Ibero-American Symposium on Computer Graphics - SIACG (2006)
P. Brunet, N. Correia, and G. Baranoski (Editors)

Interactive Cutting in Voxel-Based Objects. Application to Simulate Transurethral Resection of the Prostate

R. Joan-Arinyo and J. Vilaplana

Divisió d'Informàtica Gràfica
Centre de Recerca en Enginyeria Biomèdica
Universitat Politècnica de Catalunya
Av. Diagonal 647, 8^a, E-08028 Barcelona

Abstract

Simulating cutting processes of human tissue is an important feature in virtual reality systems for medical training and rehearsal. In this paper we present a method to simulate the cutting operation in voxel-based virtual reality objects intended to simulate volume cutting in transurethral resection of the prostate. The method allows the user to directly interact with the three dimensional model in a manner that simulates dragging a resectoscope loop through the tissue, performing the task in an intuitive, natural manner.

Categories and Subject Descriptors (according to ACM CCS): I.3.4 [Graphic Utilities]: Software suport I.6.7 [Simulation Support Systems]: Environments

1. Introduction

Despite advances in endoscopic surgical training models, most urologists learn transurethral resection of the prostate on patients under consultant supervision. Training opportunities for urologists are diminishing due to reduction in the length of surgical training, appreciation of the true costs of operating room time, pressures of long waiting lists, increasing use of alternative treatments for benign prostatic hyperplasia, [AMT97, Hol98, Mun98]. However, urological trainees still must be instructed on patients before performing transurethral prostatic resection unaided but there are significant advantages in providing even limited training in the technique before the first patient exposure.

A transurethral prostatic resection simulator is a possible solution. It would allow trainees to become familiar with the resectoscope and the approach to the limits of the resection process, and could help them surmount the learning curve associated with the technique in the absence of time constraints and without risk to patients.

Cutting is a central manipulation in computer-aided surgery simulators. The literature contains numerous methods for cutting both surface and volumetric models. In general, these methods consider mesh-cutting techniques where models are represented by meshes of geometric volume elements that must be re-meshed after primitive removal. See [BSM*02] for a survey.

Most of the reported works neglect to explain precisely how to generate and deal with the detached chips from the material that must remain and there is a paucity of papers dealing with this issue, which is paramount in transurethral resection of the prostate. The work by Wong *et al.* is a notable proposal in that respect, [WSHS98]. The main drawback of this technique is that the user defines a 2D closed contour on the volume surface from which the shape of the cutting surface is estimated through an expensive minimization of a cost function.

Padilla *et al.*, [PA04], offer an algorithm to simulate tissue removing in transurethral resection of the prostate. The algorithm assumes that the removed tissue takes the shape of a spherical region with a given radius which clearly limits its applicability. Moreover, the method does not allow to generate detached chips of material.

In this context, the motivation of this work was to design a cutting technique to shear off chips of material in transurethral resection of the prostate where the prostate gland is modelled as a regular 3D array of data, with each element representing a sampled point (measured o calculated) in the volume. The goal is achieved by designing an algo-

rithm that replicates the gestures and actions performed by the urologist in a real scenario.

In this paper we examine first the main concepts concerning transurethral prostatic surgery. Then we recall some basic concepts on 3D discrete spaces that we will use later on. Next, we consider how objects involved in the process are modelled. Here we consider the prostate gland and the resectoscope. After presenting the basic resection algorithm, we develop an improved version. Finally, we offer a brief conclusion.

2. Transurethral Prostatic Surgery

The prostate is a walnut-sized gland located in front of the rectum, at the outlet of the bladder. It contains gland cells that produce the seminal fluid, which protects and nourishes sperm cells in semen. The urethra passes through the prostate longitudinally from the bladder to the *verumontanum*. Distal to the *verumontanum* is the sphincter, which is responsible of the urinary continence. The prostatic capsule packs these anatomical structures.

The prostate tends to grow as men get older. In some cases, it becomes large enough to put pressure on the urethra, causing problems with urination, such as incomplete emptying of the bladder or dribbling of urine. The condition is known as benign prostatic hyperplasia, (BPH).

Transurethral resection of the prostate (TURP) is a minimally invasive surgical procedure to deal with patients with BPH which involves the cutting and coagulation of prostatic tissue with a resectoscope, [Mic92].

