Simulation of Sutures for Virtual Surgery Applications

Hans-Georg Menz Kup-Sze Choi[1]
Ctr. for Int. Digital Health, School of Nursing
The Hong Kong Polytechnic University, Hong Kong
hskschoi@inet.polyu.edu.hk

Abstract

Suturing is a basic operative procedure and real-time interactive simulation of sutures is thus a key component in virtual surgical simulator. Realistic simulation depends on the modeling and collision management of sutures. This paper reports a study on the use of bounding volume hierarchies to detect collisions of one-dimensional objects. Experiments are conducted to evaluate the performance of bounding spheres, axis-aligned bounding boxes and oriented bounding boxes. Among them, it is found that axis-aligned bounding box is a more appropriate abstraction of the straight line segments used for modeling suture threads. While the primary motivation of the investigation lies on the simulation of sutures, the finding is applicable to one-dimensional elastic objects in general.

1. Introduction

Suturing and knot typing are fundamental tasks in surgery. They are involved in various operations, from abdominal wall closure, laparoscopic suturing, to accurate suture placement through vessel wall in vascular surgery. To master the skills, it is necessary to practice repeatedly under different situations. A promising approach to facilitate suturing education is to simulate the procedures using computer graphics and virtual reality technology so that training can be conducted conveniently at any time in programmable virtual environments. The modeling and simulation of sutures and knots play a major role in the development process of virtual suturing simulators. The work can be divided into suture modeling and collision handing. The dynamics of sutures due to gravity, inter- or intra-suture interactions, and other forces are taken into account to compute and simulate their response. In particular, efficient collision detection algorithms are critical to real-time performance of the simulators. This paper concerns the collision detection techniques that are appropriate for the simulation of one-dimensional object like sutures. Hierarchical approaches commonly used in computer graphics, including bounding spheres, axis-aligned bounding boxes (AABB) and oriented bounding boxes (OBB), are investigated with experiments conducted to evaluate their effectiveness specifically in detecting the collision of one-dimensional objects.

In this paper, modeling of suture and the mathematical formulation are first described in Section 2. Next, the techniques used to handle the collisions of suture, i.e. the detection of collisions and the responses to those detected collisions, are reviewed in Section 3. Investigations on various collision detection techniques are then discussed in Section 4, where experiments are conducted to compare their performance. Discussion and conclusion are given in Section 5 and 6 respectively.

2. Suture modeling

A continuous suture thread can be considered a system of discrete line segments connected by nodes. A purely kinematic approach that models suture as a chain of mass points connected by rigid links has been developed to simulate the motion of a virtual thread [1]. The forces between successive pairs of nodes are not considered however. More realistic approaches based on Mass-Spring Model (MSM) have been developed where elastic springs are employed to join the discrete segments [2-4]. As shown in Figure 1, a piece of suture is modeled by a number of straight line segments connecting nodes $\mathbf{r}_i$. Each line represents a spring. The spring force $\mathbf{F}^s$ on the segment connecting $\mathbf{r}_i$ and $\mathbf{r}_{i+1}$ is given by

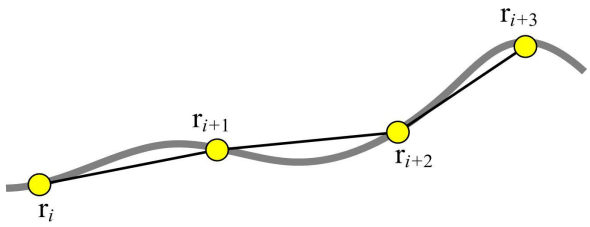


Figure 1: Discretization of continuous thread into segments.

