3rd Workshop in Virtual Reality Interactions and Physical Simulation "VRIPHYS" (2006)
C. Mendoza, I. Navazo (Editors)

Virtual Resectoscope Interface for a Surgery Simulation Training System of the Prostate

Miguel A. Padilla, Felipe Altamirano del Monte and Fernando Arambula Cosio

Image Analysis and Visualization Lab. CCADET, UNAM, P.O.Box 70-186, Mexico, D.F., 04510

Abstract

In this work is presented the current state of the development of a virtual resectocope interface for a surgery simulation system for training Transurethral Resection of the Prostate (TURP). The interface consists of two parts, the first part is a mechatronics device that emulates a real resectosope and allows to perform the most important movements of the surgical tool during a TURP. The second part is a software interface that consist on a collision detection mechanism that allows to calculate in real-time the interactions between the mechatronic device and the deformable tissue model of the prostate, in order to simulate tissue resection and deformation. The current prototype has five degrees of freedom, which are enough to have a realistic simulation of the surgery movements. The results show that the interface is suitable for a real-time surgery simulation training system of the prostate without force feedback.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Physically based modeling, Virtual reality

1. Introduction

The prostate gland is located next to the bladder in human males, with the urethra running from the bladder neck through the prostate to the penile urethra (Figure 1). A frequent condition in men above 50 years old is the benign enlargement of the prostate known as Benign Prostatic Hyperplasia (BHP), which in some cases results in significant blockage of the urinary flow. The standard surgical procedure to treat a hypertrophied prostate gland is the Transurethral Resection of the Prostate (TURP). It essentially consists of the removal of the inner lobes of the prostate in order to relieve urinary outflow obstruction. Mastering the TURP technique requires a highly developed hand-eye coordination which enables the surgeon to orientate inside the prostate, using only the monocular view of the lens of the resectoscope. Currently TURP is taught through example from an experienced surgeon. The resident of urology has very restricted opportunity to practice the procedure.

Gomes et al. [GBTD99] reported an interesting hybrid computer-assisted training system for TURP, that use a resectable physical phantom and a computer model of the prostate. The aim of using traditional physical phantoms is to provide the surgeon with more realistic haptic feedback than virtual reality techniques can provide. Positional feedback is provided by an optical tracker and the 3D computer model of the prostate. However, the 3D model used is a rigid geometric model of the prostate capsule, without deformation and resection simulation. Additionally, a phantom based simulator requires continuous expense on replacements.

Manyak et al. [MSH*02] reported the construction of a virtual reality surgery simulation system of the lower urinary track. They considered only the surface of the urinary track, reconstructed from the Visible Human Project dataset [VPH] with texture mapping for visual realism. However, the prostatic urethra behaviour depends on the conditions of the tissue from the capsule to the urethra. As a consequence, a volumetric model of the prostate should be consider in order to simulating realistic TURP procedures. Sweet, et al. [SKO*04] reported the experience of using the TURP virtual training system described in [OGW*01]. They studied the effectiveness of translating the skills acquired in their virtual environment to the operating room. The surgery simulation system for TURP reported in [OGW*01] uses an image-based approach for simulating bleeding when the resecting loop contacts surface vessels. The loop triggers precalculated movies of blood flow, that is then oriented and mapped on the virtual environment. However, how they model tissue deformation is not clear; moreover as in [MSH*02] they used only the surface representation of the urethra.

In [ACPCSM06, PCAC04] we reported the development a 3D deformable volumetric model of the prostate for TURP simulation that involves tissue deformation and resection, considering the gland as a viscoelastic solid. In this work we describe the current state of the development of the virtual resectoscope interface for our simulator. Section 2 of this paper describes the developmente of the mechatronic interface that emulates the resectoscope; section 3 describes the collision detection scheme between the virtual resectoscope and the tissue model; in section 4 we present the results of the development of the mechatronic device and the collision detection scheme; finally in section 5 we present the conclusions and future perspectives of this work.