The TURP operation is performed via a resectoscope introduced through the urethra after appropriate anaesthesia, Figure 1, [Reu82]. No cut is made on the skin and no scars result. Basically, a resectoscope includes a cutting tool and an optical system packed in a sheath. Although the cutting tool comes in a variety of designs, one of the most widely used is the electrical cutting loop, [Mic92]. In the basic technique of tissue resection, the cutting loop is extended beyond the end of the sheath, tissue is engaged and the cutting current is applied. The loop is then drawn through the tissue and retracted into the sheath. As the loop enters the sheath, the tissue is sheared off into chips which are propelled away from the field of vision by an inflowing irrigation solution. The chips collect in the bladder and are removed at the end of the operation via the resectoscope.

As with all surgery, there are potential complications like perforating the prostate capsule, and side-effects like persistent urinary incontinence as a result of damaging the sphincter. Therefore, before actually performing on a patient, urologists must develop accurate spatial orientation inside the prostate to avoid resections in unacceptable regions near the prostatic capsule, the bladder neck or the *verumontanum*.

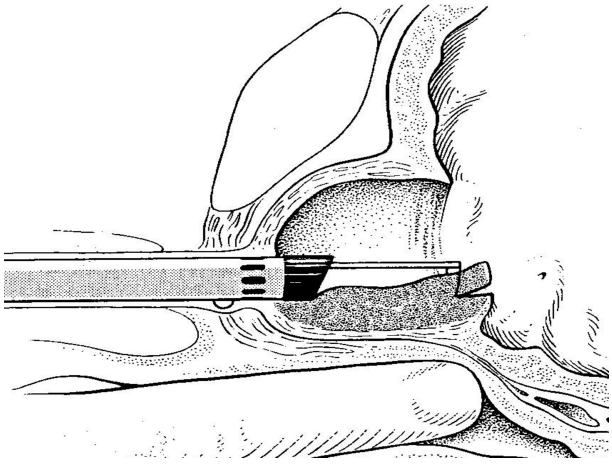

Figure 1: *Prostatic tissue resection with a resectoscope.*

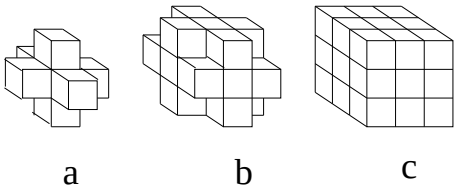

Figure 2: *The sets of 3D voxels that are N-adjacent to the voxel at the center. a) N = 6. b) N = 18. c) N = 26.*

3. Separating Two Sets of Voxels

The concepts in this section can be found in [HYFK98] and [KR89]. We include them here for the sake of completness.

A 3D discrete space Z^3 is a set of integer grid points in a 3D Euclidean space denoted Σ. A 3D grid point is a zero dimensional object defined by its Cartesian coordinate (x,y,z). The neigbourhood of a grid point p is the set of all points in the Euclidean space that are closer to p than to any other grid point. It is a unit cube around p, known as a *voxel*.

Two voxels are 26-*adjacent* if they share a vertex or an edge or a face. Every voxel v has 26 such neighbours. Eight of them share a vertex with v, twelve of them share an edge with v, and six of them share a face with v. Facesharing voxels are defined as 6-*connected*, and the voxels that are both edge-sharing and face-sharing are defined as 18-*adjacent*. Figure 2 illustrates these definitions.

Let $N \in \{6,18,26\}$ denote the adjacency relation in 3D. An *N-path* is a sequence of voxels such that consecutive pairs are N-adjacent. Two voxels, v_1, v_2, are said to be *N-connected* in Σ if there exists a connecting N-path whose bounds are voxels v_1 and v_2. A *closed N-curve* is an N-path P that either contains a single voxel or each voxel in P has exactly two N-adjacent voxels also in P. An *open N-curve* is an N-curve with two voxels each of which has only one

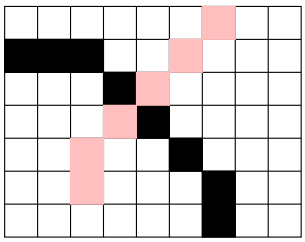

Figure 3: *An 8-connected curve penetrating through another 8-connected curve.*

N-adjacent voxel in P. These voxels are the *endpoints*. The same concepts are defined in a 2D space where N takes values in $\{4,8\}$.