$$\mathbf{F}^{s}_{\mathbf{r}_i-\mathbf{r}_{i+1}} = -\left(\frac{\mathbf{r}_i - \mathbf{r}_{i+1}}{\left\|\mathbf{r}_i - \mathbf{r}_{i+1}\right\|} \cdot \left\|\mathbf{r}_i - \mathbf{r}_{i+1}\right\| - \left\|\mathbf{r}_i^0 - \mathbf{r}_{i+1}^0\right\| \right) \cdot c_s, \quad (1)$$

where $\mathbf{r}_i$ is the current position of the node and $\mathbf{r}_i^0$ is the initial position, c_s is the spring constant. The damping force $\mathbf{F}^f$ exerted on the segment can be modeled as

$$\mathbf{F}^{f}_{\mathbf{r}_i-\mathbf{r}_{i+1}} = -\left(\mathbf{v}_{\mathbf{r}_i} - \mathbf{v}_{\mathbf{r}_{i+1}}\right) \cdot c_f \quad (2)$$

where c_f is the damping constant and $\mathbf{v}$ is the velocity of corresponding nodes. The resultant force $\mathbf{F}^t$ is the vector sum of $\mathbf{F}^s$ and $\mathbf{F}^f$ and other external forces like gravity. Taking into account the equilibrium of forces and energy conservation, the forces have to be applied at both end points of the spring, but in opposite directions, i.e.

$$\mathbf{F}_{\mathbf{r}_i-\mathbf{r}_{i+1}} = -\mathbf{F}_{\mathbf{r}_{i+1}-\mathbf{r}_i} \quad (3)$$

The velocity and position of the nodes in the thread can be obtained by solving Equation (4) and (5) with numerical integration techniques, where m_i is the mass of node $\mathbf{r}_i$ and h is the integration time step.

$$\mathbf{v}_{\mathbf{r}_i}^{t+h} = \mathbf{v}_{\mathbf{r}_i}^{t} + \frac{h}{m_i}\mathbf{F}^t \quad (4)$$

$$\mathbf{r}_i^{t+h} = \mathbf{r}_i^{t} + h\mathbf{v}_{\mathbf{r}_i}^{t+h} \quad (5)$$

This approach is straight forward and easy to implement for one-dimensional object. However, twisting of thread around the centerline is not considered and the mathematical representation could be quite complicated otherwise. Twisting can achieved by joining the nodes with torsion springs [4, 5].

A a more elegant method is to model one-dimensional object as a continuum by employing the Cosserat theory of elasticity, which considers a piece of thread as a thin elastic deformable object [6] and the orientation of each segment is represented explicitly with quaternion. The discrete potential and dissipation energies, both linear and angular, are computed to determine the dynamic behavior of threads [7].

3. Suture collision

Collision handling has been an active area in computer graphics and many methods are available. While most of them are generic and suitable for arbitrary settings, research specific to the collision handling of simulated threads is relatively few. An example is the work of Brown et al. [1] who proposed specialized contact detection and management approaches for real-time rope and knot-tying simulation.

3.1. Detection of collisions

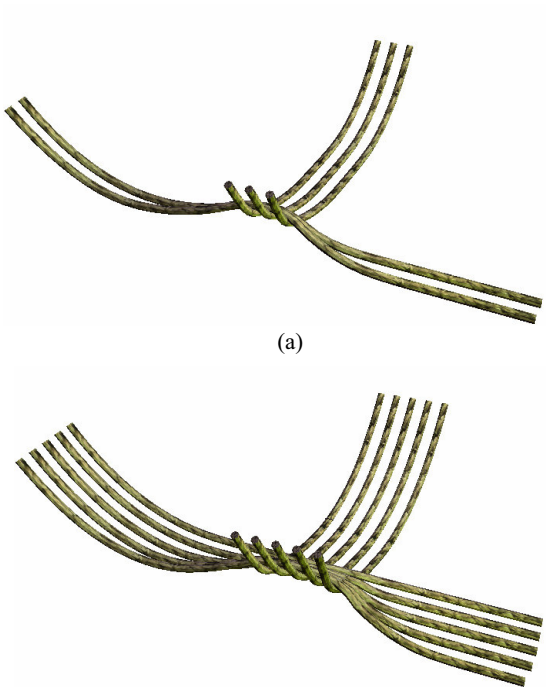
For the collision of virtual objects represented by triangulated or tetrahedral meshes, it is necessary to determine the contact point and penetration depth into a triangle in order to detect collisions. In contrast, since the primitives of suture thread can be simply considered as straight line segments, it is appropriate to detect collision by determining the distance between two lines segments, which can be obtained readily by solving the parametric equations of the corresponding line segments.

