Eurographics Italian Chapter Conference (2007)
Raffaele De Amicis and Giuseppe Conti (Editors)

Techniques for Computer Assisted Surgery.

N. Pietroni, G. Turini [†] F. Ganovelli, R. Scopigno [‡]

Abstract

In the recent decades robotic and computer science have been gaining more and more relevance in all aspects of our lives. In surgery, for example, they gave birth to procedures that would be impossible to perform otherwise (e.g. tele-surgery, nano-surgery). On this regard, these applied sciences already play an important role in assisting the surgeon both in the operative room and as a support in the education of young surgeons, but much work has still to be done. This paper presents some research and applicative results on Computer Assisted Surgery achieved in the framework of Endocas, a newly founded Center of Excellence in Pisa: a method for segmentation of anatomic parts from 3D dataset able to recover shapes from noisy 3D dataset; a technique for simulating bone drilling using a adaptive decomposition of tetrahedral meshes; a new open source library to support the implementation of techniques for simulating deformable objects.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computational Geometry and ObjectModeling]: Physically based modeling J.3 [Life and Medical Sciences]: Medical information systems I.4.8 [Scene Analysis]: Surface fitting

1. Introduction

The advancements in fields of applied sciences play an important role on the improvement of the quality of life. In most of the cases these advancements are pursued by researchers of the same kind and the results are directly available to the end users. Using applied sciences in medicine is a different story, because the difference of scientific knowledge between the technology developers and the surgeons is so wide that they can hardly communicate their needs correctly. Nevertheless the collaboration is a must: a simulator for non-invasive surgery, for example, cannot be developed without the guide of a trained surgeon as well as a flight simulator cannot be developed without the help of a trained pilot.
Endocas is a Center for Endoscopic Computer Assisted Surgery located inside the Ospedale di Cisanello, in Pisa. The center was born primarily with the purpose of putting together people from different backgrounds that would not meet otherwise. It is the result of a joint proposal of the University of Pisa, of Scuola Superiore S.Anna (CRIM Lab) and the Visual Computing Laboratory (ISTI-CNR) in the framework of MIUR funding for centers of excellence. Its goals are to address key knowledge, technology, and systems design barriers that must be overcome in the development of CAS systems.

In this paper we overview some of the results obtained in the first 2 years of activity of the center. In Section 2 we show a novel scheme for extracting anatomical organs from noisy CT data based on active contours and marching cubes techniques. This approach was successfully used to reconstruct the vena cava from CT data where standard methods failed. In Section 3 a technique for simulating bone drilling is shown. The approach is very general and it can be used whenever the solids are represented by tetrahedral meshes, with limited influence on the performance of the rest of the simulation. The final section (Section 4) is dedicated to an open source library we are writing for supporting the implementation of simulation related software. Conclusions and future work (Section 5) complete the paper.

[†] Endocas - Center for Computer Assisted Surgery
[‡] Istituto Scienza e Tecnologie dell'Informazione - CNR

2. Segmentation of noisy data

The recovery of the external surface of an organ is relevant for 3D visualization and can be used to support quantitative diagnostic measurements. However, this task may be challenging, even in the apparently easy case of visually uniform regions like vessels with injected contrast agent. In fact, the segmentation method should, in this case provide a closed surface, with smooth borders, avoiding leakages in neighboring contrasted structures but not missing bifurcations with smaller vascularizations. This means that the algorithm should provide a topology control, be robust against noise, the surface must be kept smooth, but not too much if we want to detect small structures or local pathologies.

Surface reconstruction methods proposed in literature have, in general specific advantages and drawbacks. Simple isosurface extraction [LC87] does not guarantee topology control and it is quite sensitive to noise. Holes and leakages are also a relevant problem for more advanced level sets or geodesic surfaces methods[MSV95, CKS95]. To prevent irregularities, holes and leakages the best approach is to use a parametric deformable surface instead of an implicit curve, with the same approach as the classic balloon snakes [LC93]. These methods (see [MT96] for a review of the first medical applications) are efficient and robust. The elastic forces avoid the surface block near a noisy pixel and prevent leakages.