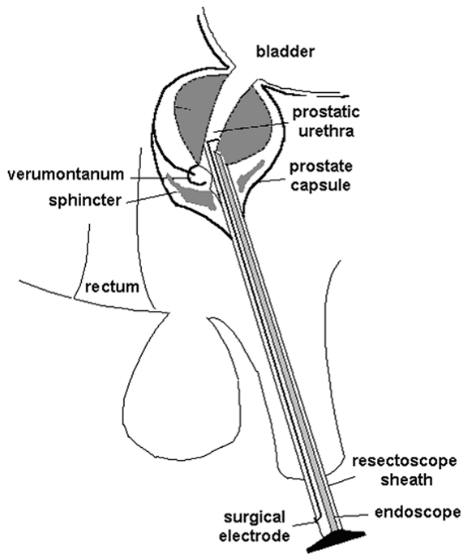


Figure 1: *Prostate gland position.*

2. Mechanical design of the interface

In order to obtain a realistic simulation of the most important movements of the surgeon during a TURP, a mechanism was designed based on a disk-ring array (Figure 2). Due to difficulties in repreducing the seven degrees of freedom, we decided at the moment to reproduce only the five most important degrees. In this manner the mechanism provides five axes of movement. Three of these axes are rotational and the other two are linear displacements of the resectoscope (Figure 3). We ignore for the moment two additional translation degrees to the resectoscope sheath The disk-ring array is mounted in a box of plastic material. Inside the box will be placed and registered a phantom of the prostate constructed

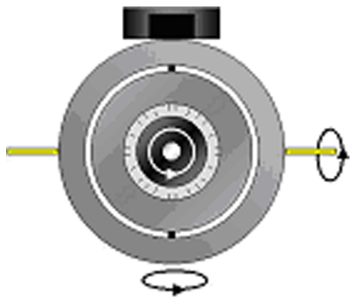


Figure 2: *Disk-ring array system.*

from the 3D model of the gland c̃itePadilla04, this phantom is the physical reference of the 3D computer model and also allows to limit the resection volume during the simulation.

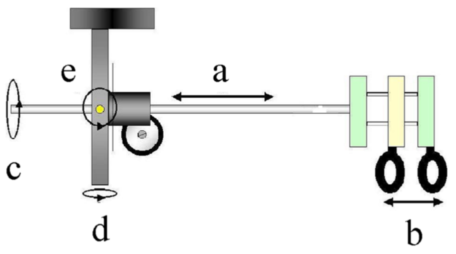


Figure 3: *Degrees of freedom of the resectoscope; a and b are linear; c, d and e are rotational.*

2.1. Axes movement sensors

2.1.1. Rotational axes

Optical encoders are used in order to sense each of the three rotational movements. Each encoder is placed in each rotation axe, so with this arrangement we can measure the direction and the angle rotated by the user on each axis (Figure 4). The output signals of the encoders are a couple of TTL-level trains of pulses with varying width and phase, according to the direction of rotation and speed of the axis. The encoders are powered with a voltage of 5V at 27mA.

2.1.2. Linear axes

The linear displacement (in/out) of the surgical tool is measured with a linear precision potentiometer of 10 k(. The shaft of the potentiometer has a metallic disk with a rubber band which is in contact with the cylindrical body of the resectoscope. The output voltage of the potentiometer varies according to the position of the resectoscope. For an input voltage of 5 V, the output varies from 3.23 to 4.50 V, which corresponds to 23 cm of useful resectoscope displacement. The resecting loop has a linear movement of 36 mm, this distance is sensed with an array of two Hall effect sensors

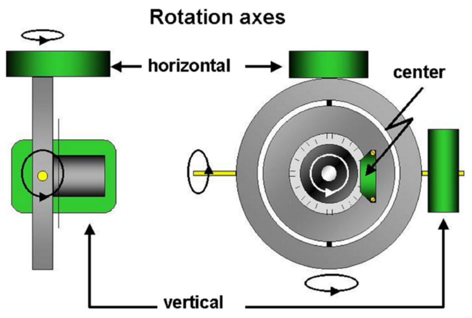


Figure 4: *Digital encoders to sense rotational axes of the resectoscope.*

and two permanent magnets, as shown in Figure 5. The Hall effect sensors are linear and are powered with voltage of 5V at 9mA.

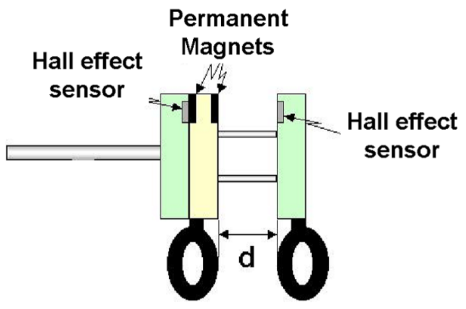


Figure 5: *Hall effect sensors array for controling the resecting loop.*

2.2. Sensors

The signals obtained from the sensors are processed with an LP3500 card (www.zworld.com), which is a low-power, single-board computer with a Rabbit 3000 8 bit microcontroller at 7,4 MHz. The programs were written in C language with the Dynamic C compiler [Fox03].