Real-time suturing simulation cannot afford to perform one-to-one checking on every segments of simulated thread for collisions. The process can be accelerated by employing Bounding Volume Hierarchies (BVH) to abstract the segments inside bounding volumes and arrange them with a hierarchical tree structure of various levels of detail. The process of collision detection begins by searching from a rough approximation of the thread, recursively down through the branches to a certain leaf that bounds the exact segment where collision has occurred, if any. The effectiveness of BVH on collision detection depends on the structure of object and the shape of bounding volume. For suture simulation, it is important to choose a proper bounding volume that matches the segmented structure of the discretized thread.

3.2. Response to collisions

After all collisions have been detected, the simulated thread has to respond accordingly by displacing the colliding segments in an autonomous manner. Here, a discretized suture thread is considered as a chain of rigid cylinders, each with certain radius and height. The direction of separation is a key factor determining how the collided threads are separated. It can be approximated by the shortest vector distance between the collided segments in the threads, which is indeed a coarse approximation and might lead to strong artifacts. A better way is to determine the relative velocity of the colliding segment, which would just separate the segments exactly at the point where they collide in the best case.

The interactions among collided threads can be simulated by employing the penalty method [8] to apply separation forces to the collided segments; or by using the impulse-based method [9] to change their velocities directly in order to move them apart. In the penalty method, the magnitude of separation forces is usually calculated based on the penetration depth into the collided segments. A major problem here lies on the difficulty to estimate the appropriate magnitude of the penalty forces so that they are large enough to counteract all the forces exerted on a collided segment. In some cases, it takes multiple time steps to separate the collided threads. While the impulse-based method is relatively more robust and stable than the penalty method, it may not simulate threads in resting contact properly due to the lack of analytical force model.



(a)

(b)

Figure 2: Simulated threads coming into contact. (a) With one end fixed, the 2 threads above are set to fall under gravity by releasing the other end. The threads then collide with the 3 threads underneath with both ends fixed. (b) Under the same settings, the 5 threads above come into contact with the 5 threads below.

4. Performance evaluation

As discussed in Section 3.1, collision detection speed depends on the structure of colliding object and the choice of appropriate shape of bounding volume. In this paper, three typical bounding volume hierarchies, including bounding spheres, axis-aligned bounding boxes and object-oriented bounding boxes, are investigated to study their performance specifically in handling the collision detection of one-dimensional threads. In all the simulation results presented below, MSM was adopted to model suture threads as a chain of mass points connected by elastic springs. The mathematical formulation described in Section 2 was followed. Implicit Euler method was employed to solve for the positions and velocities of the nodes during the simulation, i.e. Equation (4) and (5). Penalty method was utilized to compute the response of the collided threads. The simulations were implemented on an Intel E6600 Core 2 Duo 2.4 GHz personal computer with 2 GB RAM.

In general, the whole BVH is updated at each simulation step, which is computationally expensive for large hierarchical trees and may involve unnecessary update to some of the branches. In this regard, an selective update method which modifies the parts of a hierarchical tree that are relevant to the current step is adopted [10]. The method requires each node in the tree to store the information about the primitives it covers. If a bounding volume of a tree intersects with a bounding volume of another tree, all children in the next layer of the corresponding node are updated, and thus only the relevant bounding volumes are modified.

