Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XXXI, págs. 3061-3074 (artículo completo)
Alberto Cardona, Paul H. Kohan, Ricardo D. Quinteros, Mario A. Storti (Eds.)
Salta, Argentina, 13-16 Noviembre 2012

MESH GENERATION OF PELVIC ORGAN SURFACES USING ACTIVE CONTOURS WITH SELF-COLLISION DETECTION

R. Namías[a], J. P. D'Amato[b,c], M. del Fresno[b,d] and M. Vénere[b,e]

[a]*CIFASIS, French Argentine International Center for Information and Systems Sciences, UAM (France) / UNR-CONICET (Rosario), Bv. 27 de febrero 210 bis, (S2000EZP), Rosario, Santa Fe, Argentina, secretaria@cifasis-conicet.gov.ar, http://www.cifasis-conicet.gov.ar/*

[b]*Instituto PLADEMA, Universidad Nacional del Centro, Campus Universitario Paraje Arroyo Seco, (B7000), Tandil, Argentina pladema@exa.unicen.edu.ar, http://www.pladema.net/*

[c]*Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Argentina*

[d]*Comisión de Investigaciones Científicas de la Prov. de Buenos Aires (CIC-PBA), Argentina*

[e]*Comisión Nacional de Energía Atómica, Argentina*

Keywords: 3D segmentation, mesh generation, collision detection, visualization.

Abstract. Segmentation of medical imaging is one of the main processes in the development of computer-aided applications in several medical practices. In the last few years, different methods were proposed for aiding the diagnosis, treatments and even pre-surgical assistance.
In particular, we are interested in the physical simulation of pelvic dynamics to study the pelvic organ prolapse disease. Hence, our goal is to generate a geometrical model of pelvic organs from Magnetic Resonance Imaging (MRI) acquisitions, which afterwards will contribute to build a physical model of the patient organs.
In this work, we present a tri-dimensional geometrical generation method for general volumetric segmentation purposes. The geometrical models should be as accurate as possible, therefore the surface mesh obtained must keep some important quality aspects such as not presenting self intersections between its triangles. For this purpose, we introduce a modified active surface model with a novel self-collision inhibitor module.
Keywords: Mesh Generation, Self-collision Detector, Active Surfaces, Magnetic Resonance.

1 INTRODUCTION

In modern surgery practices, many diseases, such as pelvic organ prolapse or tumour extraction, are largely studied with non-invasive imaging techniques. Surgerous are prepared on visualization systems and sometimes complex situations are simulated. These kind of systems, uses digital representation of specific patients organ and run behavioral simulations, in order to support the surgeon decisions. These simulations required a very good aproximated representation of the real organ to bring realistic results.

To generate these representations, dynamic and volumetric MRI are acquired from patients and then the main organs are segmented [Xiping and Jie (2002); Singh et al. (1998)]. Afterwards, geometrical models are extracted from the segmentation results [Bay et al. (2011)]. Therefore, the segmentation step is critical to obtain more accurate geometrical models. Techniques such as thresholding and region growing are commonly used for this purpose [Lin et al. (2001); Pohle and Toennies (2001)]. Other advanced such as approaches level sets are also widely used [Ma et al. (2010); Mcinerney and Terzopoulos (1999); Osher and Fedkiw (2000)].

Active contours models (ACM) [Kass et al. (1988)] have gained popularity as it shows to be a dynamic and efficient method for multiple purposes. We have successfully applied this method for brain segmentation [del Fresno et al. (2009)], but the method have to be improved in order to detect blurry organs.

1.1 Active contours for mesh generation

Generally, the ACM starts from a parametric surface such as spheres or ellipsoids [Cohen (1991)] generally defined by an user and evolves applying forces on the nodes of the surface until it reaches some criterion. There are different approaches about initialization and evolving rules.

Some works such as [del Fresno et al. (2009)], propose an hybrid approach that starts the segmentation with a Region Growing method and then apply an active contour technique to obtained a more refined result. [Pimenta et al. (2006)] propose a 3D model reconstruction of the bladder applying 2D active contour segmentation in each slide of the MR acquisition. This approach not only needs post-processing to create a 3D model from a set of slide contours but also do not take into consideration the extra information of space and continuity in 3D which could be greatly useful in the segmentation of soft organs that are not easily distinguishable in the acquisitions.

During mesh evolution artifacts, like loops, may appear. In Figure 1 right panel we show one loop in a mesh slide proyection. These deformations not only give wrong segmentation results, but also lead to numerical instability in the force calculation that evolve the model. As the result of these perturbations we can obtain weird anomalies as we can depict in Figure 1 left panel. Bischoff et al. [Bischoff and Kobbelt (2003)] propose a geometrical control by restricting the movement of the contour vertices to grid-lines of a uniform lattice. This model called restricted snakes (r-snakes) controls efficiently the self collision problems. As the points (snakels) can only move throught the grid-lines, when two of them are in the same supporting segment of the grid and are moving towards each other they are not allowed to cross, they can just touch one to the each. Although this technique deals efficiently with the collisions problem, it restricts the points movements and more important, it is not extended to 3D models.

