Real-time Simulation of Self-collisions for Virtual Intestinal Surgery

Laks Raghupathi, Vincent Cantin, François Faure, Marie-Paule Cani

GRAVIR/IMAG, joint lab of CNRS, INPG, INRIA and UJF
INRIA RA, 655 avenue de l'Europe, Montbonnot
38334 Saint Ismier Cedex, France
Francois.Faure@imag.fr, Marie-Paule.Cani@imag.fr

Abstract. The context of this research is the development of a pedagogical surgery simulator for colon cancer removal. More precisely, we would like to simulate the gesture which consists of moving the small intestine folds away from the cancerous tissues of the colon. This paper presents a method for animating the small intestine and the mesentery (the tissue that connects it to the main vessels) in real-time, thus enabling user-interaction through virtual surgical tools during the simulation. The main issue that we solve here is the real-time processing of multiple collisions and self-collisions that occur between the intestine and mesentery folds.

Keywords: Surgical simulators, physically-based animation, soft tissue modeling, collision detection and response

1 Introduction

Enabling surgeons to train on virtual organs rather than on a real patient has recently raised a major interest for the development of pedagogical surgery simulators. Such simulators would be particularly useful in the context of minimally invasive surgery, where learning the right gestures while observing results on a screen causes major difficulties. The long term aim of this research is the development of a virtual-reality based simulator for minimally invasive colon cancer removal. Here, as the patient is resting on his back (Fig. 1), the small intestine is positioned just above the colon region, thus hiding the colon beneath. This requires the surgeon to interact with the intestine (by pulling and folding it) so that he can operate on the colon without any constraint. Hence, our aim is to simulate the behavior of the intestine when the surgeon is practicing in the virtual surgical environment. Note that the current scope of this research work does not include the simulation of the removal of the cancer itself.

The intestinal region of a human body is characterized by a very complex anatomy. The small intestine is a tubular structure, about 4 meters long, constrained within a small space of the abdominal cavity, resulting in the creation of numerous *intestinal folds*. This is further complicated by a tissue known as the

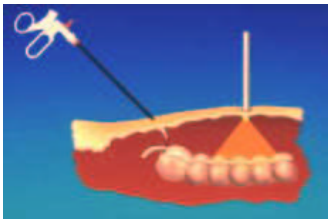


Fig. 1. Position of the small intestine when the patient is lying on his back

mesentery which connects the small intestine to the blood vessels. The mesentery suspends the small intestine within the abdominal cavity, at a maximal distance of 15 cm from the main vessels [1] (Fig. 2). Our challenge is to detect the collisions and self-collisions occurring in the intestinal region and to provide a realistic response at interactive frame rates.

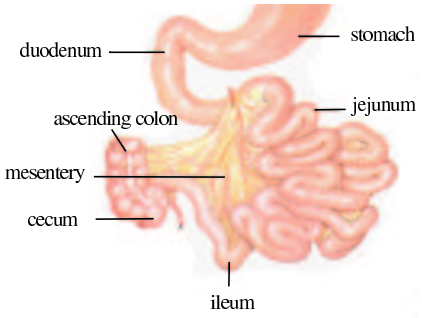


Fig. 2. Anatomy showing the intestine (*duodenum*, *jejunum* and *ileum*) and mesentery.

Section 2 describes the earlier works in the area, focussing on the main techniques for efficient collision detection and response. Section 3 describes our geometrical and mechanical models of the small intestine and the mesentery. We then describe our collision detection method and our novel approach for providing response in Sect. 4. This is followed by results in Sect. 5 and conclusions in Sect. 6.

2 Related Work

Recently, numerous researchers have focussed on the efficient simulation of deformable models [4, 5, 7, 9, 15, 16, 17, 23]. Several of them relied on adaptive, multi-resolution techniques for reaching real-time performances for complex volumetric bodies [4, 5, 9, 15]. In particular, some of these techniques were successfully applied to surgery simulators [8, 9, 16, 17, 22, 25]. In all these works, volu-

metric deformable bodies were simulated either in isolation, or were interacting with a single rigid tool, enabling the use of very specific techniques for collision detection and response, such as methods based on graphics hardware [19].