4.1. Bounding volume hierarchies

To study the performance of BVHs, experiments were conducted to measure the time required to simulate the collisions between two sets of parallel threads, with one set placed underneath the other. The situation is depicted in Figure 2. Both ends were fixed for the set of threads underneath, while only one end was fixed for the set of

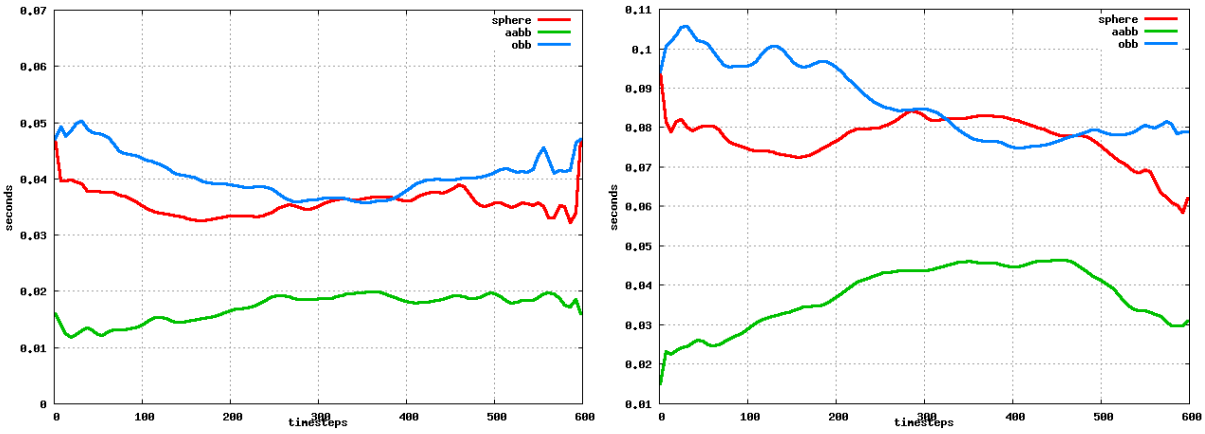


Figure 3: Collision detection in a system involving a total of 400 (left) and 800 (right) segments for modeling simulated threads.

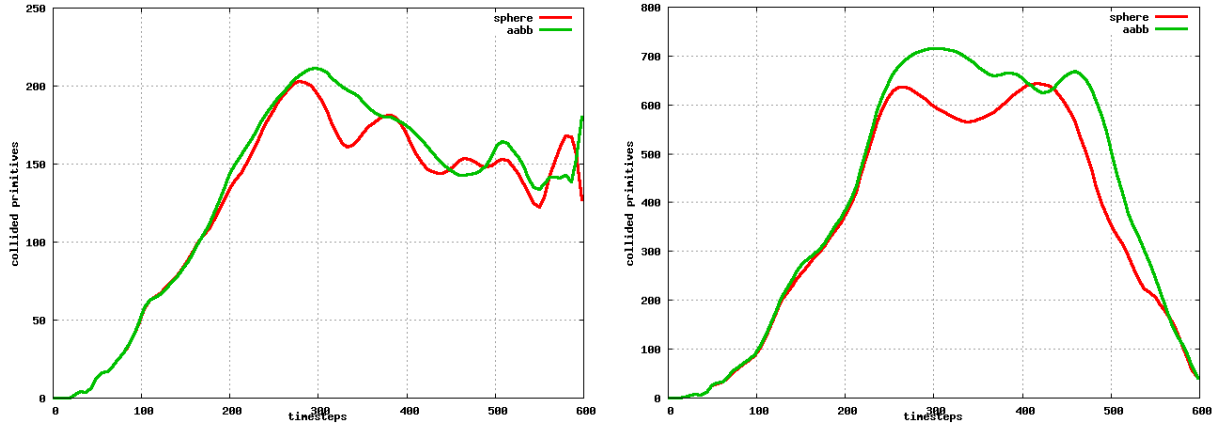


Figure 4: Number of primitives collided for a system containing 400 (left) and 800 (right) segments.