In continuous space, it is impossible to pass from the region enclosed by a curve to the region outside the curve without crossing the curve itself, [Mun75]. In discrete space, however, the opposite is possible. Figure 3 shows the situation in 2D where one 8-connected curve penetrates another 8-connected curve without meeting it. Note, however, that an 8-connected curve and an 4-connected curve cannot penetrate each other without having at least one voxel in common. It is said that they are *opposite*. In 3D a 6-connected curve is opposit to 18- and 26-connected curves. Note also that two 4-connected curves in 2D and two 6-connected curves in 3D cannot penetrate each other without meeting at some voxel.

Let S be a C^0 continuous surface in R^3, and let $\widehat{S}$ be a set of voxels that is a discrete representation of S generated by some arbitrary computation. Since S is C^0 continuous, it is not penetrable. Therefore we require $\widehat{S}$ to be not penetrable. This requirement is called *separability*, defined as follows.

Let A, B and C be three different disjoint sets of voxels. A is said to *N-separate* B and C if any N-path, P, between a voxel in B and a voxel in C meets A.

Let $S \in R^3$ be a C^0 continuous surface such that $R^3 - S$ has exactly two connected components, I and O. Let $\widehat{S}$ be a voxelization of S. If $\Sigma - \widehat{S}$ has exactly two N-connected components $\widehat{I}$ and $\widehat{O}$, then $\widehat{S}$ is said to be *N-separating*. In 2D, an 8-connected curve is 4-separating while an 4-connected curve is 8-separating. In 3D, an 6-connected surface is both 18- and 26-separating.

It is worth to note that, once the separating set $\widehat{S}$ is computed, the definitions above do not provide any tool to actually identify which connected component is $\widehat{I}$ and which one is $\widehat{O}$.

4. Geometric Modelling

For our purposes, geometric models for two objects must be provided: the prostate gland and the resectoscope.

4.1. The Prostate Gland Model

The prostate gland is modelled using the volume graphics approach, [KCY03], where objects are modelled as a regular or irregular 3D array of data, with each element representing a sampled point (measured o calculated) in the volume. For TURP simulation, this representation has a number of advantages over the use of polygons or solid geometric primitives. First, because the data organization is the same as the acquired data, a voxel-based representation is natural for the 3D digital images produced by medical scanning technologies such as Magnetic Resonance or Computed Tomography. Second, since no surface extraction or other data reformatting is required, errors introduced by fitting surfaces or geometric primitives to the scanned images can be avoided. Finally, volumetric objects can incorporate detailed information about the internal anatomical or physiological structure of organs and tissues. This information is particularly important for realistic modeling and visualization of complex tissue volumes.

We created a 3D volume model of the prostate gland based on discretizing by hand a plastic made physical model of natural size. The prostate gland is represented using a generalized voxel model, in a 3D rectilinear grid with an spatial resolution of 0.1 mm. The Z axis along the urethra has 194 voxels while X and Y axis on a plane normal to the Z axis, have 256 and 202 voxels, respectively. Each voxel is associated with a set of attributes such as its membership to anatomical regions or color.

In the model we define different sets of voxels corresponding to the anatomical structures: The prostatic capsule, the internal tissue, and the urethra, also called *channel* in this work. We assume that the set of voxels corresponding to the capsule and to the channel are 6-connected (26-separating), while the internal tissue, in general, can be 26-connected.

For computational purposes we distinguish in the capsule the voxel with minimum grid coordinates called *seed*.

4.2. Resectoscope Model

The resectoscope is modelled using the boundary representation approach, [FvDFH96]. The sheath is modelled as a hollow cylinder, and the cutting tool as one half torus attached to two cylinders (Figure 5a). With the cutting loop we associate a set of uniformly distributed mark points, (Figure 5b), that will be used to trace the path swept by the tool while defining the cut. The optical system is modelled as a light spot placed on the sheath bound.

5. The Basic Algorithm

Since our aim is to provide a tool with which the user can interact with a high degree of realism, the algorithm has been designed to cause movements of the resectoscope to control the image of the cutting loop, recognize collisions between