The problem we have to solve here is different: as will be shown in section 3, no volumetric deformable model will be needed since the intestine and the mesentery can be represented as a 1D and 2D structure respectively. Accordingly, a simple chain of masses and springs were used by France [12, 13] for simulating the intestine. France used a grid-based approach for detecting self-collisions of the intestine and collisions with its environment. All objects were first approximated by bounding spheres, whose positions were stored, at each time step, in the 3D grid. Each time a sphere was inserted into a non-empty voxel, new colliding pairs were checked within this voxel. Though this method achieved real-time performances when the intestine alone was used, it failed when a mesentery surface was added.

A well-known technique for accelerating collision detection consists of approximating the objects by a hierarchy of bounding volumes [3, 6, 14, 24, 28]. It enables to quickly get rid-off most not-intersecting cases. In particular, thanks to the tight-fitting volumes used, the OBB-trees [14] are known as the best representation for detecting collisions between volumetric rigid bodies. The hierarchies can be recursively updated when the objects undergo small deformations. However, this is not suitable for intestine-mesentery interaction where, even a small local deformation can cause a large movement of the folds. This creates a *global deformation* at large scale, which prevents the hierarchy from being efficiently updated. An alternate multi-resolution method, based on layered shells, was recently presented by Debunne [10]. It is well-suited for collision detection between deformable objects since the shells themselves are deformable structures extracted from a multi-resolution representation of these objects. Though suitable for volumetric deformable bodies, this method will not be appropriate for intestine and mesentery, since the time-varying folds cannot easily be approximated at a coarse scale.

Finally, Lin and Canny [18] exploited temporal coherence by detecting collisions between convex polyhedra by tracking pairs of closest vertices. These pairs were very efficiently updated at each time-step by propagating closest distance tests from a vertex to its neighbors. Debunne [10] adapted this technique for detecting collisions between his volumetric layered shells very efficiently. Since these shells were neither convex nor rigid, a stochastic approach was used at each time step to generate new pairs of points anywhere on the two approaching objects. These pairs were made to converge to local minima of the distance, disappearing when they reached an already detected minimum. Our work inspires from this idea of stochastic collision detection exploiting temporal coherence. It has been adapted, in our case, to the specific processing of multiple collisions and contacts between the intestine and the mesentery folds.

3 Modeling of the Intestinal System

3.1 Geometric Model

As shown in Fig. 2, the mesentery is a folded surface membrane, approximately 15 cm thick, which links the small intestine, a long tubular structure 4 m in length, to the main vessels of 10 cm length. Since the mesentery cannot be developed onto a plane, setting up its initial geometry free of self-intersections, is quite difficult. We solved the problem by approximating a possible rest position for the intestine as a folded curve lying at the surface of a cylinder of radius 15 cm. The axis of the cylinder, 10 cm in length, represents the main vessels. The folds are drawn on the cylinder such that their total length is 4 m (Fig. 3). Then the mesentery can be defined as the surface generated by a set of non-intersecting line segments linking the cylinder axis to the curve. Though this initial geometry is too symmetric to be realistic, it gives adequate local geometric properties to the mesentery membrane. This will enable the system to take correct arbitrary positions when animated under the effect of gravity. The geometry of the intestine is defined by creating tubular surface of radius 2 cm along its skeleton curve. The thickness of the mesentery membrane, which can be parameterized based on patient-specific data, was set to 1 cm.

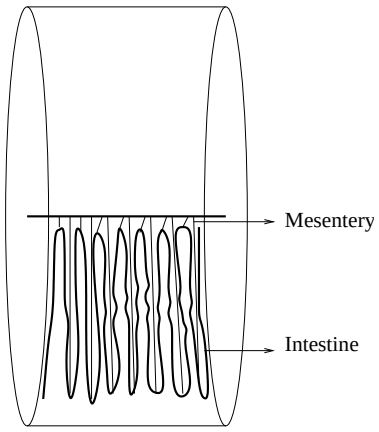


Fig. 3. Initialization of the geometric model of the intestine and the mesentery

3.2 Mechanical Model

The mechanical model representing the mesentery and its bordering curve, the intestine, should allow large displacements with local, elastic deformations. For animation, we used a simple mass-spring system since most of the computational time will be required for self-collision detection. Since the mesentery has a much

larger length (4 m near the intestine) than thickness (15 cm near the vessel), we sampled it by four sets of 100 masses each, connected by damped springs. The last set of masses requires no computation since they are attached to the main vessels, requiring only 300 masses to be integrated at each time step. No specific model is needed for the intestine since it can be simulated by adjusting the masses and stiffness values along the first bordering curve of the mesentery surface (depicted as a darker curve in Fig. 4). To increase robustness and efficiency, we relied on the integration method recently proposed by Lyard [20].