threads above and the other end was set to fall freely under gravity. That is, the threads above would come into contact with the threads below. The effect of the total primitive count on the simulation complexity was investigated by varying the number of threads involved in the experiment. Here, each thread was modeled with 40 straight line segments and different numbers of threads were allowed to collide with each other. For example, simulation involving a total of 400 segments was conducted by letting 5 threads above to come into contact with another 5 threads below. It was repeated with 10 threads colliding with another 10 threads to simulate the situation where 800 segments were involved. Figure 3 shows the performance of the three BVHs when the total number of primitives involved in the simulation is 400 and 800 respectively. The results indicate that among the three BVHs, collision detection handled with AABBs performs better than that managed by bounding spheres and OBBs, while the performance of the latter two approaches is at similar levels.

With specific attention to bounding spheres and AABBs, the corresponding number of collided segments at each time step of the simulation was recorded for investigation. From the measurement shown in Figure 4, it is noted that, in some cases, the number of collided segments reaches a maximum value and then returns to zero at the end of simulation, while it stays at some non-zero values in other cases. The former refers to the situation where the two set of threads separate again other after having touched each other; whereas the latter corresponds to the fact that the two set of threads remains in contact at the end of simulation.

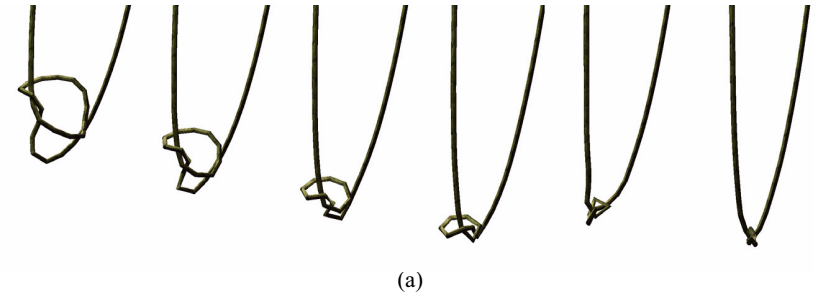
4.2. Knot formation

Besides, experiment was performed by using bounding spheres and AABBs respectively to simulate the formation of knot on a thread. The thread had been initially configured so that a knot could be formed autonomously by letting the thread falling freely in space under gravity. A sequence of snapshots during the formation of knot is shown in Figure 5(a). Simulation was performed by using a thread modeled with 60 line segments. The timing performance is shown in Figure 5(b). The result indicates that the speed of simulation using AABBs for collision detection is faster than collision detection using bounding spheres, which agrees with the finding presented in the previous sub-section. In both cases, the number of collided primitives increases gradually at the beginning, corresponding to the initial stages of knot formation where self-collision is relatively few. The number then increases more sharply at later stages when the knot is being formed and the segments representing the knots collide among themselves.

