Generalized Interactions Using Virtual Tools within the *Spring* Framework: *Cutting*

Cynthia D. Bruyns [1,2] Kevin Montgomery [1]

[1] *Center for Bioinformatics, NASA Ames Research Center, Moffett Field, CA 94035,*

[2] *National Biocomputation Center, Stanford University, Stanford, CA 94305*

Abstract

We present schemes for real-time generalized mesh cutting. Starting with the a basic example, we describe the details of implementing cutting on single and multiple surface objects as well as hybrid and volumetric meshes using virtual tools with single and multiple cutting surfaces. These methods have been implemented in a robust surgical simulation environment allowing us to model procedures ranging from animal dissection to cleft lip correction.

1. Introduction

Cutting is a common manipulation encountered in simulations such as surgical training, clothing design and CAD/CAM manufacturing. A number of techniques have been developed to simulate cutting surface and volumetric meshes [1-17]. The number of intermediate steps required to recreate the cutting procedure often limits the level of realism offered by these methods, but intermediate steps have historically been necessitated by the inability to interactively update the underlying topological changes on large meshes.

The common element missing from these previous cutting tools is the ability to represent various forms of cutting using realistic tools on irregular meshes in real-time. The goal of this paper is to demonstrate how by using a very simple scheme, one can model widely differing behaviors of virtual tools within a real-time surgical simulation environment.

2. Methods

Starting with the most basic form of cutting, the following sections will describe the implementation of various cutting tools in a virtual environment. Since we start with a very general cutting scheme, each tool is an extension of the most basic cutting method requiring very little "special purpose" routines in order to model various forms of cutting. This method allows for any arbitrary cut to be made within an virtual object, and can simulate cutting surface, layered surface or tetrahedral objects using virtual scalpel, scissors, and loop cautery tools.

The basic engine for the cutting routine is collision detection, collision response, deformable object solution, and user interface information. This paper will describe the first two phases, the reader is directed to [18] for a detailed description of the last two phases.

Collision Detection

When modeling a cutting instrument one can either pre-compute the sharp regions of the mesh or choose which areas will be allowed to cut based on inspection of the model and included as information within the object's data file. This information can be a list of edges or faces [19].

Even if an object is visually composed of faces, for the sake of collision detection, we can create bounding volumes around other primitives to directly obtain the information important to cutting. If we choose to ignore intersections away from the sharp regions of a cutting instrument, we can enclose only the sharp primitives, reducing the number of intersection tests. If we choose more than one primitive to be sharp, the bounding hierarchy will enclose all of the sharp primitives. Since most tools will have fewer sharp primitives than the number of primitives in the overall geometry, this dramatically reduces the number of intersections tests that are necessary at each iteration.

When modeling cutting tools that are only composed of edges it is possible to pass over objects due to sampling latency. To solve this problem, we can choose to detect collisions not on the edge, but on the surface swept by the edge [20]. Furthermore, by enclosing the swept surfaces of the sharp edges by bounding volumes [21], we can still exploit the benefits of a binary search tree.

Currently we are storing intersection information as collision pairs with pointers to the objects that were in collision, the point at which collision occurred, and the primitives that were intersected. This list of collision pairs is then passed to the collision response scheme of the tool and provides us with the necessary information to perform a probing or a cutting manipulation.

Collision Response

The selection of sharp edges automatically defines a cutting direction, that is, the directions that the object can be moved that cause the tool to perform the specified action. For example, motion of a scalpel along the cut direction allows the cutting action to be implemented, while motion out of the allowed range causes the object to perform the probing action [22].

Cutting

The decision whether to cut a primitive is dependent on its state. These states are stored as information in the primitive class and used during re-meshing. For example, if the primitive has not been intersected before, then the intersection is recorded and the primitive is said to be in the *start* state. If at the next iteration the primitive is still in collision, then it is thought to be in the *update* state. In subsequent iterations, if the primitive is no longer in collision, then it is said to be in the *move* state and the primitive is cut based on the configuration of face and edge intersections that have been stored previously. The primitive state is determined by tracking which primitives the cutting edge was intersecting at the last

iteration and checking that list against the list of primitives currently in collision. Figure 1 describes the cutting loop.