2.1. The balloon algorithm

The basic idea of balloon algorithms is to place a very small deformable bubble inside the structure we intend to segment, e.g. aorta, colon, etc. and then to inflate it. We implemented the bubble as a triangulated mesh. The bubble is then inflated approximating with an iterative node displacement a dynamic model where the mesh nodes are treated like masses and several forces are applied to each of them. The inflation of the bubble is accomplished through a force directed as the local surface normal vector. The magnitude of this force is inversely proportional to the difference between the local gray level and the reference value for the region to be segmented. As the bubble grows, the triangles become bigger. Since the final goal of the process is to fit the bubble over the surface of the structure, it is self evident that the size of the triangles must be of the order of the size of the voxel. As a triangle grows more than a fixed threshold, we perform a mesh refinement, splitting the longest edge and adding two new triangles.

2.2. Self intersection

During the evolution, the mesh can intersect itself, because the anatomical structure to be segmented is not always topologically equivalent to a sphere. In this case the method as explained so far would not work properly, leading to a self-intersecting surface. Worse, if no intersection control is performed, the bubble will never find the equilibrium position, since it can inflate forever without increasing the occupied volume. To restore a non self-intersecting surface (changing the topology if necessary) our solution is to perform a periodical collision detection in order to find intersecting portions of the surface and blocking the vertices belonging to intersecting faces. Then we build a distance map computed on a regular grid (coincident in out implementation with the voxel grid) assigning to each voxel of the volume the signed distance from the closest non intersected face with a sign, which is negative if the point is inside the volume enclosed by the surface and positive otherwise. Due to the use of the part of the surface that is not self-intersecting, the result is that the iso-surface of value 0 of the map is a surface defining the limits of the region of our interest with no self-intersections and the correct topology.

2.3. Results

We successfully tested our algorithm first on synthetic images of structures to validate the self collision detection/marching cubes method (see Figure1). In all the situations created, we obtained a closed smooth surface defining an internal region close to the voxelized volume of interest and with the same euler number.

The algorithm was then applied to CT scans of the abdomen in order to segment the aorta. Fig.2 shows four steps during the surface evolution. In all the cases tested segmentation results were accurate and fast, no manual corrections were required and the only user intervention performed was the seed placement. The time needed to produce the vena cava model is about 30 seconds.

3. Towards Surgical Simulator: Bone Drilling

Bone drilling is an essential task in many surgical procedures as: mastoidectomy, cochlear implantation or orbital surgery. It essentially consists in eroding the part of the bone in contact with the tip of the surgical tool when a sufficient pressure is exerted.
The bone is an almost rigid material and therefore suitable for a regular discretization of the space, for these reason most of the approaches proposed in literature use voxel-based representations so that it is easy to show material removal by playing with material density in the voxels; as in the *voxmap-pointshell* [MPT99, PPT*02, MSB*04], or in GPU voxel-based techniques [AGG*02].
Mesh based approaches are also widely used for performing cuts [BMG99, GCMS00, NvdS00] and for rendering material removal [AGP*06, SC98] but their accuracy is strictly dependant on the the size of the mesh elements.
Unfortunately, there are cases in which drilling is only a part of the task, and parts of the same object are also cut away or, worse, the bone is slightly deformable and therefore voxel-based techniques do not work well.

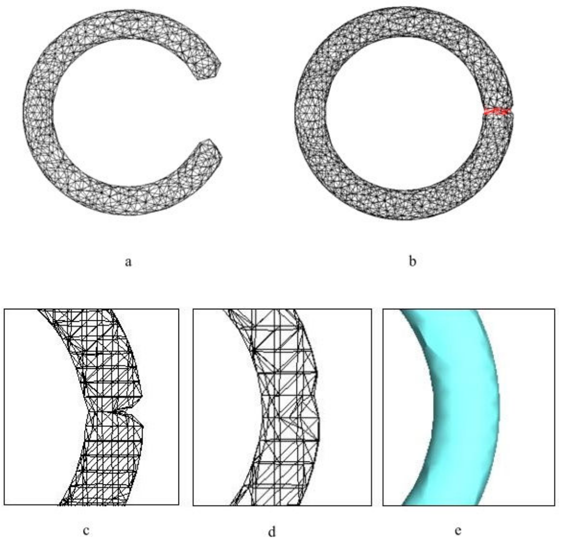


Figure 1: *Example of self intersection managing: a.The bubble starts to expand. b.The bubble fits the geometry and self intersection is detected (faces that generate self intersections are shown in red). c.d.e.Then using Marching Cubes and expansion steps we obtain the final triangle mesh.*