2.2.1. Optical Sensors

Each optical sensor give us three output signals: CH. A, CH. B and CH. I (as shown in Figure 6). We will not use signal CH. I, since this signal is useful only when the optical sensor gives a complete turn, however our three sensors will never turn completely. To determine the position and direction of rotation, channels A and B are enough. A program was developed for calculating from the two out of phase signals, the angular position of each rotational axis. The signals are acquired with six digital channels of the LP3500 card.

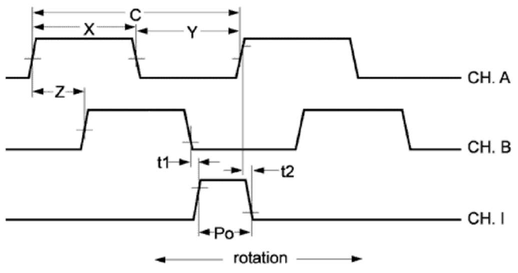


Figure 6: *Signals of each digital encoder for rotational movements of the sheath.*

The program is based on storing the previous value given by the sensor to the microcontroller and comparing it with the present value, with the aim to detect any change in the phase between CH. A and CH. B. The angular distance that the sensor moved is determined by an interruption routine of the microcontroller that counts the number of pulses of CH. A.

2.2.2. Multi-Turn Potentiometer

The acquisition of the potentiometer output signal is made through one of the analog channels of the microcontroller card. The configuration of the potentiometer is a voltage divider with a voltage of 5V for measuring the resecting loop movements.

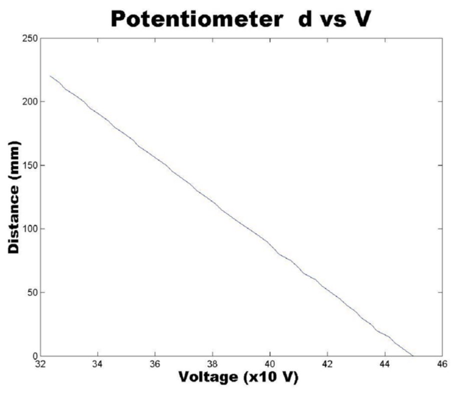


Figure 7: *Signals of Linear model of the potentiometer for translational movements of the sheath.*

The signal is in the interval from 3.23 to 4.50V that corresponds to the movement of the resectoscope from 0 to 23cm. The mathematical model of this sensor corresponds to a linear device (Figure 7) as can be observed in Equation (1). This equation was directly programmed as a routine of the microcontroller because doing interpolation using a linear model is very efficient.

$$d = -17.386 \times V + 785.95 \tag{1}$$

2.2.3. Hall Effect Sensors

The output signal of the Hall effect sensors array is measured through two analog channels of the LP3500, the program in C is in charge of signal acquisition, addition and final displacement calculation, through interpolation using a table. The mathematical model obtained is nonlinear (Figure 8) and at off-line stage, a table with 148 interpolated values that corresponds to a resolution of 0.5mm of the nonlinear curve were calculated, in order to obtain in real-time the displacement of the resecting loop. Finner displacements behind 0.5mm are calculated in real-time by doing linear interpolation of the precalculated values.

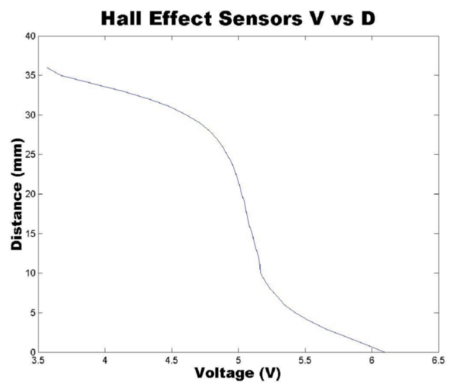


Figure 8: *Characterization of the Hall effect array for movements of the resecting loop.*

2.3. Communication with the graphical model

We are working on the communications between the mechatronics interface and the virtual computer resectoscope model. The real-time resectoscope movements consequently will must reflect the interaction between the surgeon and the tissue model. The interactions (tissue deformation and resection) between the virtual tool and the prostate model are consequence of the collision between them. The collision detection mechanism will be described in the next section.