We present our approach that deals with geometrical artifacts during surface evolution. The

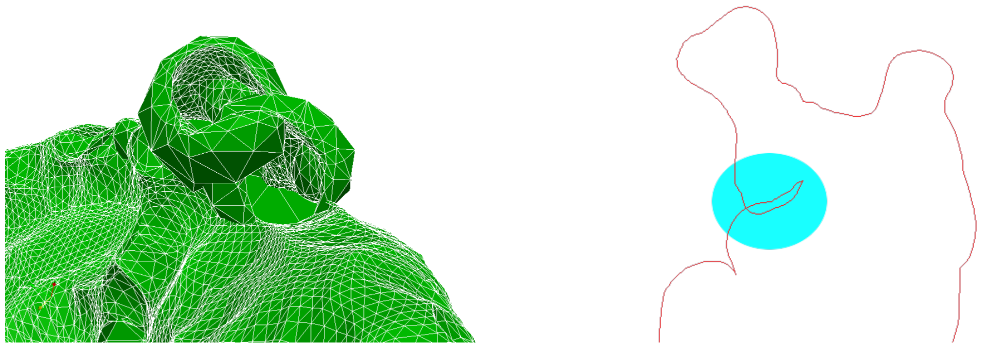

Figure 1: Loops artifacts. [left] Anomaly due to loop evolution and [right] loop in a mesh slide proyection.

method starts from a initial surface obtained with a region growing method and then evolves moving accordingly while geometry control is applied. If some surface elements are colliding, we propose rollback to a safety state (with no collisions), "freeze" conflict nodes and then continue the evolution. Collision detection requires too much computing and many times no collisions are detected, so a suitable collision detection frequence should be defined. This method was applied to segmentate the main female pelvic organs.

This work is organized as follows: in section 2 we describe the geometrical model generation including the segmentation procedure; in section 3 we present the self-collision detection scheme; then in section 4 the modified active surface model with the self-collision inhibitor is exhibited; in section 5 we show a syntetic and a real case experiment to demonstrate the correctness and benefits of the proposed model; finally, in section 6 conclusions and future work are mentioned.

2 TRIDIMENSIONAL GEOMETRICAL MODEL GENERATION

The main input of the segmentation process is a DICOM (**D**igital **I**maging and **Co**mmunication in **M**edicine:international standard in medical imaging) graylevel 3D MRI. This volume is processed by a tree stage procedure. First, a region growing (RG) technique is applied to obtain a initialization mesh.Second, the RG output is converted to a triangular 3D surface mesh. As the resulted mesh has several "staircase" artifacts, it is smoothed before passing to the active surface algorithm that finishes the segmentation task.

2.1 Region Growing Step

The region growing algorithm is quite standard. The expert manually set one or more initial points (seeds) in the organ of interest. The algorithm keeps a queue with the visited voxels. In the beginning, only the seed indexes are included. For each candidate voxel in the queue we evaluate whether it satisfies the acceptance criterion. If so, all its neighbors, that were not visited before, are placed in the queue and the candidate is added to the region. If not, the voxel is labeled as a border voxel. The algorithm finishes when the candidates queue is empty. In this work, a 26-neighborhood for each voxel is considered. The acceptance criterion is the key of the algorithm.

2.1.1 Acceptance criterion

Let us consider the set $S(r)$ of all voxels r-neighboring any seed defined in a given region (26-connectivity). The characteristic intensity $CI(r)$ and standard deviation $\sigma(r)$ of the region are calculated as:

- Caracteristic Intensity: $CI(r) = \frac{1}{N(r)} \sum_{v \in S(r)} I(v)$
- Caracteristic Deviation: $\sigma(r) = \sqrt{\frac{1}{N(r)} \sum_{v \in S(r)} [I(v) - CI(r)]^2}$

where $I(v)$ is the image graylevel intensity and $N(r)$ is the number of voxels contained in $S(r)$.

So we define the first criterion as:

$$s_v^1(k,r) = \frac{|I(v) - CI(r)|}{k\sigma(r)} \leq 1 \tag{1}$$

where $k \in (0, \infty)$ is a free parameter which weights the acceptance tolerance of the criterion. This criterion does not perform well in presence of noise. Noisy voxels are left outside the region when they should be inside. Therefore, in order to avoid this problem a new criterion is used:

$$s_v^2(k,r) = \frac{\sum_{v'} H(1 - s_{v'}^1(k,r))}{N(v)} \geq f \tag{2}$$

where H is a heavyside function applied to each neighbor v' of v and f is a parameter between 0 and 1. H function returns 1 if v' is accepted according to the $s_v^1(k,r)$ criterion, or 0 otherwise. Then, one voxel is accepted only if a fraction of neighbors over the value f are accepted by the first criterion. This last criterion, called the the fraction-neighborhood criterion, is more robust in presence of noise than the first one.

2.2 Surface mesh generation

In order to create a mesh suface of the object of interest, we take the voxels of the border after doing the region growing process. For each voxel in the frontier, we select the face between the interior and exterior of the ROI. For each face, we draw two triangles in an appropriate way; all their four vertex are added clockwise so the triangles normals point to the exterior of the ROI.

The original mesh shape is not very appropriate as input to the active surface technique. Its voxelized shape is not suitable for being evolved by an approximate differential equation towards its final position. Therefore, the whole mesh is smoothed by a geometrical filter described in [Taubin (1995)].