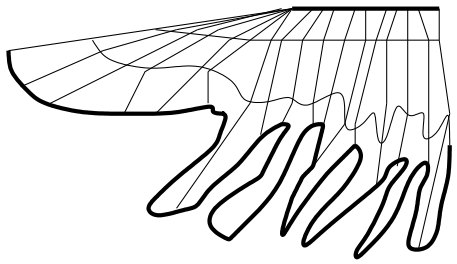


Fig. 4. Network of masses and springs used for the mechanical model

4 Real-time Collision Processing

4.1 Collision Detection

Our method for real-time collision detection exploits temporal coherence as in [10, 18], i.e., to track the pairs of closest points between the colliding bodies. The main differences here are: (1) the interacting objects have a tubular (intestine) and a membrane structure (mesentery), and (2) most collisions will be self-collisions between different folds of the same body. We first explain the collision detection method for the intestine alone, and then explain the mesentery case.

Collision detection between cylinders can be processed by computing the closest distance between their axes [11], and comparing it to the sum of their radii. For intestine, computing the distance between two segments is done by considering the distance between their principal axes. Then, we store the normalized abscissa (s, t) $(0 < s < 1,\ 0 < t < 1)$ of the closest points within the segments, and the corresponding distance d_{min}.

Adapting the notion of "closest elements pairs" to this skeleton curve means that we are willing to track the local minima of the distance between non-neighboring segments along the curve (Fig. 5). Of course, only the local minima satisfying a given distance threshold are of interest to us. We call these pairs of segments as "active pairs". Each active pair is locally updated at each time step, in order to track the local minima, when the intestine folds move. This is done

by checking whether it is the current segment pair or a pair formed using one of their neighbors which now corresponds to the smallest distance. This update requires nine distance tests (Fig. 6), and the pair of segments associated to the closest distance becomes the new active pair. When two initially distant active pairs converge to the same local minimum, one of them is suppressed. The pair is also suppressed if the associated distance is greater than a given threshold.

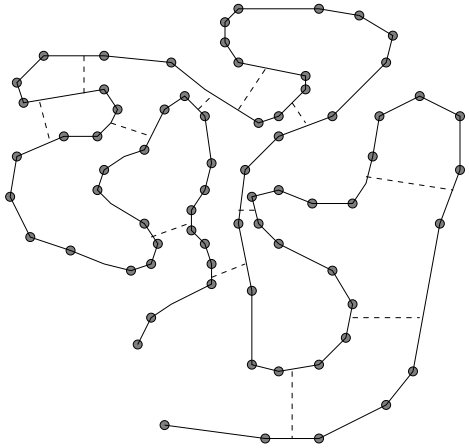


Fig. 5. Tracking of local minima of the distance between non-neighboring segments

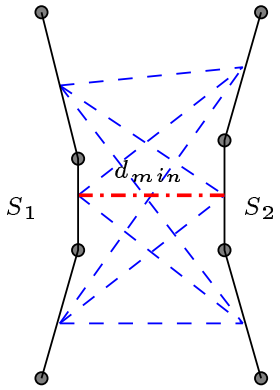


Fig. 6. Update of closest segment pairs of two adjoining intestinal folds

The above process tracks the existing regions of interest but does not the detect new ones. Since the animation of the intestine may create new folds nearby, a method for creating new active pairs of segments is needed. Our approach is

inspired from the stochastic approach of [10]. At each time step, in addition to the update of the currently active pairs, n additional random pairs of segments, uniformly distributed between the end-points but under the distance threshold, are generated. The update of these extra active pairs is similar to the update of the existing local minima, i.e., they are made to converge to a local distance minimum, the pair elements moving from a segment to one of its neighbors, and disappearing when an already detected minimum is reached. The complexity of the detection process thus linearly varies with user-defined parameter n. At each time step, collision detection consists in selecting, among the currently active pairs, the pairs of segments which are closer that the sum of their radii. Reaction forces, described in the collision response section, will then be generated between these segments.

For the mesentery, the total number of segments to be considered during each time-step is very large for real-time computation. Hence, we use the following approximation to reduce the complexity of the problem. First, since the mesentery is very thin and soft compared to the intestine, self-collisions of the membrane will almost have no effect on the overall behavior of the system. Hence, we neglect the testing of these collisions and only consider the pairs of segments from the intestine or pairs with one intestine segment and one non-neighboring mesentery segment.