3. Collision detection between the resectoscope and the prostate

Different approaches reported for collision detection between complex non-convex surface meshes relie on decomposing objects into a pre-computed hierarchy of bounding-volumes with convex shapes. In order to quickly discard large subsets of surface polygons that are far of any possible collision point. Common bounding volumes are: Axis Aligned Bounding Boxes (AABBs), Oriented Bounding Boxes (OBBs) [GLM96] or spheres [Hub96, LM97, BSL*02]. Gottschalk et al. [GLM96] reported an efficient algorithm and a system, called RAPID, for collision detection based on OBBs which shows better performance than AABBs, but only in the scope of rigid motion transformations. Van Den Bergen presented in [VdB97] an algorithm that upgrade the AABBs approach with a performance closer to OBBs; the interest in Van Den Bergen is placed on the quick update of the AABB tree as the objects deform. Although the approximation of the geometry of high complex objects with spheres is less accurate than with AABBs, it seems faster to update the hierarchy for mesh topology modification after tissue cuttings. Brown et al. [BSL*02] reported a modified Quinlan algorithm [Qui94] that uses a sphere-tree representation of objects and proved the suitability of their method in the field of real-time tissue deformation simulation.

Two interesting works, Lombardo et al. [LCN99] and Wagner et al. [WSM02], use an image-based approach that use graphic hardware calculations. These techniques use the rasterizing process that maps the scene objects coordinates into the camera coordinate system, clip the tetrahedral faces outside the viewing volume, and project the remaining visible polygons into rasterized pixels. In this way, objects that are not rasterized at the same pixel do not collide. Although this method is simpler and faster, realistic soft tissue simulation must consider those areas that even thought are not visible, are physically in contact and affect the global behavior of the soft body. Unfortunately, projection loses important tridimentional information about the mesh topology useful for calculating the penetration field of the tool penetrating the organ.

An important paper presented by Teschner et al. [TKH*04] contains a complete summary of the state of the art in collision detection for deformable objects. They made a review of different approaches that includes bounding volume hierarchies, distance fields, image-space techniques, and stochastic methods, and presented applications in cloth modeling and surgery simulation.

In this section we the present the collision detection mechanism of our simulator based on the representation of objects with hierarchical sphere-trees. Our implementation detects all collisions between the resectoscope and the prostate in real-time and allows the hierarchical structure updating for mesh modification after tissue cuttings.

3.1. Hierarchical sphere-tree construction

Our approach uses the sphere-tree structure for objects representation. Using spheres as a primitive bounding-volume seems a better representation in the field of surgery simulation due to the simplicity for updating the structure after object deformations, and as we explain later, for cutting.

Figure 9 shows the sphere-tree construction algorithm. The idea is to find an almost balanced binary tree ST, where

```
Algorithm CreateSphereTree( S )
1. L←CreateLeafSpheres( S )
2. root[BT]←CreateBoxChild( NIL )
3. leafs[root[BT]]←L
4. DivideLeafs( root[BT] )
5. SphereFromBox( BT )
6. ST←sphere[root[ST]]
7. return ST
```

Figure 9: *Algorithm for building the hierarchical representation of objects with bounding spheres.*

the leaf spheres completely contain the surface S of the object and node-spheres completely contain the spheres of its descendant nodes. The sphere-three building starts by covering all polygons of the object surface S with a set of small leaf spheres L. The sphere-tree is constructed by recursively spliting S into two approximately subsets of leaf spheres at each recursion. For this purpose, we used an auxiliary bounding box-tree BT, where at each recursion a bounding-box for covering the leaf spheres is computed, and the spheres are divided through the orthogonal plane placed at the middle of the principal component of the box. The root of BT contains all leaf-spheres in L, and every node n in BT has exactly two children left and right, such that $|leafs[n]| = |leafs[left]| \cup |leafs[right]|$. The ST is found by walking BT in preorder, where at each recursion the algorithm creates a sphere in ST that covers all the leaf spheres of L contained by the box node in BT. At each recursion the radius and center of the sphere needed to completely cover all its descendants leaf spheres is calculated; this criteria guarantees by induction that every node covers all the leaf spheres of its descendant nodes and allows efficiently updating the tree after mesh deformations or cuttings. Figure 10 illustrates the binary sphere-tree structure, where every node covers the leaf spheres of its descendants.