2.3 Active Surface Models

The final stage of the procedure is the active surface model usually called snake. Different techniques were proposed for the evolution of the snake, such as finite differences, finite elements or dynamic programming. A model based on the formulation proposed by Mcinerney et al. [Mcinerney and Terzopoulos (1999)] is applied. In our case, we consider that each **ac-**

tive vertex s_i (not every vertex) of the snake mesh evolves according to the following motion equation:

$$\gamma_i = \frac{ds_i}{dt} - a\alpha_i(t) + b\beta_i(t) = q\rho_i(t) + pf_i(t) \tag{3}$$

where $\alpha_i(t)$, $\beta_i(t)$, $\rho_i(t)$ and $f_i(t)$ are the tension, flexion, inflation and external forces respectively, and γ_i is a damping coefficient. The internal energy simulates the characteristics of an elastic membrane. The internal tension and flexion acting on the vertex s_i represent the snake resistance to stretching and bending respectively, and are calculated as:

$$\alpha_i = \frac{1}{m} \sum_{j \in N(i)} s_j(t) - s_i(t) \tag{4}$$

$$\beta_i = \frac{1}{m} \sum_{j \in N(i)} \alpha_j(t) - \alpha_i(t) \tag{5}$$

where $N(i)$ is the set of nodes s_j neighboring the node s_i and m is the number of these neighbors. The respective derivatives correspond to the Laplacian and the squared Laplacian and they are approximated using the umbrella operator by considering the local mesh topology at the node s_i.

The inflation ρ_i and the external force f_i are calculated as:

$$\rho_i(t) = F(I(s_i(t)))n_i(t) \tag{6}$$

where n_i is the unitary vector normal to the surface at node s_i and F is a binary function relating ρ_i to the intensity field I:

$$F(I(s_i(t))) = \begin{cases} +1 & if\ \frac{|I(s_i) - CI(r)|}{k\sigma(r)} \leq 1 \\ -1, & otherwise \end{cases} \tag{7}$$

The local external force which contains the expansion of the snake at significant edges, acts in each node emulating a potential gradient:

$$f_i(t) = G[\varphi(s_i)] \tag{8}$$

where the G is the gradient vector and the potential φ is defined as:

$$\varphi(s_i) = -grad[FI(s_i)] \tag{9}$$

The scalar gradient is $grad[.]$ and $FI(s_i)$ is the intensity $I(s_i)$ smoothed with a Gaussian Filter. Since the initial guess provided by the growing process is a close approximation of the final model, Eq. (3) can be solved directly by applying an explicit first-order Euler scheme:

$$s_i^{(t+\Delta t)} = s_i^{(t)} - \frac{\Delta t}{\gamma_i}(-a\alpha^{(t)} + b\beta_i^{(t)} - q\rho_i^{(t)} - pf_i^{(t)}) \tag{10}$$

provided that the time steps are sufficiently small.
The iteration proceeds until the following stopping criterion is achieved.

2.3.1 Stopping Criterion

We use a two parameter stopping criterion. We measure the fraction of points f that moves less than a maximum tolerance distance t_{max}. The parameter t_{max} (tolerance) is the maximum displacement tolerance value of a point and f_min is the minimal fraction of points that must be under the tolerance t_{max} to achive the criterion. So, when the fraction of points f that are displaced at most t_{max} is over f_{min} the criterion is achieved.

The advantage of using the RG result to initialize the snake model is that the deformation is then limited to minor perturbations requiring fewer iteration steps. The deformable mesh is composed of subvoxel triangular elements, so the final segmentation has subvoxel accuracy and yields a smooth surface representation, matching concavities and convexities that may be present on complex geometries.

3 SELF-COLLISION DETECTION SCHEME

The self-collision detection scheme aims to notice when two triangles of the mesh, which are not neighbors, collide or self-intersect. We say that two triangles are not neighbors if they do not share any vertex. The objective is to get a list of triangles that are colliding. As comparing each triangle with each other in the mesh has a quadratic complexity (not suitable for this problem), triangles should be classified and complexity reduced.

3.1 CLASSIFICATION

The first step of the detection process is a classification stage using Uniform Grids. We discretize the space of the image that contains the mesh by dividing the volume into a grid where each cell of this grid is a cube inside the analyzed volume. A visual representation can be appreciated in Figure 2.

The number of bins in each axis is determinated empirically and is calculated automatically as the range of the axis divided by the edge mean length (μ) of the triangles:

$$Bins[i] = \frac{Max[i] - Min[i]}{\mu} \qquad i = 1, 2, 3 \tag{11}$$

We use a tridimensional matrix data structure to reprecent the grid. Each cell of the matrix is a list of triangles' identification (*ids*), initially empty. So, we iterate for each triangle in the mesh and compute its bounding box to calculate its intersection with the cells of the grid. If a triangle's bounding box intersects a cell, its *id* is added to the cell's list of triangles. Finally, for each cell we have a list of triangles that intersects it. After visiting all the triangles of the mesh, the cell's lists are complete and the classification step is finished.

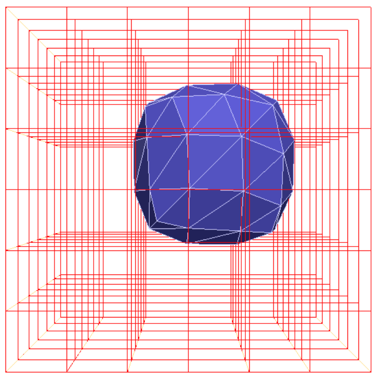
Figure 2: Volume discretization.