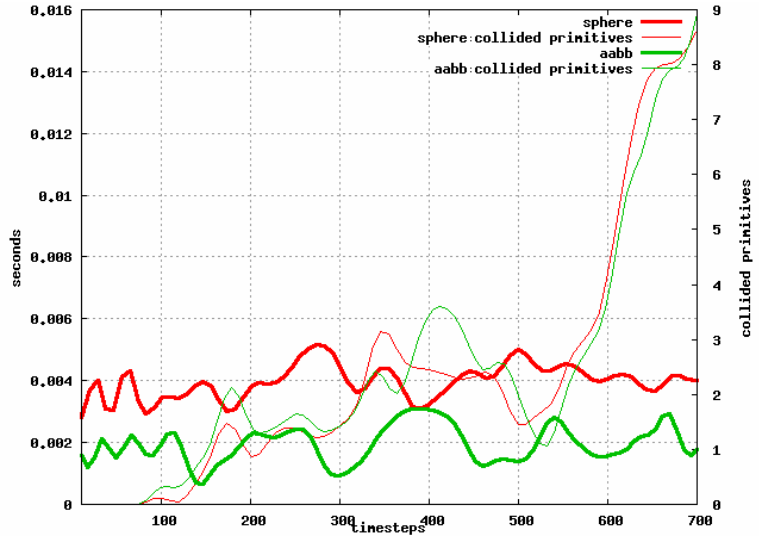
5. Discussion

Among the BVHs experimented for suture simulation, collision test of bounding spheres is most convenient and efficient. A leaf bounding sphere is simply defined by a circumscribing sphere that touches the endpoints of a segment and is centered at the mid-point of that segment. Only one single test is required to check whether two bounding spheres intersect, i.e. to test if the distance between two spheres is less than the sum of their radii. Despite the simplicity and efficiency, approximating straight line segments with spheres is the worst case scenario for this kind of bounding volume, leading to frequent "false-positive" indications of segment collisions although the bounding spheres do intersect between themselves. This agrees with the finding shown in Figure 3 that bounding sphere hierarchy is not the fastest collision detection approach for simulating virtual sutures.

Like bounding spheres, collision detection using AABBs



(a)



(b)

Figure 5: (a) Sequence of snapshots captured during the process of knot formation. (b) Timing performance of knot formation by using bounding spheres and AABBs for collision detection.

is also simple and efficient but the latter is a better approximation to straight line segments (see the first row of Figure 6). Aligned with the coordinate axes, AABBs are further defined by having the endpoints of a segment touching two diagonally opposite corners of an AABB [11]. To check whether two AABBs intersect, it is necessary to make six tests by comparing the corner coordinates for the two AABBs. On the other hand, OBB fits a line segment most tightly among the bounding volumes used in the other BVHs (see the second row of Figure 6). The update of hierarchical tree is, however, complicated by the need to determine new box orientation based on that of the parent and children OBBs. Furthermore, the intersection test is also more complicated, which is the generalization of the intersection test for AABBs, e.g. Separating Axis Theorem [12]. It requires fifteen tests for collision detection, involving 6 face normals of OBBs and 9 cross products of the edge vectors. Although OBBs provides the most accurate approximation to the underlying thread, the intersection test and the update process are more complex. On the other hand, from the perspective of memory storage, six scalars are required to define an AABB while fifteen scalars are used to represent an OBB. Hence, OBB tree consumes more memory than AABB tree for the same number of bounding volumes, and AABB is advantageous over OBB in terms of memory storage. From the experimental results and the analyses discussed above, it is suggested that AABB is an optimal solution to handle the collisions of one-dimensional suture threads.

6. Conclusion

The paper concerns the simulation of suture threads for virtual surgery application. After the discussions on the modeling of one-dimensional objects, the collision detection and response handling approaches, the paper

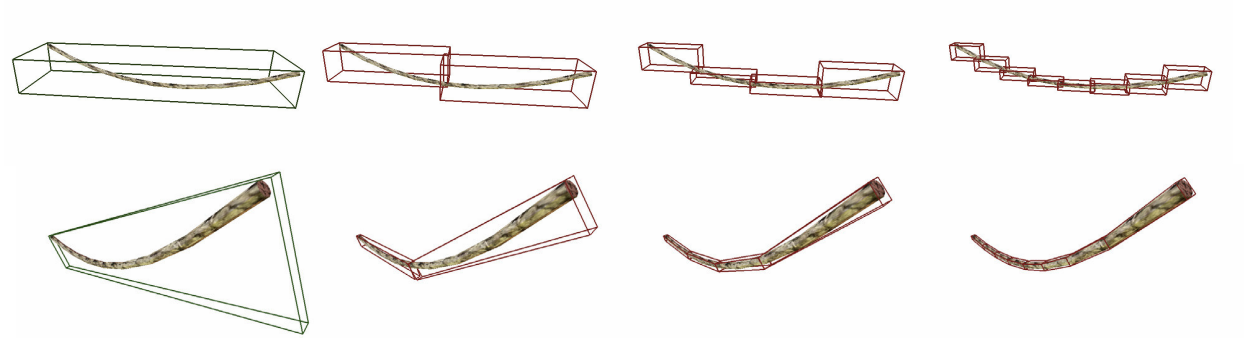


Figure 6: Using AABB (top row) and OBB (bottom row) to detect collision of simulated thread. The root and the leaf boxes are shown on the left and right column respectively.