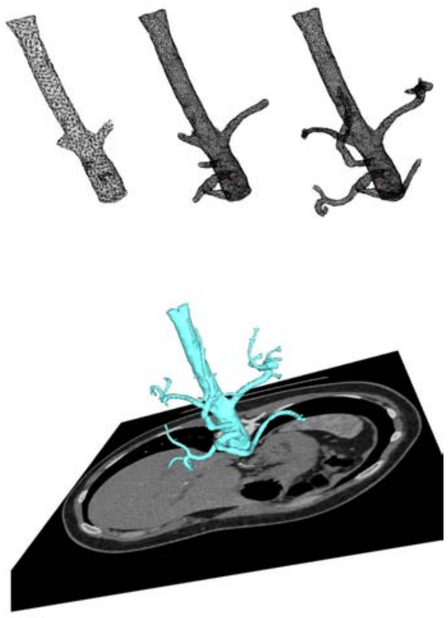


Figure 2: *Different phases of extraction of an aorta.*

For these reasons we have developed a novel method to perform drilling on objects represented explicitly by means of tetrahedral meshes. These meshes can be easily generated from CAT/RM data, can be used to perform physical simulation of solids, are easy to render and can also be used to simulate material with non uniform density.
Since the number of tetrahedra strongly influences the performance of the physical simulation, we have to face two opposite constraints: a high number of tetrahedra for rendering drilling and a low number for physical simulation.

3.1. Our drilling approach

We have chosen to decouple the drilling from the physical simulation representing the drilling at sub-tetrahedral level, that is defining a hierarchical decomposition scheme of the tetrahedron recursively splitting tetrahedra in 8 new tetrahedra (see Figure 3).

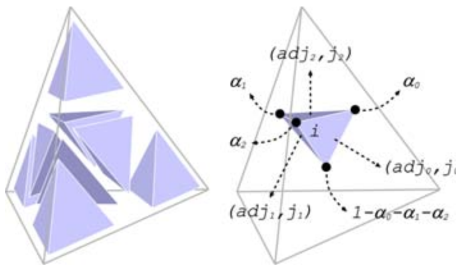


Figure 3: *Tetrahedral subdivision scheme.*

Our algorithm manages the collisions between the tool and the mesh, so when a tetrahedron of the original mesh comes to contact with the tool, the decomposition scheme is applied to adapt the mesh detail to those of the tool, in this manner the removal of the material can be done by simply removing tetrahedra.
So we consider each tetrahedron as the root of a hierarchy, where each node is a tetrahedron and its children are obtained by refining the parent node in 8 smaller tetrahedra. Moreover we represent the subtraction of a region of a tetrahedron by computing a frontier in the hierarchy such that no tetrahedra bigger than a given threshold cross the border of the region and by tagging the nodes of the hierarchy inside the region as *erased* (see Figure 4).

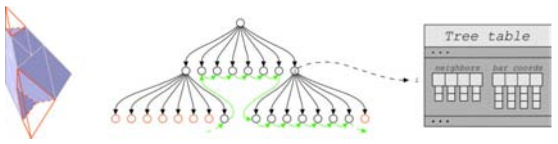


Figure 4: *A tetrahedron eroded (left), its hierarchy (center) and the TreeTable (right).*

We have developed a pre-processing step to compute part

of the data needed for the application of our decomposition scheme, in order to optimize the refinement during the simulation. All these data are stored in a static *Tree Table* in such a way that they are easily accessible. Moreover we have used another data structure to implement a spatial indexing technique essential to speed up the collision detection. Then a list of visible node faces *v_list* was added to each tetrahedron to improve the rendering time.
Finally our method relies on two basic operations: *Refine* and *Erase*. Both transorm a single hierarchy node: the first refines a node in 8 new tetrahedra while the second tags a node as *erased* (updating the data structure that refer to that node).
The advantages of our approach are twofold: 1) it is independent on the method adopted for physical simulation. 2) collision detection, haptic and visual rendering rendering are modeled at sub-tetrahedral level.

3.2. The simulation system

The simulator developed is a *multi-threaded* application formed by three threads: the visual rendering thread, the collision detection thread and the haptic thread.