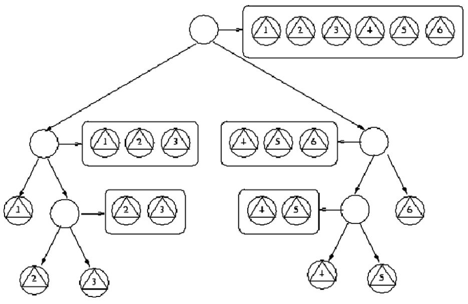


Figure 10: *Binary tree structure of an object surface with 6 triangles. Each node covers at least two leaf spheres and its respective triangles. The root of the tree covers all the sphere leafs of its descendant.*

3.2. Collision detection algorithm

Figure 11 shows the algorithm for collision detection that uses the sphere-tree of the last section. If in step 4 two spheres s_1 and s_2 are leafs, both s_1 and s_2 are stored in the colliding sphere lists: s_1 in L_1 and s_2 in L_2. If s_1 belongs to object o_1 and s_2 to object o_2 or vice versa (step 5). If in step 6 the triangles of s_1 and s_2 intersect, the collision between triangles is stored in the list C of collisions (steps 7-9). Otherwise, if during step 4, s_1, s_2 or both are not leafs, the algorithm continues the recursion until two colliding leafs are found. Since we do not want to stop recursion after the first collision is detected, the algorithm must walk thorough ST_1 and ST_2 taking one of eight different paths at each step, as follows.

```
Algorithm DetectCollision(o1,o2,s1,s2,C)
1.  If Intersection( s1, s2 ) = FALSE Then return
3.  Else Then
4.    If IsLeaf(s1)=TRUE and IsLeaf(s2)=TRUE Then
5.      Register(s1,s2, L1, L2)
6.    If Intersection(triangle[s1], triangle[s2])=TRUE
7.      c←CreateCollision( triangle[s1], triangle[s2] )
8.      Insert( C, c )
9.      return
10.   Else Then
11.     If IsLeaf(s1)=FALSE Then sx←s1, sy←s2
12.     Else Then sx←s2, sy←s1
13.     n1←left[sx], n2←right[sx]
14.     If IsLeaf(n1)=FALSE and IsLeaf(n2)=FALSE
        Then
15.       d1←distance(center[sy],center[n1])
16.       d2←distance(center[sy],center[n2])
17.       If d1 > d2 Then n1←right[sx], n2←left[sx]
18.     If n1 <> NIL Then
19.       If Intersection(sy,n1)=TRUE Then
20.         r1← radius[sy], r2← radius[n1]
21.         If r1<r2 Then DetectCollision(o1,o2,n1,sy,C)
22.         Else Then     DetectCollision(o1,o2,sy,n1,C)
23.     If n2 <> NIL Then
24.       If Intersection(sy,n2)=TRUE Then
25.         r1← radius[sy], r2← radius[n2]
26.         If r1<r2 Then DetectCollision(o1,o2,n2,sy,C)
27.         Else Then     DetectCollision(o1,o2,sy,n2,C)
```

Figure 11: *The collision detection algorithm.*

In steps 11-13 the algorithm selects the sphere sx to explore its children n_1 and n_2, and the base sphere s_y as: if s_1 is not a leaf, the algorithm must continue the recursion from s_1 (step 11), otherwise continues from s_2 (step 12). However, if in steps 14-17 both spheres s_1 and s_2 are not leafs, then the algorithm selects the first child n_1 to continue, where n_1 is the children of n_x closer the colliding sphere s_y.

At steps 18-22 the algorithm walks through paths (n_1, s_y) or (s_y, n_1), depending on the sizes of n_1 and s_y: if s_y is smaller than n_1, walks through n_1 (step 21); if s_y is bigger than n_1, walks through s_y (step 22). In steps 23-27 after covering all paths from n_1, the algorithm walks through n_2 in the same manner as for n_1.

In this way, the algorithm tends to explore first those regions of objects that are more likely to collide and therefore tends to avoid unfruitful paths. At the end, if the list C of collisions is empty, there are no collisions detected. On the other hand, if C is not empty, the list L_1 and L_2 contains the colliding leafs that will provide, at the post-collision stage, historic information for penetration field and collision response calculation with respect of the penetrating volume of the tool into the soft tissue body, as will be explained in section 3.3.

As step 5 could be made in constant time $O(1)$ by turning on a flag for each s_1 and s_2, indicating that s_1 and s_2 has been previously visited and inserted in L_1 and L_2, and considering that ST_1 and ST_2 are almost balanced binaries trees, the complexity of the algorithm is $O(n\log n)$, where n is the number of spheres in ST_1 and ST_2.