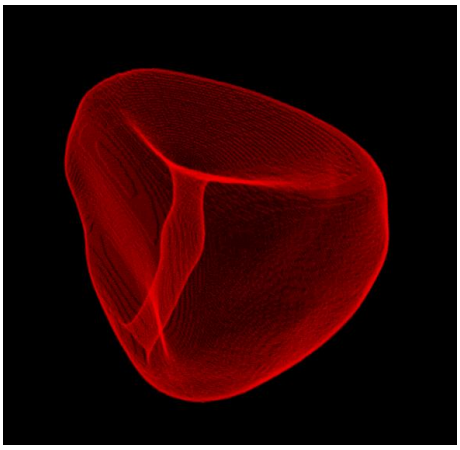

Figure 4: *Transparent view of a prostate model.*

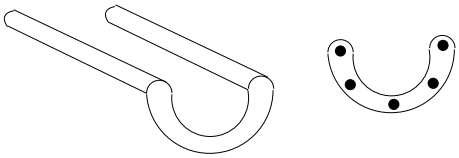

Figure 5: *Model of the cutting tool. a) The loop. b) Set of associated mark points.*

the loop and the prostate, and cause elimination of tissue when the cutting loop occupies the same virtual space as the prostate, creating sheared off chips and the corresponding furrow in the prostate lumen.

According to what is considered a good TURP practice, [Mic92], our algorithm is designed under the assumption that during the resection process, prostatic material is removed through interaction with the channel while the cutting tool is not allowed to penetrate the prostate capsule or any other anatomical structures that could lead to potential complications or side-effects.

We assume that the prostate model is available as global information. Moreover, there is a global boolean, from now on denoted `flipflop`, with an initially assigned given value, let us say, true. Similarly, we assume that all the voxels in the prostate model have a boolean field, say `flipflop`, which initially have been assigned the same value as the global `flipflop`. If the cutting tool is placed at the initial point in the channel where the cut must be performed, and `seed` is the distinguished voxel in the prostate model, the algorithm is outlined in Figure 6. Let's see each step in more detail.

First of all, the algorithm computes the *cut*, that is, the set of voxels in the prostate tissue that separates those voxels that will be sheared off as chips from the prostate tissue. The cut is computed in three stages. In a first stage, as the user moves the cutting tool, the algorithm captures from the user interface the cut mesh, that is, a set of discrete locations swept out in the prostate tissue by the loop mark points (Figure 7a). Then the cut triangulation is computed by triangulating the set of vertices in the cut mesh (Figure 7b). Finally, the cut is computed as a 26-separating voxelization of the cut triangulation, [HYFK98].

```
procedure BasicResection ()
    ComputeTheCut (cut)
    flipflop := not flipflop
    FlipMarks (seed)
    ConnectAndMoveChips (cut)
endprocedure
```

Figure 6: *Basic resection algorithm.*

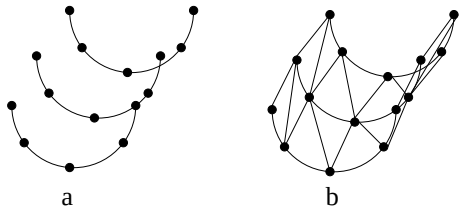

Figure 7: *Computing the cut. a) Cut mesh. b) Cut triangulation.*

The cut separates the set of voxels in the model into two 26-connected components. The resulting topology of the cross section of the prostate is sketched in Figure 8. To actually detach the chips, we need to distinguish the component that must remain from the component that will be sheared off. This is achieved visiting each voxel in one component and flipping its `flipflop` field.

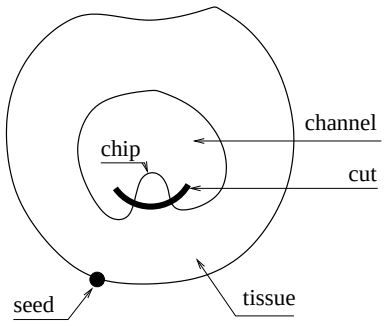

Figure 8: *Topology of the cross section of the prostate.*

```
procedure FlipMarks (v)
    for vn in the 26-neighbourhood of v do
        if not IsCut (vn) and vn.flipflop ≠ flipflop then
            vn.flipflop := flipflop
            FlipMarks (vn)
        endif
    endfor
endprocedure
```

Figure 9: *Flipping voxels marks.*

To visit the voxels in a connected component all what we need to know is one voxel known to be contained within the component, [FvDFH96]. This is the *seed*. Note that, in our scenario, to visit the voxels in the component corresponding to the chip arises some difficulties because selecting a seed would require a dialog with the user. However, visiting the voxels in the component that must remain is easy because any voxel known not to be in the chip is a valid seed. In particular, the capsule voxel in the prostate model with minimum grid coordinates fulfils this requirement. In these conditions, the algorithm to label the component that will remain is the one given in Figure 9.