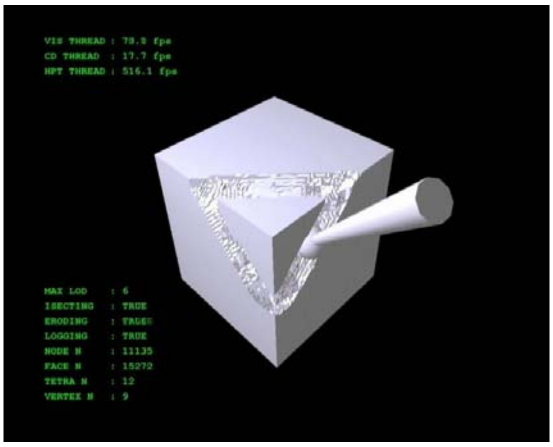
The rendering thread simply renders all the visible faces of tetrahedra. If a tetrahedron has been eroded, then we run through its *v_list* and render each visible face of the nodes in the list. We employ a simple optimization consisting in compiling a display for each tetrahedron, containing the rendering of its *v_list*. The display list is compiled every time the *v_list* is modified and called to render the tetrahedron when it is not being eroded.

The collision handling thread manages the collision detection that is crucial for our purposes because it determines all the actions to be performed on the tetrahedra (*Refine* and *Erase*) and the feedback to return to the haptic interface.
Moreover the surface of the drilling instrument is sampled with an evenly distributed set of *sample points* that is used to detect collisions applying the spatial hashing technique proposed in [THM*03]. During collision detection we also compute the penetration depth and direction to render forces.

The haptic interaction thread updates the tool position accordingly to the haptic interface movements and ensures the high rendering frequency of force feedback that is essential to simulate a realistic haptic interaction. Since the collision detection cycle, on which penetration depth depends, has an updating rate too low, we use extrapolation to compute the forces to give to the haptic interface at high frequency (around 500Hz).

3.3. Results

Our tests concern the collision detection, the handling of geometry and the updating rate of the three threads. The object is still static, even if this is not assumed anywhere in the implementation.


```
VIS THREAD : 78.5 fps
CD THREAD  : 17.7 fps
HPT THREAD : 516.1 fps

MAX LOD    : 6
ISECTING   : TRUE
ERODING    : TRUE
LOGGING    : TRUE
NODE N     : 11135
FACE N     : 15272
TETRA N    : 12
VERTEX N   : 9
```

Figure 5: *A screenshot of the video* ***cube****.*

Figure 6 shows the performance of the algorithm in a test case made of a cube decomposed in 12 tetrahedra. As can be seen in Figure 6.(a) the number of tetrahedra (i.e. nodes) increases rapidly during the erosion but this growth does not influence the performance of the method as one could expect. In terms of memory occupation, the node requires only 5 bytes instead of the 16 required by the tetrahedron. Moreover the rendering time of the visible faces of the nodes of the tetrahedron is optimized using the display list associated with the tetrahedron.
Figure 5 shows a snapshot of the cube at the end of the process.

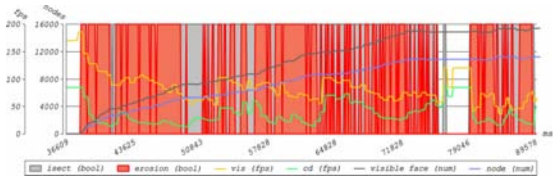


Figure 6: *Data produced during the recording of the video* ***cube****: the number of nodes and visible faces and the updating rates of the visualization and collision handling threads.*

4. IdoLib: Interactive Deformable Objects Library

Idolib (http://idolib.sourceforge.net) is a C++ library initially developed at the Visual Computing Laboratory and now also developed at Endocas. The goal of Idolib is to provide a set of basic tools and definitions for supporting the developer of a simulation-related application.

4.1. The Core

Idolib is specialized for explicit representation of the objects, both of the surface (triangular meshes, point based) and of the solid (tetrahedral meshes, point sampled volumes).

The core of Idolib is based on two main identities: *particles* and *relations*. The particle is intended as a one dimension sampling of the volume with a parametric set of geometric and physical properties (position, velocity, acceleration, mass etc.) and the relation is, as the name says, a relation among a set of particles. For example in the simplest case of the mass spring system the particle is the mass and the relation is the spring. The relation does not have to correspond to a geometric figure as for spring or triangles of tetrahedra: for example in the implementation of a point based method the relation binds together all the particles closer than a given radius to each other.