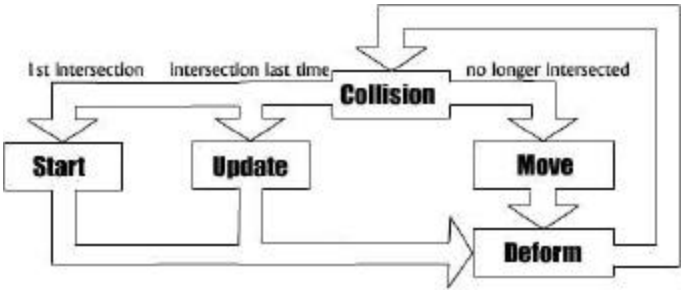

Figure 1. Flow diagram of the cutting loop.

As described in [18], each instrument has it's own dynamics and allowable behaviors. The following sections describe the results of implementing three kinds of instruments: a scalpel, a pair of scissors, and a cauterizing wire; on various mesh representations.

3. Results

On average, the simulation can detect collisions, compute the collision response, compute the deformation equations, update the bounding hierarchy and display the results on one processor of a Sun (Mountain View, CA) E3500 8x400 MHz UltraSparc workstation at 15 frames/second while the objects are intersecting. This frame rate increases to 70 frames/second if we display on one thread and run the simulation on another.

Scalpel Cutting

Single Surface

Figure 2 demonstrates the use of a single sharp edge to cut a single deformable object composed of 5,000 triangles.

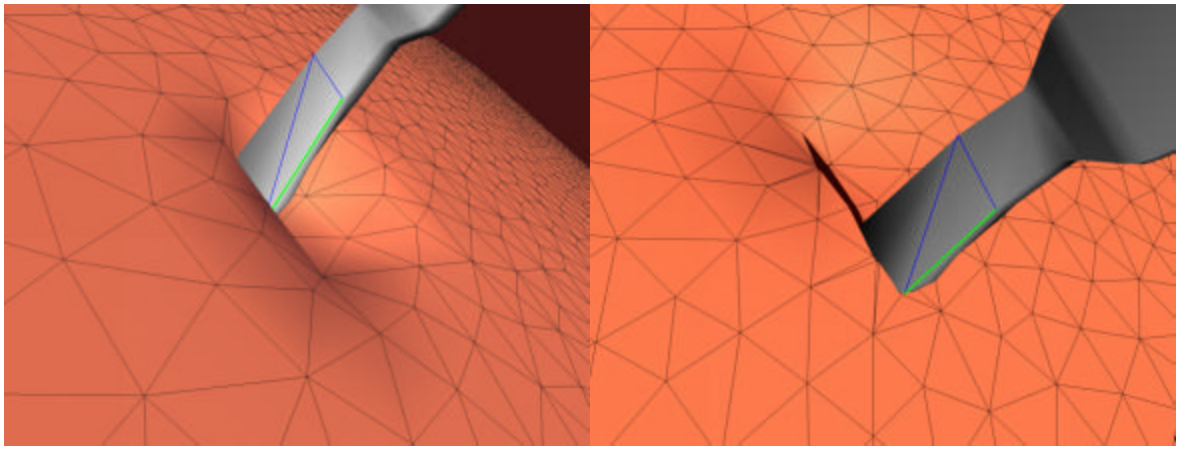

Figure 2. Edge states. [Left]: Surface deforms until the yield limit is reached. [Right]: The sharp edge cuts as the user moves the scalpel.

Multiple Surfaces

Because cutting is based on primitive states and not an object-wide state, it is possible to cut complex surfaces with folds, and objects that model a volume by extruding a surface thereby creating two surfaces connected by springs [23]. In these cases, the list of last primitives intersected might contain non-adjacent triangles and triangles with variable compliance and attributes. These triangles might also come from different objects. Figure 3 demonstrates the use of a single primitive to cut multiple surfaces. The deformable object is composed of 10,000 triangles.