3.2 COLLISION DETECTION

The detection step must decide whether or not two triangles of the same cell of the grid collide. A pre-processing stage is done to eliminate all the lists that contain less than three triangles *ids* in it. The heuristic considers that due to the mesh is a closed surface if a vertex or an edge is inside the cell, then its neighbor will also intersect this cell, therefore we need at least three triangles to have a real collision. With the remaning lists an all-to-all self-intersection proof is done. This proof consists in three checkings. First, if the two triangles have any vertex in common, they are discarted because they are neighbors. Second, if they are not neighbors and their bounding boxes do not self-intersect they are also discarted. And third, after passing the two previous tests a geometrical spatial intersection test is ran. If they collide, their *ids* are kept in the colliding list.

In Figure 3 we depict a representation of a tipical self-intersection during the mesh evolution process. On the left panel the arrows represent the displacement of the nodes in the iteration it_k. After evolving the nodes, on the center panel we see the new state of the mesh where one node has intersected other triangle which it is not its neighbor. When we detect a self-intersection, we rollback the whole mesh to the previous checkpoint (in this case it_k) and fix the nodes that where involved in the collision. Even though geometry is classified, this step is still quite expensive.On panel right we show this situation.

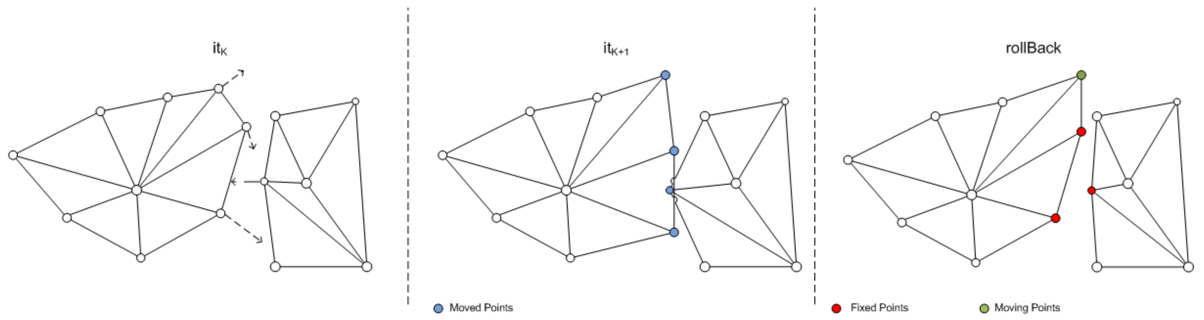

Figure 3: Self-intersection prevention. [left] Nodes movements, [center] collision detection, [right] collision resolution.

4 ACTIVE SURFACE WITH SELF-COLLISION DETECTION

Adding the self-collision detection technique previously explained to the active surface algorithm is not straight forward.

Toward the objective of avoiding collisions two other issues, apart from the self-collision detection, must be considered. The first, is to keep a *collision-free* safe state of the mesh to rollback the iterations when a collision is detected. And the second, is to define a strategy to prevent the triangles' collisions. These two operations (collision detection and mesh state saving and rollback) are **computally expensive** hence, the number of iterations between checkpoints must balance quality and efficience. We will call this free parameter of the algorithm $Each$.

We keep a copy of the last *collision-free* state and the iteration number that is kept in a proper data structure. So, when a control is performed, if no collisions are detected, we keep the current state as well as the iteration number. Otherwise, the previous safe state must be recovered, discarding the iterations and the *prevention strategy* must be applied.

Regarding the *prevention strategy*, we employed the simpliest solution. We freeze all the triangles' nodes evolution that collide and continue evolving after rolling back to a safe state. Thus, in the active surface algorithm an other data structure keeps track whether a point is enabled to evolve or not. This solution strongly limits the surface movement, and it does not take in account how it affects the final geometry; but results in closed conformant meshes.

4.1 Implementation

We use the Insight Segmentation and Registration Toolkit (ITK)[Ibanez et al. (2005)] which is an open-source system for medical imaging processing in C++. We implemented the Snake-filter as a standard MeshToMesh filter and the **SelfCollision-Detector** as a lightobject. Two different implementations of **SelfCollision-Detector** were programmed. The first one is single-threaded and the other one using the multi-threading support provided by ITK.

```
/* Check the initial mesh */
Initialize();
...
/* Save Initial free-collision state */
RollBack->SaveState( Mesh, nIteration )
while ( nIteration < max_Iterations and !Criterion)
{
    /* Evolve the mesh */
    Advance();
    /* Checking iteration */
    if( nIteration%Each == 0 )
    {
        /* Any Collisions? */
        if( ColliderDetector->EvaluateTriangles() )
        {
            /** Collisions : Get the list of triangles */
            Collisions = ColliderDetector->GetCollidingList();
            /* Freeze the triangles */
            CompleteFrozen(Collisions);
            /* Load the previous safe state */
            RollBack->GetState ( Mesh, &nIteration );
        } else // No collisions
        {
            /* Save the safe state */
            RollBack->SaveState(Mesh, nIteration);
        } /* End collide if */
        /* Clear the list */
        ColliderDetector->CleanTriangles();
    } /* End checking if */
    /* Check the Stopping Criterion */
    Criterion = StopCriterion();
    nIteration++;
} /* End while */
...
```

Figure 4: Pseudocode of active suface evolution with self-detection

Figure 4 shows a high-level pseudocode of the active surface evolution with self-colliding detection. Before starting the points evolution, the original mesh must be *collision-free*. The surface generation method, that takes the selected voxels from a previous process, checks that no collisions are generated. In this case we assure that the mesh is *collision-free*. So, the first safe state is kept and the evolution starts. In each iteration the points are evolved following equation 10, if the iteration number is a checkpoint, the triangle evaluation process is called. If no collisions are found, the mesh state is saved as the actual safe state as well as the iteration number. Otherwise, the list of collided triangles points is added to the frozen list and the previous safe state is recovered discarding the last $Each$ number of iterations.