Secondly, we use adaptive convergence to reduce the large number of distance computation required in this case. We first replace the first segment S_1 of the pair (S_1, S_2) by its closest neighbor S to S_2 (S is S_1 if all neighbors are farther than S_2). We then update S_2 by replacing it, if needed, by its neighbor which is the closest to S. This update requires 12 distance computations at most (i.e., when one segment belongs to the intestine, and the other to the inside of the mesentery). When a collision is detected, a recursive search starts across the neighbors to find all the colliding pairs in the area.

4.2 Collision Response

We initiate the response whenever the distance between the two segments is less than the sum of their radii. The earlier approaches such as penalty method [2, 27] and reaction constraint method [21, 26] implemented collision response by altering the force matrix in the mass-spring method. In our simulations, we observed that the stability of the system was reduced when we applied penalty and constraint methods.

Our new method alters the displacements and velocities of the two colliding segments in such a way so as to avoid interpenetration. Let the end-point velocities of segment S_1 be $\mathbf{v}_1$ and $\mathbf{v}_2$ and that of segment S_2 be $\mathbf{v}_1^{'}$ and $\mathbf{v}_2^{'}$ respectively. Let $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_1^{'}$ and $\mathbf{x}_2^{'}$ be the corresponding positions. Let $\mathbf{v}$ and $\mathbf{v}^{'}$ be the velocities of the closest approaching point within each segment (already stored in the neighborhood data structure) and $\mathbf{x}$ and $\mathbf{x}^{'}$ be the positions of the closest points (Fig. 7).

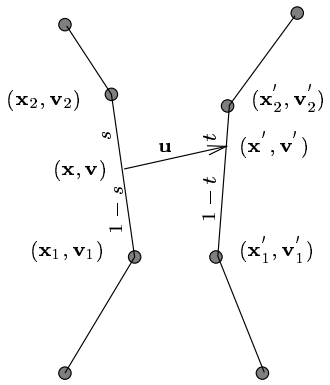


Fig. 7. Collision response by displacement-velocity correction

If s and t are the normalized abscissa of the closest points on the two segments we have:

$$\mathbf{v} = (1-s)\mathbf{v}_1 + s\mathbf{v}_2 \qquad \mathbf{v}^{'} = (1-t)\mathbf{v}_1^{'} + t\mathbf{v}_2^{'} \tag{1}$$

Let two forces per time-step, f and $f'(=-f)$, be applied along the direction of collision $\mathbf{u}$ to cause a change in the velocities such that the relative velocities along the direction of collision is zero. These forces should set the new velocities $\mathbf{v}_{new}$ and $\mathbf{v}_{new}^{'}$ to values satisfying the condition:

$$(\mathbf{v}_{new} - \mathbf{v}_{new}^{'}).\mathbf{u} = \mathbf{0} \tag{2}$$

The force f acting on the point of collision can be split between the end-points according to their barycentric coordinates. Expressing the new velocities in terms of the force and old velocities at the segment end-points yields:

$$\begin{aligned} \mathbf{v}_{new1} &= \mathbf{v}_1 + (1-s)f\mathbf{u} & \mathbf{v}_{new2} &= \mathbf{v}_2 + sf\mathbf{u} \\ \mathbf{v}_{new1}^{'} &= \mathbf{v}_1^{'} + (1-t)f\mathbf{u} & \mathbf{v}_{new2}^{'} &= \mathbf{v}_2^{'} + tf\mathbf{u} \end{aligned} \tag{3}$$

Again, expressing the new velocity of the colliding point $\mathbf{v}_{new}$ in terms of the end-point velocities $\mathbf{v}_{new1}$ and $\mathbf{v}_{new2}$:

$$\begin{aligned} \mathbf{v}_{new} &= (1-s)\mathbf{v}_{new1} + s\mathbf{v}_{new2} \\ &= \mathbf{v} + ((1-s)^2 + s^2)f\mathbf{u} \end{aligned} \tag{4}$$

Similarly for segment S_2:

$$\mathbf{v}_{new}^{'} = \mathbf{v}^{'} - ((1-t)^2 + t^2)f\mathbf{u} \tag{5}$$

Substituting the new velocity values from (4) and (5) into (2) and solving for f, we have:

$$f = \frac{(\mathbf{v}^{'} - \mathbf{v}).\mathbf{u}}{(1-s)^2 + s^2 + (1-t)^2 + t^2} \tag{6}$$