3.3. Collision response

After a collision is detected the soft tissue must slightly deform before the tissue resection occurs. Tissue deformations result from the reacting forces produced after the collision. For computing the external reacting forces we used a penalty-based method, where reacting forces are the forces needed to separate the penetrating objects. These forces depends on the penetration depth field and the stiffness and damping properties of the soft model [DJL98]. For calculating the penetration field we use the history of the last collision stored in the colliding sphere lists L_1 and L_2 computed as explained in section 3.2. From L_1 and L_2 we extract the submeshes in contact S_1^c and S_2^c and we obtained the signed distance field of submesh S_2^c (the resectoscope) with respect S_1^c (the prostate) [BA02], in order to use the distance field as measure of penetration; the vector field of the distances is used as the direction of the reacting forces and the sign is used to discriminate the triangles of the resectoscope in contact that does not penetrate the prostate from the penetrating ones. The deformable behaviour of the prostate is modeled with the mass-spring method, where the soft body is discretized as a set of nodal masses interconected by springs and dampers [PCAC04].

3.4. Sphere-Tree updating for mesh deformation

Due to the form that the sphere-tree is constructed every sphere contains, by induction, the leafs of its descendants. For this reason, updating operations are efficiently performed by bottom-up recursively updating of the center and the radius of the parents, from the leaf that belongs the modified triangle to the root of the tree.

3.5. Sphere-Tree updating for mesh cutting

For the moment, tissue resection is done without local mesh refinement, so the sphere-tree cutting is performed as two operations: removing triangles from surface mesh and adding the inner triangles exposed by the cut [PCAC04]. For this reason the collision detection mechanism should provide two basics operations for mesh topology modifications, adding and deleting triangles from the sphere-tree. Deleting triangles to the tree consists on: 1) removing the leaf to be cut and its parent and linking its brother to its grandparent; 2) recursive bottom-up updating the sphere geometry of the ancestors of the triangle removed, as described in section 3.3. Since the sphere-tree is an almost balanced hierachy, adding a triangle simply consists on: 1) creating a new leaf and inserting operation on binary trees at the lowest level position, where the searching criteria is the principal component of spheres at each recursion; 2) upditing again the sphere-tree geometry as described in point 3.3. The complexity of inserting a triangle and updating the sphere-tree after modifying or deleting a triangle is $O(\log n)$, where $\log n$ is the height of the tree. Although techniques for tetrahedral mesh cutting has been reported [BGTG04,GCMS01,BMG99], adapting these techniques that performs tissue cutting for scalpels seems too difficult for carving of small tissue chips, as it occurs during TURP resections. For this reason, we are currently working on producing tissue resections by local mesh refinement based on regular 8-subtetrahedron subdivision techniques [AB96] in order to increase the resolution around the resecting zone and reduced the decimation effect of removing volume elements around the cutting radius of the resecting element [PCAC04].

4. Results

Figure 12 shows two views of the mechatronic device that emulates the resectoscope. The device reproduce the five main movements of a real resectoscope, including the resecting loop movements. The three optical sensors and the multi-turn potentiometer measure the rotation and traslation of the resectoscope sheath accordingly, while the Hall effect sensors array measure the resecting loop displacements. All measurements are monitoring in real-time with the LP3500 card. We are currently working in transforming the sensor measurements into movements commands from the LP3500 to the collision detection mechanism.

We modeled the virtual resectoscope as two objects, the loop shifting inside the sheath. The number triangles of the loop and the sheath are 330 and 158 accordingly (488 triangles). The prostate model were inplemented as a deformable mass-spring system at different resolution (variable number of mesh triangles from 1296 to 8640 triangles). Figure 13 shows the response rate of the tests that we performed with the prostate models; the response rate runs from 50 to 60 Hz on a 3GHz PC Computer with 512 Mb of memory, which indicates that the algorithm is suitable for real-time interac-