5 EXPERIMENTS

We present two different experiments to address the correctness, the behaviour and the advantage of the self-collision detection strategy algorithm. The experiments were ran in an Intel(R) Core(TM) i7 CPU 860 @ 2.80 HGz (Quad Core) with Hyper-Threading, RAM memory: 8Gb DDR3 @ 1066 GHz, under the Linux 3.0.0.16 generic kernel(64bits).

5.1 Synthetic Evolution

The first proposed experiment shows empirically the desired property of the self-collision detection strategy algorithm. Five spheres of the same radius were placed at the same height, four of them form a square and the remaining is in the middle of the square but a bit closer to one of the others as shows Figure 5, left panel. All the spheres were considered as one mesh, and they were expanded during the whole process to provoke their collisions.

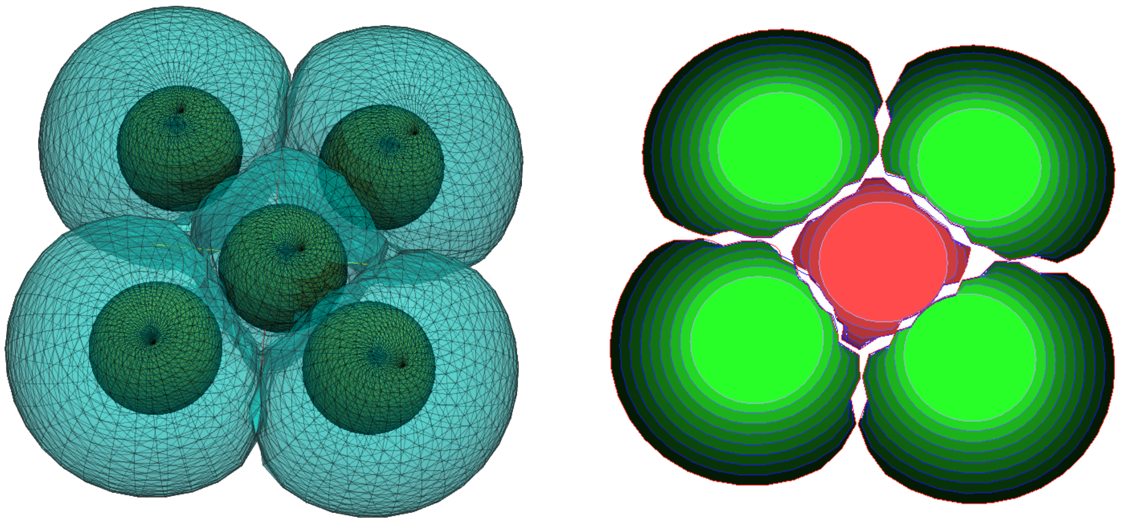

Figure 5: Synthetic evolution colliding detection. [left] The inicial state & final result checked each 5 it. and [right] evolutions slides shown each 50 it.

As the result of the synthetic experiment, the inner sphere ended squashed by the four exterior ones, and these last also remained deformed because of their approximation. The left panel of Figure 5 shows, in cyan, the final state of the mesh. In Figure 5 right panel we present colored slides of the mesh evolution each 50 iterations and its final position without collisions.

5.1.1 Checking Step Evaluation

The only free parameter of the detection algorithm is the number of iterations between checkpoints ($Each$). Concerning the evaluation of this parameter we performed the next experiment: We used 4 different checking steps, each 5, 10, 25 and 50 iterations. We computed 401 iterations in every simulation using these snakes parameters: $a = 1$ $b = 1$ $p = 0.0$ $q = 5$, step: $\Delta t = 0.01$. These 4 simulations were ran both with a single-threaded detector and again using the multi-threaded detector to compare their performance. The simulations results were exactly the same for the single-threaded and multi-threaded detector except for the total time they last.

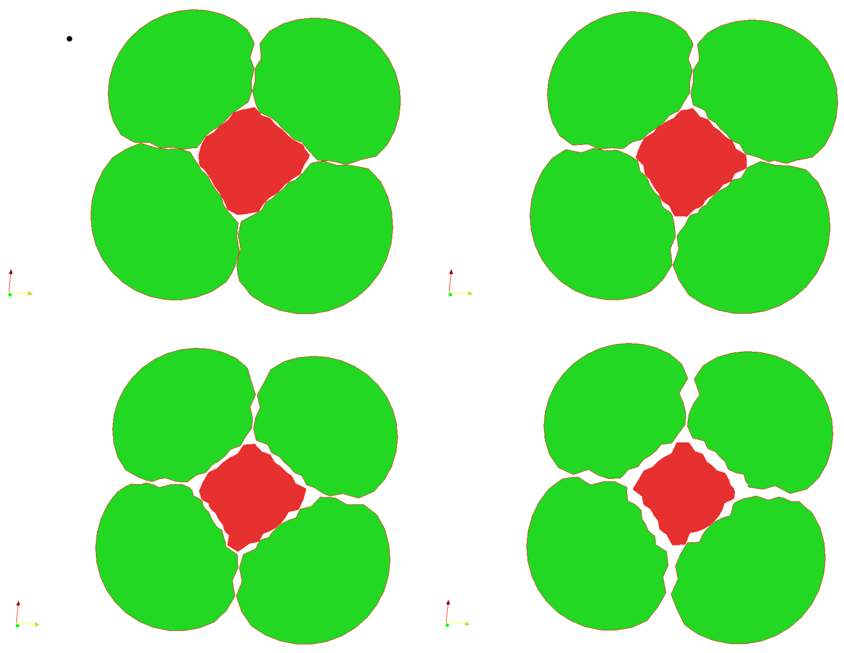

Figure 6: Quality vs. Checking Iterations Parameter (Each). Final results: [top-left] Each: 5 it., [top-right] Each: 10 it., [bottom-left] Each: 25 it., and [bottom-right] Each: 50 it.