Using this value of f, we compute the new velocities of the end-points from (3). We use a similar formulation for correcting the positions of colliding segments. The only difference is in the condition for avoiding interpenetration, which takes the segment's radii r and r' into account:

$$(\mathbf{x}_{new} - \mathbf{x}^{'}_{new}).\mathbf{u} = r + r' \tag{7}$$

The force value g to change the positions in order to enforce the above condition, is then:

$$g = \frac{(\mathbf{x} - \mathbf{x}^{'}).\mathbf{u} + r + r'}{(1-s)^2 + s^2 + (1-t)^2 + t^2} \tag{8}$$

g is used for modify the positions $\mathbf{x}_{new1}$, $\mathbf{x}_{new2}$, $\mathbf{x'}_{new1}$ and $\mathbf{x'}_{new2}$ of the segments end points using similar expressions as in (3).

5 Results

5.1 Validation

In order to compare the effectiveness of this method, we developed a simple testing methodology for case of the intestine in isolation. We would like to know if our algorithm detects all the regions of collisions. To do so, we compared our method with a naive $O(n^2)$ approach (i.e., do a check of all possible pairs to detect all the active collision regions). So, we let the simulation run and took snapshots of the model during different time intervals (Fig. 8a and 8b). At the same time, we also collected data on the segment pairs (index values) stored in the neighborhood data structure. We carried out this procedure for both the methods. During the simulation, the mass points were resting on a plane. Figures 8b and 8d plot the values of the active segment-pair indices for the two configurations (all segments under the distance threshold in the case of the $O(n^2)$ detection). Results show that since our method only tracks local minima of the distance, it considers much fewer segments (Table 1). They also show that all regions of interest are adequately detected, i.e., there are no colliding folds with no detected local minima, as depicted in Fig. 8b. The corresponding plot in Fig. 8d shows an increased density of points in the local minima region, thereby indicating that it has been detected by the algorithm. The resulting animations also showed no interpenetration with realistic collision response. The figure only shows the collisions detected by the converging pairs. Once a pair converges to a collision, the recursive propagation ensures that all the collisions of a collision region are detected. Our method can detect all the regions of collisions at 30 Hz on a PC. Comparing with the $O(n^2)$ method, this method uses a relatively small number segment-pairs and is hence a lot faster.

5.2 Qualitative Results

Snapshots from the real-time animation of the intestinal system including both the small intestine and the mesentery are depicted in Fig. 9. Note that realistic