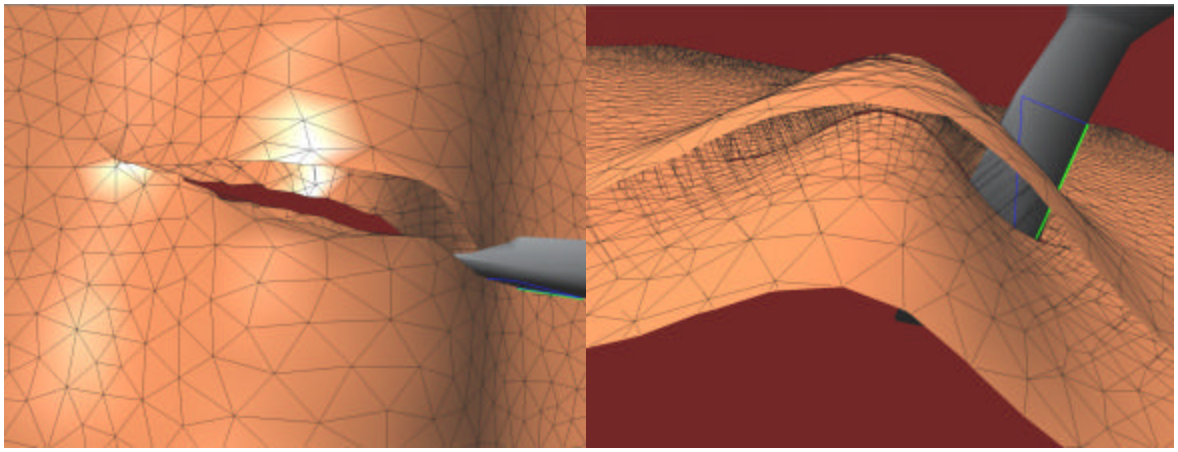

Figure 3. Cutting multiple surfaces. [Left]: Top View. [Right]: Side view.

Hybrid surface

In addition one might want to model separating a layer of a multiple surface object. In that case, one needs to use the hybrid method of collision detection enclosing the edges *and* faces of the surface with bounding spheres. In this case, the cutting scheme that is implemented is dependent on the type of primitive that is in collision. The triangular primitive is cut as before, however we choose to simply remove the intersected edges instead of subdividing it. This allows us to avoid creating edges with nodes that are not anchored and do not provide any structural or visual information to the model. Figure 4 shows a scalpel being used to cut a hybrid surface composed of 25,000 triangles.

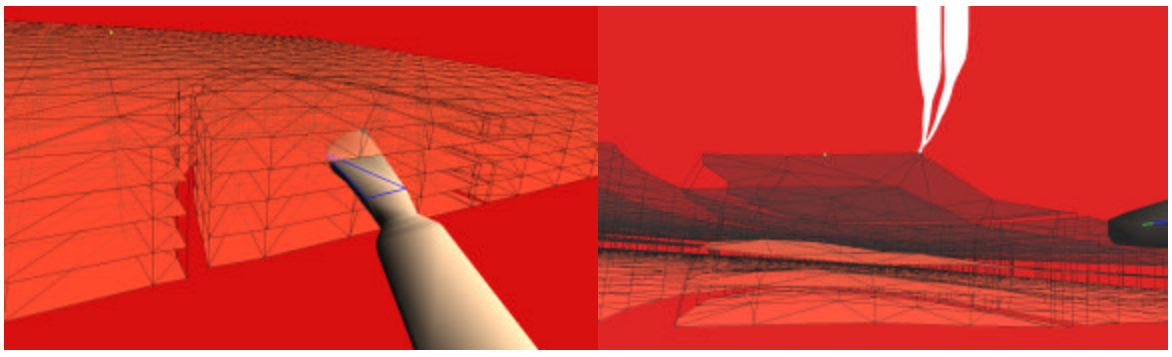

Figure 4. [Left]: Scalpel cutting away top layers of a multi-layered object. [Right]: Virtual forceps peeling back top layers.

Volume

When cutting an irregularly meshed volumetric object, tool motion is not as straightforward as moving across a surface. It is possible to be in several tetrahedra at once and when using

a tool that is more complex than a single long edge [11], it is possible to partially intersect tetrahedra requiring additional re-tetrahedralizing cases. Another factor to consider when using tetrahedra is that since we only record face and edge intersections, internal motion and depth of cut information is lost *within the tetrahedral primitive.* If these motions inside a tetrahedral primitive are important to the given simulation scenario, one might want to use a progressive cutting option. Progressive cutting re-tetrahredralizes the original primitive as the user moves the tool within the original tetrahedra, instead of waiting to re-tetrahedralize once the tool has left the original tetrahedra. One must be aware however, that taking each of these internal motions literally will result in an increase in the number of tetrahedra unless additional mesh condensation schemes are employed. Figure 5 shows a scalpel cutting an irregular mesh of 300 tetrahedra.