In general, one cut can yield several chips each built as a disjoint 26-connected set of voxels. Conceptually, a chip is separated from the prostate tissue by the path swept out by the cutting loop. Therefore we include in the sheared chip the voxels in the cut set. Therefore, seed voxels from which chips are built are taken from the cut set.

Assume that the function `IsOldChannel(v)` returns true if and only if the `flipflop` field of voxel `v` is different from the global `flipflop` value. Then the algorithm that builds the chips is given in Figure 10.

Finally the tissue chips generated in the cutting process are moved toward the bladder where they are collected. This algorithm is just routine matter.

6. Improving the Efficiency of the Basic Algorithm

The main drawback in the algorithm presented in Section 5 is that, for each cut operation, it has to flip the mark of $194 \times 256 \times 202$ voxels what yields an algorithm useless for real-time simulation.

To improve the efficiency of the algorithm, we first define in the prostate model the following sets of voxels (see Figure 11): The prostatic capsule, the internal tissue, and the *channel*. In the prostate capsule we distinguish two sets, called *rings*, that separate the capsule voxels from those of the channel. Besides, we also distinguish in one of the ring sets a specific voxel which will be the *seed*. The capsule, the channel and the rings are 26-separating sets.

The general process of material removal is modelled as follows. At the beginning, the prostate is full of tissue, as

```
procedure ConnectAndMoveChips (cut)
    while not EmptyCut (cut) do
        chip := ∅
        ExtractVoxel (cut, v)
        ConnectChip (cut, v, chip)
        MoveChip (chip)
    endwhile
endprocedure

procedure ConnectChip (cut, v, chip)
    TransferVoxelFromProstateToChip (v, chip)
    for vn in the 26-neighbourhood of v do
        if IsCut (vn) or
        (IsTissue (vn) and IsOldChannel (v) then
            ConnectChip (cut, vn, chip)
        endif
    endfor
endprocedure
```

Figure 10: *Connecting and moving tissue chips.*

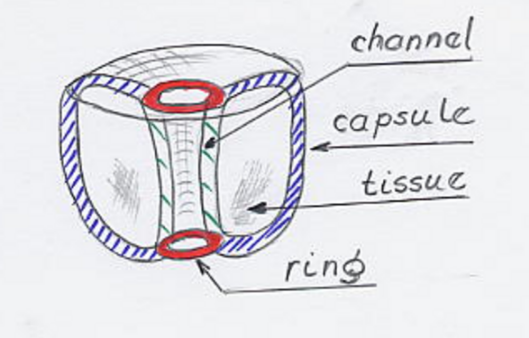

Figure 11: *Sets of voxels.*

depicted in Figure 11. Then tissue is removed by the user interacting through the channel, (Figure 12 top shows the prostate with tissue partially removed) until the prostate is emptied, Figure 12 bottom.

As in Section 5, we assume that the prostate model is available as global information and that there is a global boolean, `flipflop`, with an initially assigned given value, let us say, true. Similarly, we assume that at the beginning, channel voxels have their `flipflop` field assigned the same value as the global `flipflop`. If the cutting tool is placed at the initial point in the channel where the cut must be performed, and `seed` is the special voxel in one of the model's ring set, the algorithm that performs the cut is outlined in Figure 13.

The *cut* is computed as in Section 5. Once the cut is figured out, the algorithm updates the channel set. Two things need to be done. First the set of voxels that will result exposed to fresh air after removing the tissue chips

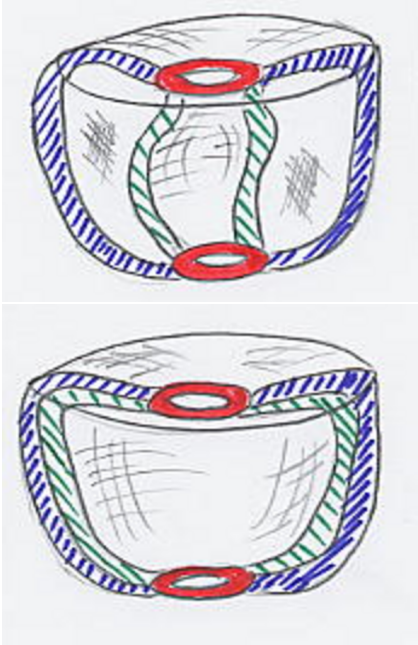

Figure 12: *Material removal process.*

```
procedure Resection ()
    ComputeTheCut (cut)
    flipflop := not flipflop
    UpdateChannel (seed, cut)
    ConnectAndMoveChips (cut)
endprocedure
```

Figure 13: *Efficient resection algorithm.*

is computed. Then the flipflop field of channel voxels is flipped. Figure 14 illustrates the channel updating process. We need to guarantee that the updated channel is a 26-separating set of voxels. This is achieved through the function `CutIn26Neigbourhood(v)` which returns true if and only if there is a voxel labeled as cut in the 26-neighbourhood of voxel `v`. In these conditions, the algorithm to update the channel is given in Figure 15.