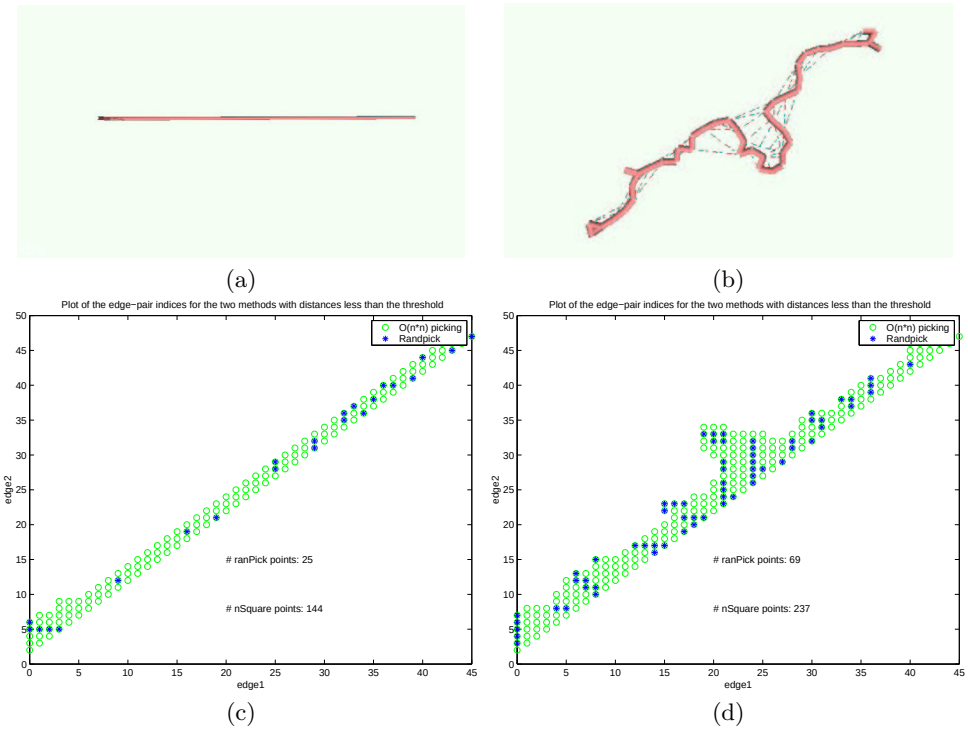


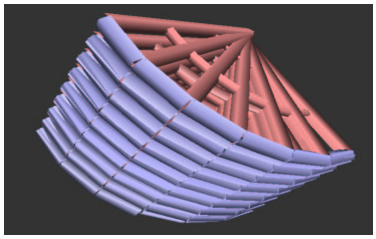
Fig. 8. (a) and (b) Snapshots of the simulation of an isolated intestine. (c) and (d) Plot of the active pairs of segments compared with the pairs from the $O(n^2)$ method.

Table 1. Running time (in ms) for both the methods

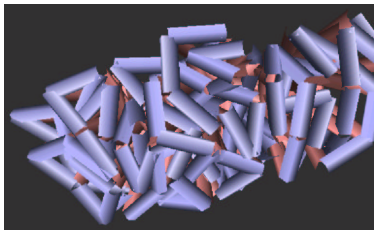
Number of segments	Time (ms)	
	Our method	$O(n^2)$
50	10	30
100	17	120
200	27	473

rendering is not our concern here. Our displacement-velocity method for collision response produces highly stable simulations.

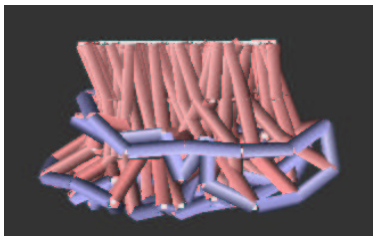
As a companion to this paper, a dynamic real-time demonstration of our results is available at: `http://www-imagis.imag.fr/Membres/Francois.Faure/papers/intestine/index.html`.



(a) The system in its initial state with the intestine represented by the curve along the cylindrical surface edge and the mesentery by segments connecting it to the cylinder axis.



(b) The intestine has reached a more plausible shape.



(c) The intestine makes a loop around the mesentery (medically unrealistic).

Fig. 9. Snapshots from our real-time simulator of the intestinal system.

6 Conclusion

We have developed a model which can accurately determine all the active regions of self-collisions in the intestine and the mesentery. We have also developed a

method for providing realistic and stable collision response. Our models run at interactive frame rates which can be used in a virtual surgical environment.

Future work will include the incorporation of these algorithms in the intestinal surgery simulator developed by our collaborators in Lille [12, 13], thus enabling the use of convincing geometric coating, texturing and rendering of the organs, in addition to the use of a force-feedback device for user-interaction.

Acknowledgments

This work is supported by INRIA (French National Institute for Research in Computer Science and Control) as part of the ARC SCI (research action for Intestine Surgery Simulator). The authors would like to thank Luc Soler (IRCAD) for his insights and suggestions for creating the geometry of the mesentery and for providing anatomical information and Laure France (LIFL) for fruitful discussions and for providing data from her earlier research.

References

[1] L. Augusten, R. A. Bowen, and M. Rouge. *Pathophysiology of the Digestive System - Hypertexts for Biological Sciences.* Colorado State University, Fort Collins, CO, `http://arbl.cvmbs.colostate.edu/hbooks/pathphys/digestion/index.html`, 2002.

[2] D. Baraff and A. Witkin. Large steps in cloth simulation. In *Proc. SIGGRAPH '98*, pages 43–54. ACM Press, July 1998.

[3] G. Bradshaw and C. O'Sullivan. Sphere-tree construction using dynamic medial axis approximation. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 33–40. ACM Press, July 2002.

[4] S. Capell, S. Green, B. Curless, T. Duchamp, and Z. Popović. Interactive skeleton-driven dynamic deformations. In *Proc. SIGGRAPH '02*, pages 586–593. ACM Press, July 2002.

[5] S. Capell, S. Green, B. Curless, T. Duchamp, and Z. Popović. A multiresolution framework for dynamic deformations. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 41–48. ACM Press, July 2002.