focuses specifically on the study of collision detection techniques that are suitable for suture thread simulation. In particular, three kinds of BVHs – bounding spheres, AABBs and OBBs – are investigated and compared in order to evaluate their performance and efficiency. Experimental results suggest that the use of AABBs is a more appropriate approach for detecting the collision of objects modeled with a chain of straight line segments. Based on the finding on collision detection methods, a virtual-reality based suturing training system is being developed using a pair of haptic devices as user interface. While the motivation of the study lies on the simulation of sutures, the finding is applicable to one-dimensional elastic objects in general.

Acknowledgement

This work was supported in part by the Research Grants Council of the HKSAR (No. PolyU 5147/06E and PolyU 5145/05E). The authors would like to thank Dr. K. Sylvain and Dr. A.W. Siu for their support to the projects.

References

[1] J. Brown, J.-C. Latombe, and K. Montgomery, "Real-time knot-tying simulation," *The Visual Computer*, vol. 20, pp. 165-179, 2004.

[2] P. Marshall, S. Payandeh, and J. Dill, "Suturing for surface meshes," presented at Proceedings of 2005 IEEE Conference on Control Applications, 2005, CCA 2005, pp. 31-36, 2005.

[3] J. Phillips, A. Ladd, and L. E. Kavraki, "Simulated knot tying," presented at IEEE International Conference on Robotics and Automation 2002, ICRA '02, pp. 841-846 vol.1, 2002.

[4] F. Wang, E. Burdet, A. Dhanik, T. Poston, and C. L. Teo, "Dynamic thread for real-time knot-tying," presented at First Joint Eurohaptics Conference, 2005 and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2005, pp. 507-508, 2005.

[5] F. Wang, E. Burdet, R. Vuillemin, and H. Bleuler, "Knot-tying with Visual and Force Feedback for VR Laparoscopic Training," presented at 27th Annual International Conference of the Engineering in Medicine and Biology Society, 2005, IEEE-EMBS 2005. , pp. 5778-5781, 2005.

[6] D. K. Pai, "STRANDS: Interactive Simulation of Thin Solids using Cosserat Models," *Computer Graphics Forum*, vol. 21, pp. 347-352, 2002.

[7] J. Spillmann and M. Teschner, "CORDE: Cosserat Rod Elements for the Dynamic Simulation of One-Dimensional Elastic Objects," presented at Proc. ACM SIGGRAPH, pp. 209-217, 2007.

[8] J. Lenoir, P. Meseure, L. Grisoni, and C. Chaillou, "Surgical Thread Simulation," presented at Proceedings of ESAIM'02, Modelling and Simulation for Computer-aided Medicine and Surgery, pp. 102-107, 2002.

[9] J. Bender and A. Schmitt, "Constraint-based collision and contact handling using impulses," presented at Proceedings of the 19th international conference on computer animation and social agents, pp. 3-11, 2006.

[10] T. Larsson and T. Akenine-Möller, "A dynamic bounding volume hierarchy for generalized collision detection," *Computers & Graphics*, vol. 30, pp. 450-459, 2006.

[11] G. v. d. Bergen, "Efficient collision detection of complex deformable models using AABB trees," *J. Graph. Tools*, vol. 2, pp. 1-13, 1997.

[12] S. Gottschalk, M. C. Lin, and D. Manocha, "OBBTree: A Hierarchical Structure for Rapid Interference Detection," *Computer Graphics*, vol. 30, pp. 171-180, 1996.