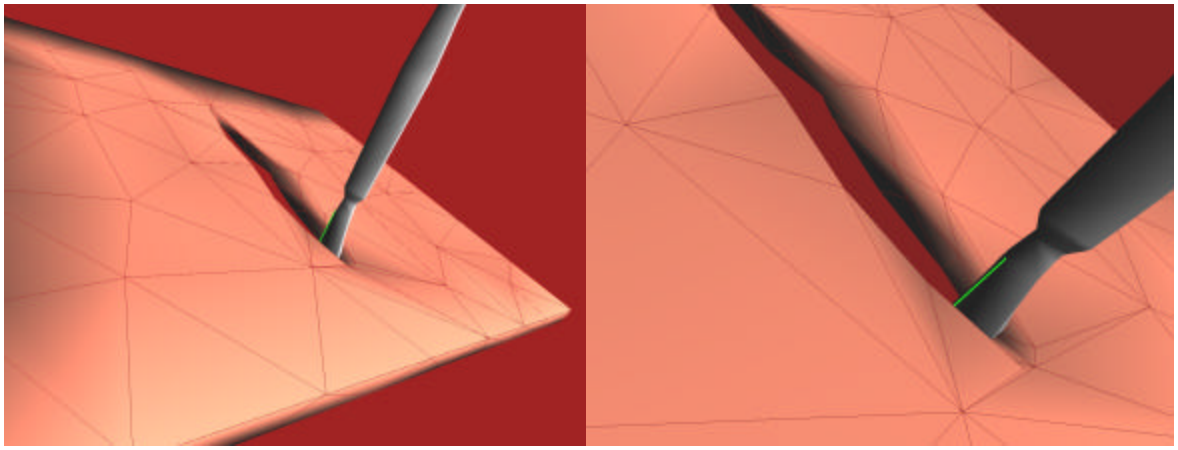

Figure 5. [Left]: A scalpel cutting a volumetric mesh. [Right]: Close-up on the cut path.

Scissors Cutting

When implementing scissors cutting, one must choose at least two edges as sharp. As mentioned previously, the allowable cutting direction is further restricted to permit cutting only when the cutting edges are moving towards each other. These two edges can either straddle the surface using the bottom edge to stabilize the surface while the top edge closes downward. Or if the edges are on the same side of the surface, the edges pinch the surface together until the two edges form a junction with the simulated tissue. Figure 6 shows a pair of virtual scissors cutting a deformable model consisting of 5,000 triangles. The figure also demonstrates how surface relaxes as the scissors are opened.

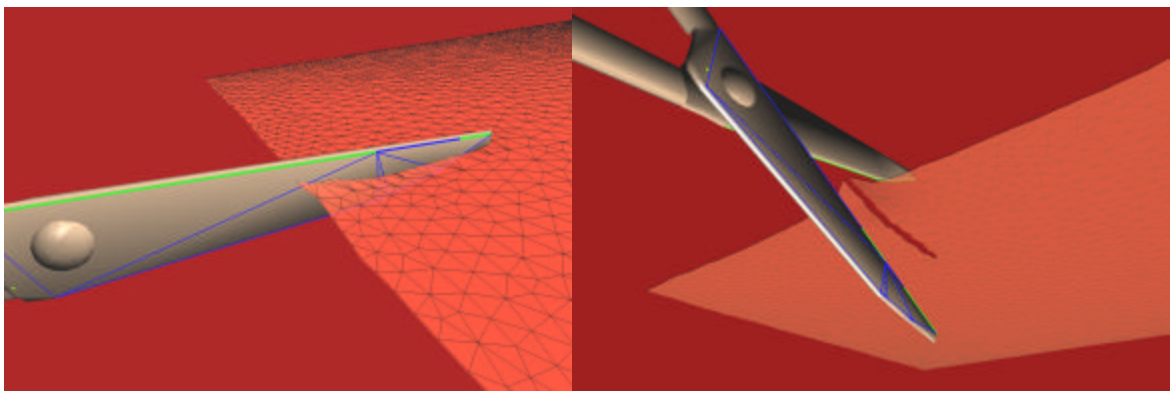

Figure 6. [Left]: Virtual scissors cutting. [Right]: Surface relaxing as scissors are opened.

Loop Cautery

When modeling a loop cautery tool, one needs to choose several edges as sharp. These edges have their own wire dynamics that must be modeled as well. Figure 7 shows a wire that has been assigned 20 cutting edges and is cutting a virtual polyp consisting of 5,000 triangles.