Figure 7 shows a scheme of the Idolib structure. The arrows follow the parts of the library in the order they are required in a step of simulation. A simulation step is essentially a cycle over all the particles to compute, for each particle, its position at the end of the time step. Computing such position typically requires two things: the force acting on the particle and an integration scheme.
The computation of the force acting on the particle depends on the state of the system and on the specific method used to implement the physical behavior of the object. In a mass spring system, for example, the force is computed by summing the contribution of all the springs connected to the particle while in the explicit FEM the contribution is given by the stress of the all the tetrahedra sharing the particle. More in general, the force acting on a paricle depends on all the relations that include the particle.
The integration scheme is the way forces and state of the system are used to compute the position of the particle at the end of the step. Idolib provides a number of common integration schemes and a callback-based manner to add new schemes. Furthermore, one can switch among different integration schemes during the simulation. Similarly, new methods can be easily added to those already implemented in IdoLib (refer to [CA99, OH99, MHTG05, CB01, MKN*04] for the details).
Note that IdoLib does not take control of the simulation loop, the main call only performs a single step of simulation.

Dependencies.
Idolib is written in C++ with STL library and relies only on another library, called VCG (Visualization and Computer Graphics Library, http://vcg.sourceforge.net), for the basic geometry structures and linear algebra.

4.2. Further work

IdoLib is still at a early stage. Although the structure is quite stable, further work is needed to provide a simple way to visualize the state of the simulation, i.e. all the numerical quantities of the system (such as the energy, the local stress and strain etc.) which is always essential in the debugging phase. Similarly, a complete documentation and examples is still to be done.

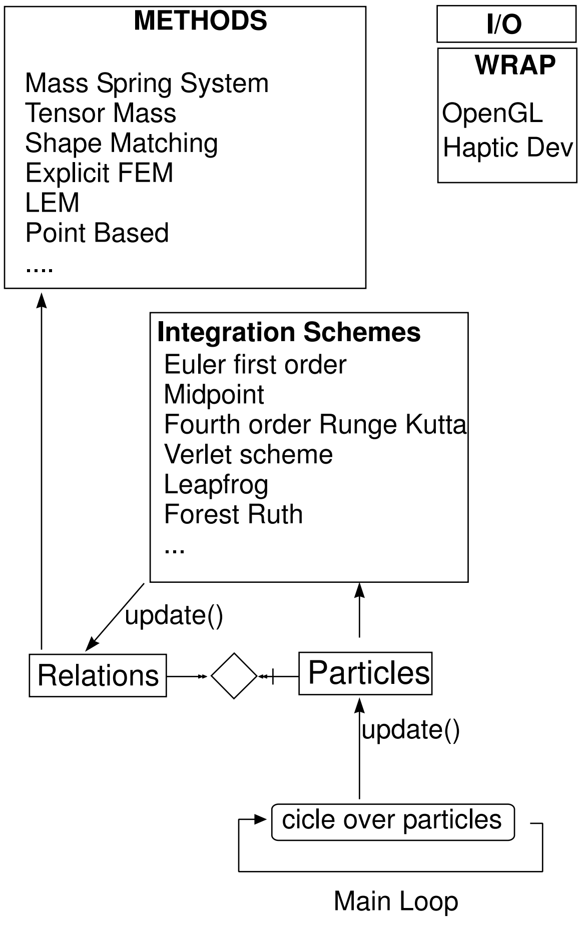


Figure 7: *Description of the data structure of Idolib.*

5. Conclusions

This paper reports some recent results obtained in the framework of the center for endoscopic computer-assisted surgery Endocas. We chose two different applications of physical simulation to medicine: a method for automatic segmentation and reconstruction of shapes from a noisy 3D dataset and a method for simulating drilling over quasi rigid objects. The paper also presents the library for physical simulation which is being developed and over which the applications where built.

