005: Proceedings of the 19th International Conference on Computer Assisted Radiology and Surgery, 2005, p. 1366.
[29] D. Zerbato, S. Galvan, and P. Fiorini, "*Calibration of Mass-Spring Models for Organ Simulations*", in IEEE/JRS Int. Conf. Intelligent Robots and Systems (IROS'07), San Diego (CA), 2007.
[30] M. Segal and K. Akeley, The OpenGL Graphics System. A Specification. Version 2.1, Dec. 2006.
[31] J. Kessenich, The OpenGL Shading Language, Sept. 2006. Version 1.20, Revision 8.
[32] Y. Lee, D. Terzopoulos, and K. Waters, "Realistic modeling for facial animation," in SIGGRAPH 95, pp. 55–62, 1995.
[33] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. Silva, "Computing and rendering point set surfaces," *IEEE Transactions on Computer Graphics and Visualization*, vol. 8, no. 4, 2002. [Online]. Available: citeseer.ist.psu.edu/alexa02computing.html
[34] M. Altomonte, P. Fiorini, D. Botturi and D. Zerbato, "*Surgical Simulator on GPU*", in IEEE/JRS Int. Conf. Intelligent Robots and Systems (IROS'07), Nice (France), 2008.