The algorithm to connect chips is similar to the one given in Section 5. Assume that the function `IsOldChannel(v)` returns true if and only if the flipflop field of voxel `v` is different from the global `flipflop` value. Then the algorithm is given in Figure 16. Note that the 26-separating updated channel precludes the chip from growing inside the prostate. See Figure 14 bottom.

7. Results

The hardware platform used in the development consists of a Pentium III personal computer running at 600 MHz,

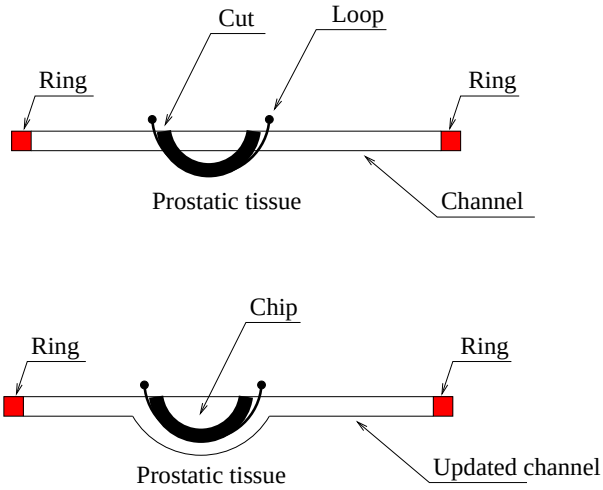

Figure 14: *Updating the channel.*

```
procedure UpdateChannel (seed, cut)
    for vn in the 26-neighbourhood of seed do
        if NotVisitedYet (vn) then
            MarkVoxelAsVisited (vn)
            if IsRing (vn) then
                UpdateChannel (vn, cut)
            else if IsChannel (vn) then
                vn.flipflop := flipflop
                UpdateChannel (vn, cut)
            else if IsTissue (vn) and
                    CutIn26Neigbourhood (vn) then
                vn.flipflop := flipflop
                LabelVoxelAsChannel (vn)
                UpdateChannel (vn, cut)
            endif
        endif
    endfor
endprocedure
```

Figure 15: *Updating the channel.*

with 256 MB random access memory and a Nvidia Geforce 4400 with 256 MB. The operating systems is GNU Debian LINUX, the programming languages used are **C** and **C++**. The Graphic User Interface for rendering the transurethral resectoscope navigation is implemented in `OpenGL/X` from Nvidia.

In the simulator so far developed, the user interacts with the system through the keyboard what results in a rather cumbersome interaction. However, in the current setting we get a minimum display rate of 18 frames per second which is clearly higher than the minimum required for real-time visual interactivity.

Figure 17 shows computer generated transurethral views

```
procedure ConnectAndMoveChips (cut)
    while not EmptyCut (cut) do
        chip := ∅
        ExtractVoxel (cut, v)
        ConnetChip (cut, v, chip)
        MoveChip (chip)
    endwhile
endprocedure

procedure ConnectChip (cut, v, chip)
    TransferVoxelFromProstateToChip (v, chip)
    for vn in the 26-neighbourhood of v do
        if IsCut (vn) or IsTissue (vn)
                or IsOldChannel (vn) then
            ConnectChip (cut, vn, chip)
        endif
    endfor
endprocedure
```

Figure 16: *Connecting and moving tissue chips.*

including fully textured prostatic lumen, and resectoscope loop. The picture on the top shows the loop sweeping out the space occupied by the prostatic tissue to define the cut. The picture in the middle shows the loop after completing the cut and a detached chip which is moving toward the bladder and that partially occludes the furrow created. The picture on the bottom shows the prostatic lumen after completing several cuts.

8. Conclusion

We have reported on an algorithm to simulate volume cutting in virtual transurethral resection of prostatic tissue. The software was developed on a standard personal computer which allowed images of the lumen of the prostatic urethra and resectoscope loop to be generated and interacted with in real time.