In Figure 6 we can appreciate that the smaller the parameter $Each$ is, the better simulation quality we have. This result is quite evident because as we are using a freezing strategy, the minimum precision is inversely proportional to the number of iterations between checkpoints. Nevertheless, using a short $Each$ parameter increases the computational cost as we can appreciate in section 5.3

5.2 Real Case Evolution

In this section we present a real case situation of a rectum segmentation in a volumetric MRI study. We describe common artifacts in the mesh produced during the point evolution and how they are avoided thanks to the colliding detection technique.

5.2.1 Loops

In Figure 7 left panel we show a region of the mesh with a loop. The white line is a slide of the contour of the mesh. We ran the same simulation with the self-collision detector and as it can be seen in Figure 7 right panel, we eluded the loop. The same slide is spotted as the white line in the resulting mesh but now without loops. It is also important to remark that the rest of the mesh is almost similar.

5.2.2 Results

We computed the complete segmentation process previously described in section 2. For region growing stage, we used the parameters shown in Table 1 left panel. We needed 3 different seeds to obtain an accurate result in this stage. Next, we created the mesh, smoothed it with the Taubin filter and finally evolved it with the Snake filter with the self-colliding detector using the parameters of Table 1 right panel. The evolution process ran for 74 iterations until the stop

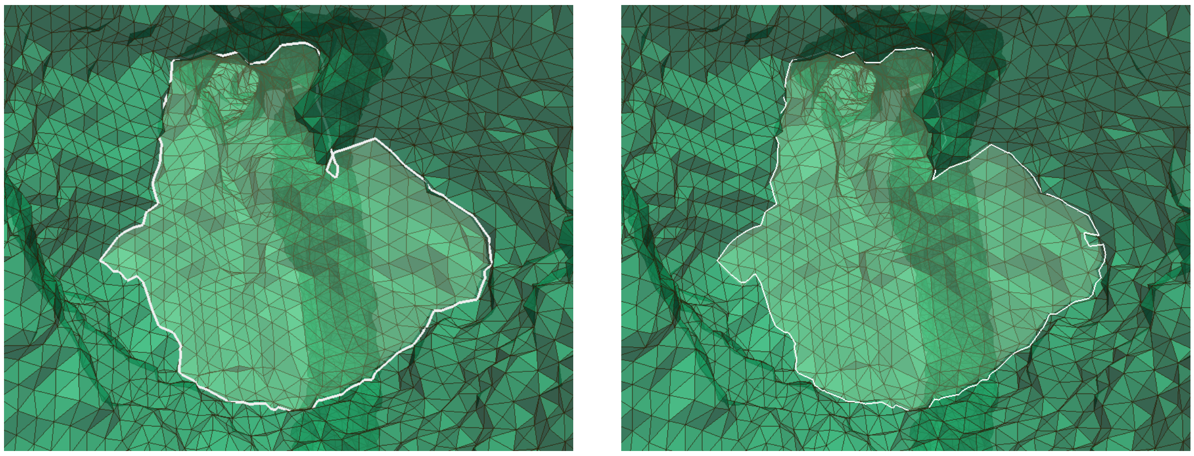

Figure 7: Real case loop collision prevention. [left] 3D loop collision and [right] avoided collision.

criterion was reached.

Region Growing:

Seeds	x	y	z
I	168	155	50
II	181	087	39
III	204	110	60
k:	2.0	f:	0.44

Snake:

a:	4.5	b:	4.0
p:	1.5	q:	2.3
f:	0.25	t:	0.25
Δt:	0.01	Each:	15

Table 1: Segmentation parameters.

As the result of the mesh evolution we obtain the surface representation of an organ, in this case a rectum without collisions. In Figure 8 we show 3 proyections of the MRI acquisition and in green the organ mesh surface representation. On the bottom-right panel we show a volume view of the MRI and the mesh surface of the rectum also in 3D.

5.3 Performance

We performed two different analysis to test the efficience of the multi-threaded implementation over the single-threaded one. The first one sublty analyzes the time spent at each part of the collision detector algorithm and the second one corroborates the efficience differences in the whole simulation process.

5.3.1 Subtle Part by Part Analysis

In this experiment we analyzed the time spent in each part of the self-collision detection algorithm in 5 checks during the rectum segmentation process of two different patients.

From Table 2 we can conclude that the speed-up factor of the parallelized stages of the algorithm is almost proportional to the number of cores (4) of the used computer. In the total spent time, the parallel implementation is approximately as twice faster than the secuential one. Parallelize the classification stage could give an extra speed-up to the whole process.