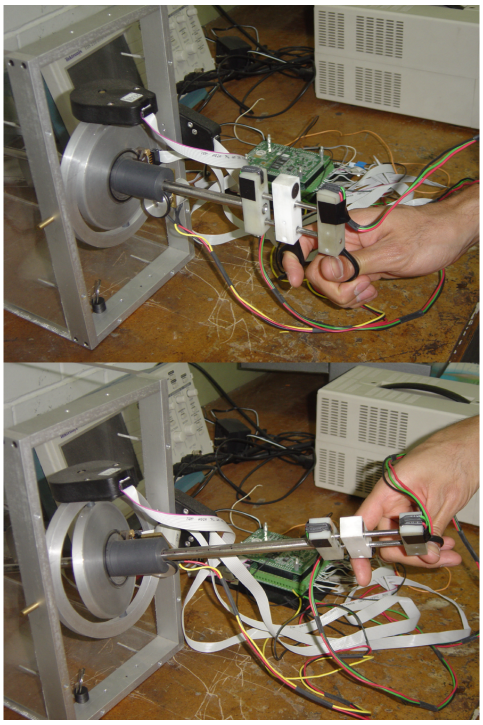


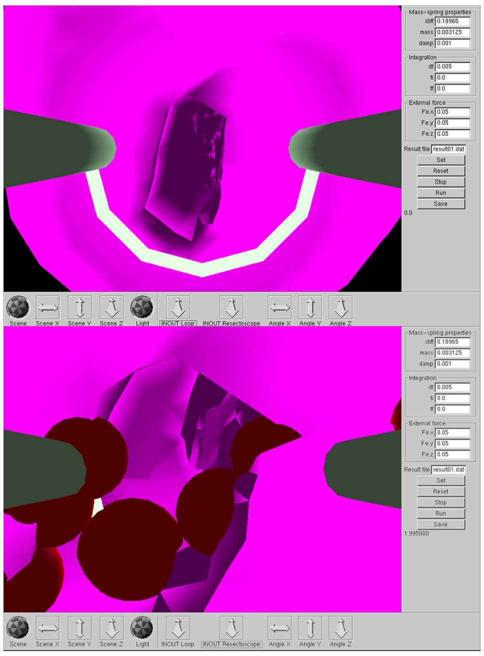
Figure 12: *Two views of the mechatronic interface that illustrates the sheath and loop movements.*

tions and gives the enough time rate for rendering and tissue deformation calculation.

Prostate mesh triangles	Collision detection response mean rate [Hz]
1296	**64.09**
1482	**64.08**
1560	**64.25**
2069	**64.58**
2816	**63.35**
3128	**63.15**
3744	**51.45**
5292	**52.48**
5670	**55.28**
7260	**50.93**
8640	**52.23**

Figure 13: *Response rate of the collision detection algorithm between a virtual resectoscope with 488 triangles and some prostate models with different number of triangles.*

Figure 14 illustrates the collision detection mechanism. After some movements, the loop of the virtual resectoscope collides with the prostate body. As a consequence, the penetrating field is calculated as the distance field of the resectoscope submesh with respect to the prostate submesh (the part of the surgical tool inside the prostate) and the proportional external forces are calculated and therefore the mass-spring system deforms.



Mass–spring properties
stiff 0.18965
mass 0.003125
damp 0.001
Integration
dt 0.005
ti 0.0
tf 0.0
External force
Fe.x 0.05
Fe.y 0.05
Fe.z 0.05
Result file result01.dat
0.0

Mass–spring properties
stiff 0.18965
mass 0.003125
damp 0.001
Integration
dt 0.005
ti 0.0
tf 0.0
External force
Fe.x 0.05
Fe.y 0.05
Fe.z 0.05
Result file result01.dat
1.995000

Figure 14: *Two views of the virtual environment that illustrates collision of the virtual resectoscope with the prostate model. The virtual resectoscope must follow in real-time the movements of the mechatronic interface.*

5. Conclusions

In this paper we have presented the current state of the development of a virtual resectoscope interface for a surgery simulation system of the prostate, without force feedback for the moment. Comparing the papers presented in [OGW*01, MSH*02] our system models the prostate as a viscoelastic solid and not only the urethra surface, we are also working on a tissue resection mechanism for removing small volumetric tissue chips. With the system of sensors installed in the mechatronics interface good results have been obtained comparing the measurements made with the card LP3500 and the measurements done in our laboratory (using a Vernier caliper), nevertheless we will have to do a metrological evaluation to obtain accurate specifications of the mechanic parts. During the tests made with the system of acquisition of signals problems did not appear, it mean we obtained correct readings to the real displacements, nevertheless when programming all the routines in a single program on card LP3500 were observed delays and lost of some data of the sensors. Therefore more work must be made in order to evaluate the effects of these data lost on the simulation. We implemented a collision detection algorithm based on sphere-trees for detecting in real-time the interactions of the virtual resectoscope and the deformable model of the prostate. At the moment we modeled tissue deformation due to collisions by using a penalty based method where external collision reacting forces depends on the penetrating field. We are currently working in modelling tissue resection by removing and adding the triangles, produced by mesh refinement, to the mesh (and to the sphere-tree). We are also working in the integration of the LP3500 card movements commands and registering the prostate model with respect of the physical phantom. Due to difficulties in repreducing all the degrees of freedom -seven, including two additional translation degrees to the resectoscope sheath, we decided at the moment to reproduce only the five most important degrees, but we are planing to modify our device, in order to include all movements. In opinion of urology specialists, visual feedback is more important than haptic feedback and does not seem mandatory for TURP simulation, however we are also planning in the near future to include force feedback to the mechatronic device.