References

[AGG*02] AGUS M., GIACHETTI A., GOBBETTI E., ZANETTI G., ZORCOLO A.: Real-time haptic and visual simulation of bone dissection. In *IEEE Virtual Reality Conference* (Conference held in Orlando, FL, USA, March 24–28, Feb. 2002), pub-IEEE, pp. 209–216. 2

[AGP*06] AGUS M., GOBBETTI E., PINTORE G., ZANETTI G., ZORCOLO A.: Real-time cataract surgery simulation for training. In *Eurographics Italian Chapter Conference* (Conference held in Catania, Italy, 2006), Eurographics Association. 2

[BMG99] BIELSER D., MAIWALD V., GROSS M.: Interactive cuts through 3-dimensional soft tissue. *Computer Graphics Forum (Eurographics'99 Proc.) 18*, 3 (Sept. 1999), C31–C38. 2

[CA99] COTIN H. D. S., AYACHE N.: A hybrid elastic model allowing real-time cutting, deformations and force-feedback for surgery training and simulation. In *CAS99 Proceedings* (May 1999), pp. 70–81. 5

[CB01] COSTA I. F., BALANIUK R.: Static solution for real time deformable objects with fluid inside, Jan. 08 2001. 5

[CKS95] CASELLES V., KIMMEL R., SAPIRO G.: Geodesic active contours. In *Fifth International Conference on Computer Vision.* (1995). 2

[GCMS00] GANOVELLI F., CIGNONI P., MONTANI C., SCOPIGNO R.: Enabling cuts in multiresolution representation. In *CGI 2000 Proceedings* (2000), N.Magnenat-Thalmann, D.Thalmann, (Eds.), p. (in press). 2

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3D surface construction algorithm. In *ACM Computer Graphics (SIGGRAPH 87 Proceedings)* (1987), vol. 21, pp. 163–170. 2

[LC93] L.D. C., COHEN I.: Finite element methods for active contour models and balloons for 2d and 3d images. *IEEE trans. on Pattern Analysis Machine Intelligence 15*, 11 (1993), 1131–1147. 2

[MHTG05] MÜLLER M., HEIDELBERGER B., TESCHNER M., GROSS M. H.: Meshless deformations based on shape matching. *ACM Trans. Graph 24*, 3 (2005), 471–478. 5

[MKN*04] MÜLLER M., KEISER R., NEALEN A., PAULY M., GROSS M., ALEXA M.: Point based animation of elastic, plastic and melting objects. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Symposium on Computer Animation* (Aug 2004). 5

[MPT99] MCNEELY W. A., PUTERBAUGH K. D., TROY J. J.: Six degree-of-freedom haptic rendering using voxel sampling. In *SIGGRAPH* (1999), pp. 401–408. 2

[MSB*04] MORRIS D., SEWELL C., BLEVINS N., BARBAGLI F., SALISBURY K.: A collaborative virtual environment for the simulation of temporal bone surgery. In *MICCAI (2)* (2004), Barillot C., Haynor D. R.,, Hellier P., (Eds.), vol. 3217 of *Lecture Notes in Computer Science*, Springer, pp. 319–327. 2

[MSV95] MALLADI R., SETHIAN J., VEMURI B.: Shape modeling with front propagation: A level set approach. *IEEE Trans. PAMI 17*, 2 (1995), 158–175. 2

[MT96] MCINERNEY T., TERZOPOULOS D.: Deformable models, in medical image analysis: A survey. *Medical Image Analysis 1*, 2 (1996), 91–108. 2

[NvdS00] NIENHUYS H.-W., VAN DER STAPPEN A. F.: Combining finite element deformation with cutting for surgery simulations. In *EuroGraphics Short Presentations* (2000), de Sousa A., Torres J., (Eds.), pp. 43–52. 2

[OH99] O'BRIEN J. F., HODGINS J. K.: Graphical modeling and animation of brittle fracture. In *SIGGRAPH* (1999), pp. 137–146. 5

[PPT*02] PETERSIK A., PFLESSER B., TIEDE U., HÖHNE K. H., LEUWER R.: Haptic volume interaction with anatomic models at sub-voxel resolution. In *Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems* (2002), pp. 66–72. 2

[SC98] STEPHANE COTIN HERVE DELINGETTE N. A.: *Efficient Linear Elastic Models of Soft Tissues for real-time surgery simulation*. Tech. rep., Institut National de Recherche en Informoatique et en Automatique, 1998. 2

[THM*03] TESCHNER M., HEIDELBERGER B., MÜLLER M., POMERANETS D., GROSS M.: Optimized spatial hashing for collision detection of deformable objects. In *Proceedings of the Conference on Vision, Modeling and Visualization 2003 (VMV-03)* (Berlin, Nov. 19–21 2003), Ertl T., Girod B., Greiner G., Niemann H., Seidel H.-P., Steinbach E.,, Westermann R., (Eds.), Aka GmbH, pp. 47–54. 4