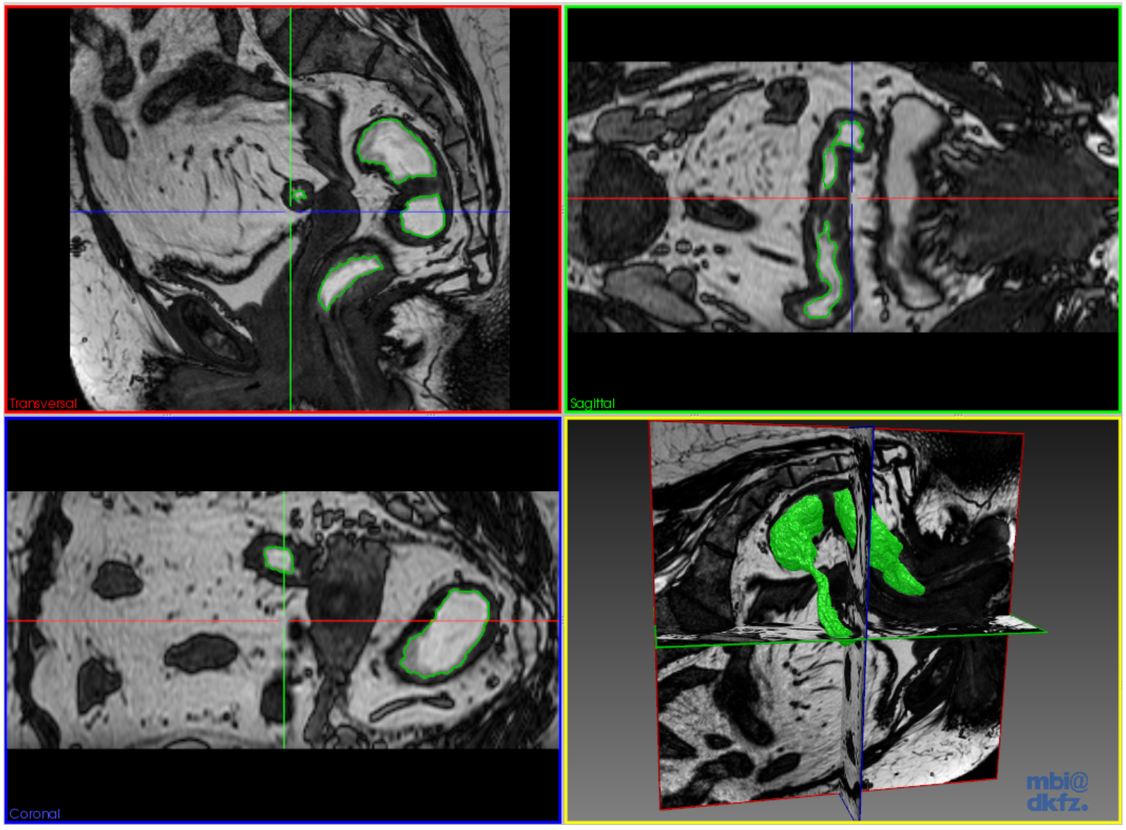

Figure 8: Final Mesh Model of the Patient's Rectum. [top-left] Axial View - [top-right] Sagital View - [bottom-left] Coronal View - [bottom-right] Volume View

	Rectum 13			Rectum 51		
Time	Secuential	Parallel	Spd-up Fac	Secuencial	Parallel	Spd-up Fac
Inicialization:	0.138694 s	0.033848 s	4.1	0.15074 s	0.005423 s	2.78
Classification:	0.462882 s	0.449207 s	1.03	0.552061 s	0.537174 s	1.03
Detection:	0.728716 s	0.193627 s	3.76	0.856607 s	0.236252 s	3.63
Total:	1.206402 s	0.646458 s	1.87	1.441756 s	0.778825 s	1.85

Table 2: Part by part efficience run-times comparison

5.3.2 Complete Simulation Comparison

We ran the syntetic simulation described in 5.1 varying the each parameter, and averaged five different simulations measures. This experiment not only stands the speed-up factor of the parallel implementation but also the computational overhead of the checking step $Each$ in the complete simulation process.

Figure 9 sumarizes the result of the experiment. First, we stand out the time overhead due to the self-collision detection technique. As the $Each$ parameter decreases, the total time of the simulation grows exponentialy (red and green bars) while the snake evolution time (grey) remains constant. Second, we can appreciate the time difference between the sequencial and parallel implementations. The more checkings we perform, the bigger the time difference is. As a conclusion of this experiment, in order to balance quality and efficience we propose to choose the $Each$ parameter between 10 and 25 iterations.