References

[AB96] Anwei L., Barry J.: Quality local refinement of tetrahedral meshes based on 8-subtetrahedron subdivision. *Mathematics of Computation 65*, 215 (1996), 1183–1200. 106

[ACPCSM06] Arambula Cosio F., Padilla Castañeda M., Sevilla Martinez P.: Computer assisted prostate surgery training. *International Journal of Humanoid Robotics* (2006). In press. 102

[BA02] Baerentzen J. A., Aanaes H.: Generating signed distance fields from triangle meshes. *imm-technical report-2002-21* (2002). 106

[BGTG04] Bielser D., Glardon P., Teschner M., Gross M.: A state machine for real-time cutting of tetrahedral meshes. *Graphical Models 66* (2004), 398–417. 106

[BMG99] Bielser D., Maiwald V., Gross M.: Interactive cuts through 3-dimensional soft tissue. *Eurographics'99 18*, 3 (1999). 106

[BSL*02] Brown B., Sorkin S., Latombe J., Montgomery K., Stephanides M.: Algorithmic tools for real-time microsurgery simulation. *Medical Image Analysis 6*, 4 (2002), 289–300. 104

[DJL98] Deguet A., Joukhadar A., Laugier C.: A collision model for deformable bodies. In *IEEE Int. Conf. on Intelligent robots and systems* (1998). 106

[GBTD99] Gomes M., Barret A., Timoney A., Davies B.: A computer assisted training/monitoring system for turp structure and design. *IEEE Transactions on Information Technology in Biomedicine 3*, 4 (1999), 242–250. 101

[GCMS01] Ganovelli F., Cignoni P., Montani C., Scopigno R.: Enabling cuts on multiresolution representation. *The Visual Computer 17* (2001), 274–286. 106

[GLM96] Gottschalk S., Lin M., Menocha D.: Obb-tree: A hierarchical structure for rapid interference detection. In *Proc. of ACM Siggraph'96* (1996), pp. 171–180. 104

[Hub96] Hubbard P. M.: Approximating polyhedra with spheres for time-critical collision detection. *ACM Transactions of Graphics 15*, 3 (1996), 179–210. 104

[LCN99] Lombardo J., Cani M., Neyret F.: Real-time collision detection for virtual surgery. In *Proc. of IEEE Computer Animation'09* (1999), pp. 33–39. 104

[LM97] Lin M., Menocha D.: Efficient contact determination between geometric models. *International Journal of Computational Geometry and Applications 7*, 1 (1997), 123–151. 104

[MSH*02] Manyak M., Santangelo K., Hahn J., Kaufman R., Carleton T., Hua X., Walsh R.: Virtual reality surgical simulation for lower urinary tract endoscopy and procedures. *Journal of Endourology 16*, 3 (2002), 185–190. 101, 102, 107

[OGW*01] Oppenheimer P., Gupta A., Weghorst S., Sweet R., Porter J.: The representation of blood flow in endourologic surgical simulations. *Stud Health Technol Inform 81* (2001), 365–371. 101, 107

[PCAC04] Padilla Castañeda M., Arambula Cosio F.: Deformable model of the prostate for turp surgery simulation. *Computers and Graphics 28* (2004), 767–777. 102, 106

[Qui94] Quinlan S.: Efficient distance computation between nonconvex objects. In *Proc. of the IEEE Int. Conf. on Robotics and Automation* (1994), pp. 3324–3329. 104

[SKO*04] Sweet R., Kowalewski T., Oppenheimer P., Weghorst S., Satava R.: Face content and construct validity of the university of washington virtual reality transurethral prostate resection trainer. *Journal of Urology 172*, 5 (2004), 1953–1957. 101

[TKH*04] Teschner M., Kimmerle S., Heidelberger B., Zachmann G., Raghupathi L., Führmann A.: Collision detection fo deformable objects. *Eurographics State-of-the-art Report* (2004), 119–139. 104

[VdB97] Van den Bergen G.: Efficient collision detection of complex deformable models using aabb trees. *Journal of Graphics Tools 2*, 4 (1997), 1–13. 104

[VPH] VPH: The visible human. *National Institutes of Health.* www.nlm.nih.gov/research/visible/visible-_human.html. 101

[WSM02] Wagner C., Schill M., Männer R.: Collision detection and tissue modelling in a vr-simulator for eye surgery. In *Eight Eurographics Workshop on Virtual Environments* (2002), pp. 27–36. Barcelona, Spain. 104