[6] J. D. Cohen, M. C. Lin, D. Manocha, and M. K. Ponamgi. I-COLLIDE: An Interactive and Exact Collision Detection System for Large-Scale Environments. In *Symposium on Interactive 3D Graphics*, pages 189–96, 1995.

[7] S. Cotin, H. Delingette, and N. Ayache. Real-time elastic deformations of soft tissues for surgery simulation. *IEEE TVCG*, 5(1):62–73, March 1999.

[8] S. Cotin, H. Delingette, and N. Ayache. A hybrid elastic model for real-time cutting, deformations, and force feedback for surgery training and simulation. *The Visual Computer*, 16(8):437–452, 2000.

[9] G. Debunne, M. Desbrun, M. P. Cani, and A. H. Barr. Dynamic real-time deformations using space and time adaptive sampling. In *Proc. SIGGRAPH '01*, pages 31–36. ACM Press, August 2001.

[10] G. Debunne and S. Guy. Layered Shells for Fast Collision Detection. To be published, 2002.

[11] D. H. Eberly. *3D Game Engine Design: A Practical Approach to Real-Time Computer Graphics*. Morgan Kaufmann, 2000.

[12] L. France, A. Angelidis, P. Meseure, M. P. Cani, J. Lenoir, F. Faure, and C. Chaillou. Implicit representations of the human intestines for surgery simulation. In *Modelling and Simulation for Computer-aided Medicine and Surgery*, Rocquencourt, France, November 2002.

[13] L. France, J. Lenoir, P. Meseure, and C. Chaillou. Simulation of a minimally invasive surgery of intestines. In *Virtual Reality International Conferenc*, Laval, France, May 2002.

[14] S. Gottschalk, M. C. Lin, and D. Manocha. OBBTree: a hierarchical structure for rapid interference detection. In *Proc. SIGGRAPH '96*, pages 171–80. ACM Press, 1996.

[15] E. Grinspun, P. Krysl, and P. Schröder. CHARMS: A Simple Framework for Adaptive Simulation. In *Proc. SIGGRAPH '02*, pages 281–290. ACM Press, July 2002.

[16] D. L. James and D. K. Pai. Artdefo - accurate real time deformable objects. In *Proc. SIGGRAPH '99*, pages 65–72. ACM Press, August 1999.

[17] D. L. James and D. K. Pai. DyRT: Dynamic Response Textures for Real Time Deformation Simulation with Graphics Hardware. In *Proc. SIGGRAPH '02*, pages 582–585. ACM Press, July 2002.

[18] M. C. Lin and J. F. Canny. Efficient Collision Detection for Animation. In *Proc. of the 3rd Eurographics Workshop on Animation and Simulation*, 1992.

[19] J. C. Lombardo, M. P. Cani, and F. Neyret. Real-time Collision Detection for Virtual Surgery. In *Proc. Computer Animation '99*, May 1999.

[20] E. Lyard and F. Faure. Impulsion springs: a fast and stable integration scheme dedicated to the mass-spring model. To be published, 2002.

[21] D. W. Marhefka and D. E. Orin. Simulation of contact using a nonlinear damping model. In *Proc. IEEE ICRA*, pages 1662–68, 1996.

[22] P. Meseure and C. Chaillou. A deformable body model for surgical simulation. *Journal of Visualization and Computer Animation*, 11(4):197–208, September 2000.

[23] M. Müller, J. Dorsey, L. McMillan, R. Jagnow, and B. Cutler. Stable real-time deformations. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 49–54. ACM Press, July 2002.

[24] I. J. Palmer and R. L. Grimsdale. Collision detection for animation using sphere trees. *Computer Graphics Forum*, 14(4):105–16, May 1995.

[25] G. Picinbono, J. C. Lombardo, H. Delingette, and N. Ayache. Improving realism of a surgery simulator: linear anisotropic elasticity, complex interactions and force extrapolation. *Journal of Visualization and Computer Animation*, 13(3):147–167, 2002.

[26] J. C. Platt and A. H. Barr. Constraints methods for flexible models. In *Proc. SIGGRAPH '88*, pages 279–88. ACM Press, 1988.

[27] D. Terzopulous, J.C. Platt, K. Fleischer, and A. H. Barr. Elastically deformable models. In *Proc. SIGGRAPH '87*, pages 205–14. ACM Press, 1987.

[28] G. van den Bergen. Efficient Collision Detection of Complex Deformable Models using AABB Trees. *Journal of Graphics Tools*, 2(4):1–14, 1997.