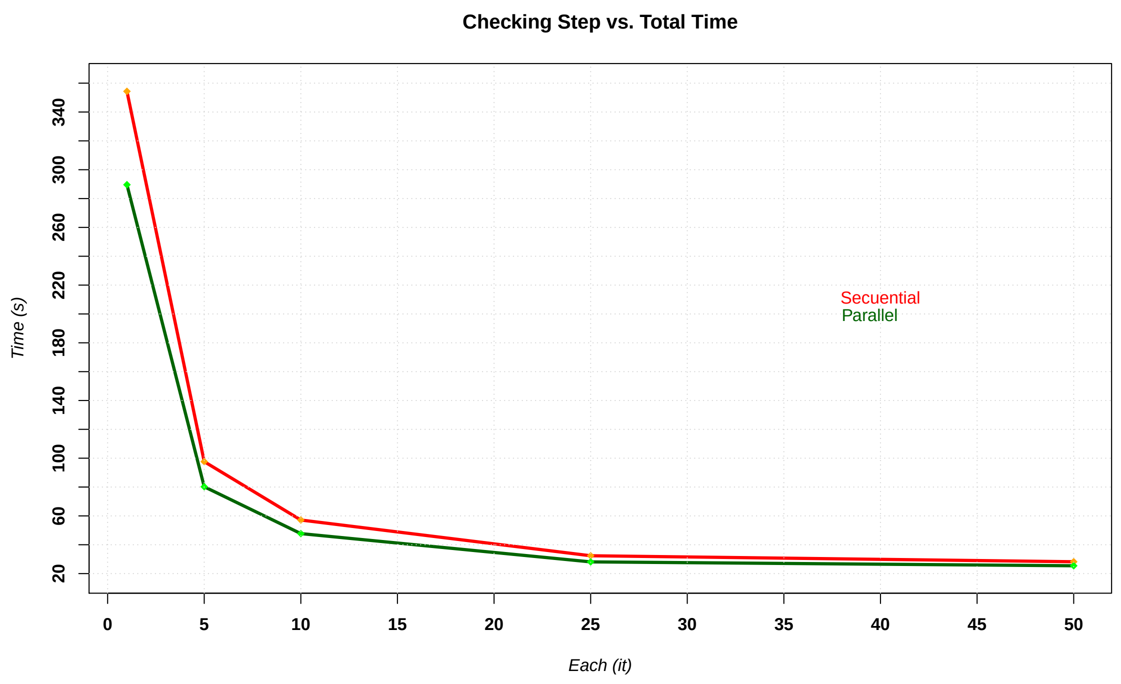

Figure 9: Total time simulation comparison: The colors (red, green) represent the collision detection time (secuential, parallel) and in grey the snake evolution time.

6 CONCLUSION

In this paper, we have presented a segmentation scheme based on active contours with geometry control. When self-intersections were detected, we propose to rollback to a "*safe state*", fix the nodes positions and continue the evolution. The proposed scheme effectively prevents the appearance of loops in the mesh, without having to limit the forces on the nodes as are proposed in other works. The resolution method applied is very simple and can be improved or replaced by another, thanks to the modular design that has been followed in developing the algorithm. The collision detection has a high computational cost, which may be reduced easily using a parallel architecture, with either multi-cores CPUs or GPUs. In upcoming works, this strategy will be used during the evolution of several snakes to solve the simultaneous segmentation of multiple tissues or organs.

REFERENCES

Bay T., Chambelland J.C., Raffin R., Daniel M., and Bellemare M.E. Geometric modeling of pelvic organs. In *Engineering in Medicine and Biology Society,EMBC, 2011 Annual International Conference of the IEEE*, pages 4329 –4332. 2011. ISSN 1557-170X. doi: 10.1109/IEMBS.2011.6091074.

Bischoff S. and Kobbelt L. Snakes with topology control. *The Visual Computer*, 2003.

Cohen L. On active contour models and balloons. *CVGIP: Image understanding*, 53(2):211–218, 1991.

del Fresno M., Vénere M., and Clausse A. A combined region growing and deformable model method for extraction of closed surfaces in 3d ct and mri scans. *Computerized Medical Imaging and Graphics*, 33(5):369 – 376, 2009. ISSN 0895-6111. doi:10.1016/j.compmedimag.2009.03.002.

Ibanez L., Schroeder W., Ng L., and Cates J. *The ITK Software Guide*. Kitware, Inc. ISBN 1-930934-15-7, http://www.itk.org/ItkSoftwareGuide.pdf, second edition, 2005.

Kass M., Witkin A., and Terzopoulos D. Snakes: Active contour models. *INTERNATIONAL JOURNAL OF COMPUTER VISION*, 1(4):321–331, 1988.

Lin Z., Jin J., and Talbot H. Unseeded region growing for 3d image segmentation. In *Selected papers from the Pan-Sydney workshop on Visualisation - Volume 2*, VIP '00, pages 31–37. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 2001. ISBN 0-909-92580-1.

Ma Z., Tavares J., Jorge R., and Mascarenhas T. A review of algorithms for medical image segmentation and their applications to the female pelvic cavity. *Computer Methods in Biomechanics and Biomedical Engineering*, 13(2):235–246, 2010.

Mcinerney T. and Terzopoulos D. Topology adaptive deformable surfaces for medical image volume segmentation. *IEEE Transactions on Medical Imaging*, 18:840–850, 1999.

Osher S. and Fedkiw R.P. Level set methods. In *in Imaging, Vision and Graphics*. Springer, 2000.

Pimenta S., Tavares J., Jorge R., Alexandre F., Mascarenhas T., and El Sayed R. Reconstruction of 3d models from medical images: Application to female pelvic organs. 2006.

Pohle R. and Toennies K. A new approach for model-based adaptive region growing in medical image analysis. In *Computer Analysis of Images and Patterns*, pages 238–246. Springer, 2001.

Singh A., Terzopoulos D., and Goldgof D. *Deformable Models in Medical Image Analysis*. IEEE Computer Society Press, Los Alamitos, CA, USA, 1st edition, 1998. ISBN 0818685212.

Taubin G. A Signal Processing Approach to Fair Surface Design. *Computer Graphics*, 29(Annual Conference Series):351–358, 1995.

Xiping L. and Jie T. Active contour based segmentation of medical image series. 2002.