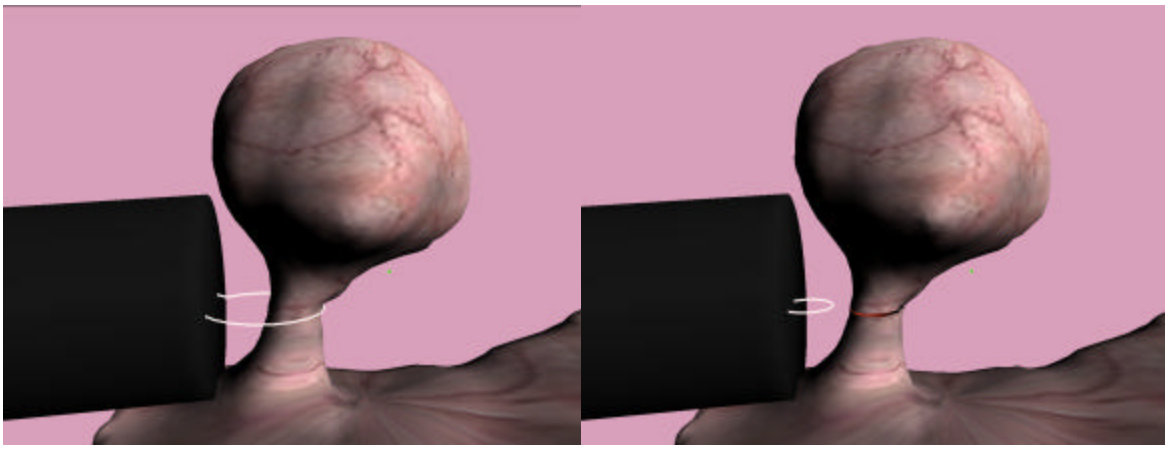

Figure 7. Loop cautery cutting. [Left]: Snaring a virtual polyp. [Right]: Resulting cut.

4. Conclusions

Creating cuts through very large meshes is extremely simple using the schemes presented in this paper. These schemes have been employed in a rat dissection simulation system [24], a virtual polypectomy simulator [25], and a virtual hysteroscopy simulation system [26].

Acknowledgements

We wish to thank Richard Boyle and the Center for Bioinformatics at the NASA Ames Research Center for their support of this research. Special thanks to the National Biocomputation Center and including Joel Brown, Steven Sorkin, Fredric Mazzella, Anil Menon, Jeremie Roux, Julien Durand, Bharat Beeedu, Guillame Thonier and Jean Claude Latombe. This work was supported by grants from NASA (NCC2-1010), NIH (NLM-3506, HD38223), NSF (IIS-9907060), and a generous donation from Sun Microsystems.

References

[1] Pieper, S., Rosen, J., Zeltzer, D.: Interactive Graphics for Plastic Surgery: A Task-level Analysis and Implementation. Symposium on Interactive 3D Graphics ACM Press, New York, (1992) pp.127–134.

[2] Song, G. and Reddy, N.: Tissue Cutting in Virtual Environments. In: Medicine Meets Virtual Reality. IOS Press, Amsterdam (1995) pp. 359-364.

[3] Keeve, E., Girod, S., and Girod, B.: Computer-Aided Craniofacial Surgery. Journal of the Int. Society for Computer Aided Surgery. 3 (1996) 6-10.

[4] Mazura, A., Seifert, S.: Virtual Cutting in Medical Data. In: Westwood, J., et. al. (eds.) Medicine Meets Virtual Reality. IOS Press, Amsterdam (1997) pp. 420-429.

[5] Wong, K. C., Siu, T. Y., and Heng, P.: Interactive Volume Cutting. Technical Report, Department of Computer Science and Engineering, Chinese University of Hong Kong (1998).