Although the algorithm has been designed to replicate the gestures urologists perform when practicing with human bodies, this is not fully achieved because so far users interact with the system through the keyboard. Currently we are working to replace the keyboard with a dummy resectoscope.

Acknowledgments

This research has been partially funded by the Institut Català d'Urologia i Nefrologia S.L., by the Centre de Referència en Enginyeria Biomèdica de Catalunya, by the Red de Grupos del Instituto Carlos III, IM^3: Imagen Molecular y MultiModalidad, and by Ministerio de Educación y Ciencia and by FEDER under grant TIN2004-06326-C03-01. Thanks to Joel García for his efforts in implementing the software.

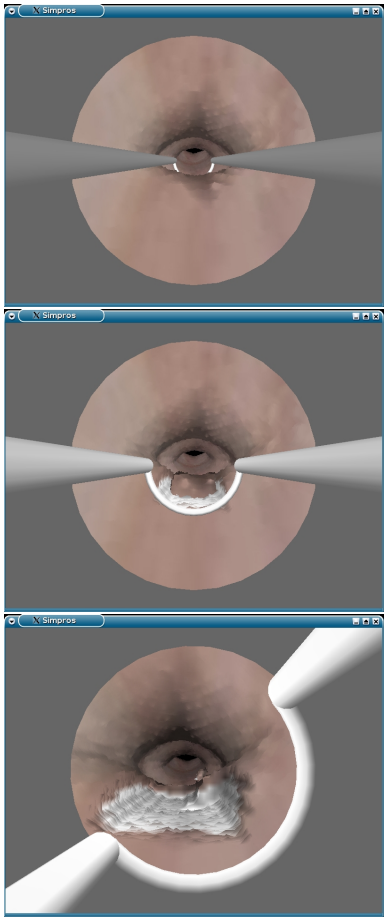

Figure 17: *Computer generated transurethral views.*

References

[AMT97] AHMED M., MEECH J., TOMETEY A.: Virtual reality in medicine. *British Journal of Urology 80* (1997), 46.

[BSM*02] BRUYNS C., SENGER S., MENON A., MONTGOMERY K., WILDERMUTH S., BOYLE R.: A survey of interactive mesh-cutting techniques and a new method for implementing generalized interactive mesh cutting using virtual tools. *The Journal of Visualization and Computer Animation* (2002), 21–42.

[FvDFH96] FOLEY J., VAN DAM A., FEINER S., HUGHES J.: *Computer Graphics. Prinviples and Practice*, 2nd ed. Addison-Wesley Pu. Co., Reading, MA, 1996.

[Hol98] HOLTGREWE H.: The economics of urological practice in the twenty first century. *Urology Clinics 25* (1998), 1.

[HYFK98] HUANG J., YAGEL R., FILIPPOV V., KURZION Y.: An accurate method for voxelizing

polygon meshes. In *Proceedings of the 1998 IEEE Symposium on Volume Visualization* (Research Triangle Park (NC) US, 1998), ACM Press, pp. 119–126.

[KCY03] KAUFMAN A., COHEN D., YAGEL R.: Volume graphics. *IEEE Computer 26*, 7 (July 2003), 51–64.

[KR89] KONG T., ROSENFELD A.: Digital topology: Introduction and survey. *Computer Vision, Graphics, and Image Procesing 48*, 3 (December 1989), 357–393.

[Mic92] MICHAELS E.: Cystourethroscopy and transurethral resection of the prostate and bladder. In *Master of Surgery. Urologic Surgery*, Jr. J. F., (Ed.). Little, Brown and Company, Boston, 1992, ch. 80, pp. 641–655.

[Mun75] MUNKRES J. R.: *Topology: a first course*. Prentice-Hall Inc., 1975.

[Mun98] MUNDY A.: The future of British urology. *British Journal of Urology 82* (1998), 476.

[PA04] PADILLA M., ARÁMBULA F.: Deformable model of the prostate for TURP surgery simulation. *Computer & Graphics 28* (2004), 767–777.

[Reu82] REUTER H. J.: *Atlas of urologic endoscopic surgery*. Georg Thieme Verlag, Stuttgart, 1982.

[WSHS98] WONG K., SIU Y.-H. S., HENG P.-A., SUN H.: Interactive volume cutting. In *Graphics Interface* (1998), pp. 99–106.