[6] Basdogan, C., Ho, C., and Srinivasan, M.A.: Simulation of Tissue Cutting and Bleeding for Laparoscopic Surgery Using Auxiliary Surfaces. In: Westwood, J., et. al. (eds.) Medicine Meets Virtual Reality. IOS Press, Amsterdam (1999) pp. 38-44.
[7] Bro-Nielsen, M., Helfrick, D., Glass, B., Zeng, X., Connacher, H:. VR Simulation of Abdominal Trauma Surgery. Westwood, J., et. al. (eds.) Medicine Meets Virtual Reality, IOS Press, Amsterdam (1999) pp.117-123.
[8] Voss, G., Hahn, J.K., Muller, W., Lineman, R.W.: Virtual Cutting of Anatomical Structures. In: Westwood, J., et. al. (eds.) Medicine Meets Virtual Reality, IOS Press, Amsterdam (1999) pp.381-383.
[9] Beisler, D., Gross, M.: Interactive Simulation of Surgical Cuts. Proceedings of Pacific Graphics, IEEE Computer Society Press, (2000) pp. 116-125.
[10] Neumann. P.: Near Real-Time Cutting. Siggraph 2000, Sketches and Applications. New Orleans, La. (2000).
[11] Ganovelli, F., Cignoni, P., Montani, C., Scopigno, R.: A Multiresolution Model for Soft Objects Supporting Interactive Cuts and Lacerations. Proceedings of the 21st European Conference on Computer Graphics. Blackwell, Cambridge (2000) pp. 271-282
[12] Mor, A., Kanade, T.: Modifying Soft Tissue Models: Progressive Cutting With Minimal New Element Creation. In: Niessen, W.J. et. al. (eds.) MICCAI 2000, LNCS 1935, Springer, Berlin Heidelberg, (2000) pp. 598-607.
[13] Schutyser, F., Van Cleyenbreugel, J., Nadjmi, N., Schoenaers, J., Suetens, P.: 3D Image-Based Planning for Unilateral Mandibular Distraction. In. Heinz. U., et. al. (eds.) Computer Assisted Radiology and Surgery, Elsevier, Amsterdam (2000) pp.899-904.
[14] Bruyns, C., Senger. S.: Interactive Cutting of 3D Surface Meshes. Computer and Graphics, 25 (2001) 635-642.
[15] Nienhuys, H-W., van der Stappen, A.F.: A Surgery Simulation Supporting Cuts and Finite Element Deformation, In: Niessen, W.J., et. al. (eds.) MICCAI 2001, LNCS 2208, Springer, Berlin Heidelberg, (2001) pp. 145-152.
[16] Meier, U., Monserrat, C., Parr, N-C., Garcia, F.J., Gil, J.A.: Real-Time Simulation of Minimally-Invasive Surgery with Cutting Based in Boundary Element Methods. In: Niessen, W.J., et. al. (eds.) MICCAI 2001, LNCS 2208, Springer, Berlin Heidelberg, (2001) pp. 1263-1264.
[17] Serby, D., Harders, M., Szekely, G.: A New Approach to Cutting into Finite Element Models. In: Niessen, W.J., et. al. (eds.) MICCAI 2001, LNCS 2208, Springer, Berlin Heidelberg, (2001) pp. 425-433.
[18] Montgomery, K., Bruyns, C., Brown, J., Sorkin, S., Mazzella, F., Thonier, G., Tellier, A., Lerman, B., Menon, A.: Spring: A General Framework for Collaborative, Real-time Surgical Simulation, In: Westwood, J., et. al. (eds.): Medicine Meets Virtual Reality, IOS Press, Amsterdam, (2002).
[19] Bruyns, C., Senger, S., Wildermuth, S., Montgomery, K., Boyle, R.: Real-time Interactions Using Virtual Tools. In: Niessen, W.J., et .al. (eds.) MICCAI 2001, LNCS 2208, Springer, Berlin Heidelberg, (2001) pp. 1349-1351.
[20] Boyse, J.W.: Interference Collision Detection Among Solids and Surfaces. Communications of the ACM 22 (1979) 3-9.
[21] Sorkin, S.: Distance Computing Between Deformable Objects. Honors Thesis, Computer Science Department, Stanford University, (2000).
[22] Bruyns, C., Montgomery, K.: Generalized Interactions Using Virtual Tools within the *Spring* Framework: *Probing, Piercing, Cauterizing and Ablating*, In: Westwood, J., et. al. (eds.): Medicine Meets Virtual Reality, IOS Press, Amsterdam, (2002).
[23] Mazzella, F.: Auto Acquisition of Elastic Properties for Surgical Simulation, http://biocomp.stanford.edu/papers/
[24] Bruyns, C., Montgomery, K., Wildermuth, S.,: A Virtual Environment for Simulated Rat Dissection. In: Westwood, J., et. al. (eds.) Medicine Meets Virtual Reality, IOS Press, Amsterdam (2001) pp. 75-81.
[25] Wildermuth, S., Bruyns, C., Montgomery, K., Marincek, B., Virtual Colon Polyp Extraction (Simulation and Preoperative Planning), In: Niessen, W.J., et .al. (eds.) MICCAI 2001, LNCS 2208, Springer, Berlin Heidelberg, (2001) pp. 1347-1348.
[26] Montgomery, K.; Heinrichs, L., Bruyns, C., Wildermuth, S., Hasser, C., Ozenne, S., Bailey, D.,: Surgical Simulator for Operative Hysteroscopy and Endometrial Ablation, In: Lemke, H., et. al. (eds.) Computer-Aided Radiology and Surgery, Elsevier, Amsterdam (2001) pp. 79-84.