Efficient Collision Detection between Rigid and Deformable Objects for Simulation and Execution of Telerobotic Process

Christophe Caby, André Crosnier
LIRMM – UMR CNRS / UM II
161, rue Ada
34392 Montpellier Cedex 5, France
{caby, crosnier}@lirmm.fr

Abstract

A solution to the problem of real time collision detection during simulation and execution of telerobotic process is suggested in this paper. The solution allows to deal with dynamics and prediction of collisions between rigid and deformable polyhedral objects. The method finds its essence in combining graphics hardware capabilities and simplified geometric representation (Oriented Bounding Box) in order to precisely extrapolate the swept volume generated by the tool during its motion and to dynamically estimate the collisions between the tool and the environment. During virtual or real teleoperation, collision detection aims at providing the operator with real time sensorial information feedbacks that improve immersion in the virtual scene.

1. Introduction

This paper deals with a real time dynamic predictive collision detection between rigid and deformable polyhedral objects for telerobotic process simulation and execution. Telerobotic process requires to provide the operator with a realistic virtual environment for preparing, simulating and executing his task [1]. Within this environment, the operator must have facilities to interact in a real way with virtual bodies so as to improve his feeling of telepresence [2]. As we consider tasks that include contact phases between the robot and objects of the workspace and deformations, simulation and modeling of physical phenomena resulting from these interactions play a crucial role. They principally contribute to the synthesis of sensorial information feedbacks (visual, sound, tactile) which increase the operator immersion. An interaction model, able to take into account physical phenomena associated with geometric interference, is being established. Collision detection is the first step of interaction model. The other two stages are computation of the force collision, based on the interpenetration distance between the bodies, and of the deformations of the elastic object submitted to this force [1].

The problem of collision detection between two objects is one among the fundamental topics in dynamic simulation and robotics [3]. Collision detection aims at automatically reporting a geometric interference between two or more objects in a virtual scene, before or when the contact occurs. However collision detection is considered as the major bottleneck of real time dynamic simulations. Studies have shown that 90% of the computation time of a deformable bodies simulation is spent by the collision detection routines [4]. This paper only focuses on collision detection that has already motivated a number of studies in different areas. The next subsection underlines other works with efficient techniques.

1.1. Previous works

Basic collision detection method performs static interference tests at discrete time instants near enough for collisions not to be missed [5]. Interference among polyhedral objects is detected by testing all combinations of face and edge couples. The average time complexity for such a test can be computationally expensive as the objects complexity increases. Thus most efficient methods first perform an acceleration of the detection and then a fine detection [6].

The aim of collision detection acceleration techniques is to eliminate couples of objects that could not be in contact. They consist of possible non contacts detection, and allow the acceleration of the collision detection by performing rejection tests of geometric entities. Three kinds of techniques [4] are generally used: bounding volumes hierarchies, spatial subdivisions and geometric coherence.

Bounding volumes hierarchies consist in approximating objects with various resolution levels of bounding volumes. These volumes can either be boxes [4][7][8] or spheres [9]. Their advantages in collision detection are the use of simplified geometric representations instead of the original objects and the definition of various resolution levels to fit the object geometry. In [10], another strategy based on bounding volumes is described. It consists in adding a temporal dimension in order to create a space time bound structure which allows the estimation of the nearest collision time. These methods are particularly well fitted for rigid bodies which have a fixed topology. However when applied to deformable objects, the update

of bounding volumes hierarchies at each time step can be time consuming.

The spatial subdivisions are computed thanks to regular divisions, voxels [11] or octrees [5][6] techniques. Two volumes located in different zones cannot interpenetrate each other. These methods present two limitations: the course of the subdivisions set and the accuracy of the subdivision hardly adaptable to the geometry of deformable objects.

Acceleration techniques using geometric coherence exploit the information of the last time step for solving the current collision detection problem. In [12][13], authors consider that geometric relations between objects do not change in a considerable way between two time steps.

The acceleration phase must be followed by a fine collision detection which aims at finding the accurate parameters of the geometric contact. Most of past works are based on the distance computation between two convex polygonal objects [13][14].

The GJK algorithm [14][15] is an iterative method of distance computation between convex hull of two points sets. Its goal is to calculate the Euclidean distance between two objects from the nearest points of their convex hulls. The algorithm returns either this distance when the hulls are separated, or an approximation of the negative distance.

The Lin-Canny algorithm [13] consists in finding the two nearest characteristics (faces, edges or vertices) between two moving convex polyhedra. It relies on the fundamental concept of Voronoï regions. Given these two characteristics, the distance between two polyhedra is computed with simple geometric formulas. The Lin-Canny algorithm gives the nearest points of two rigid convex polyhedra in linear time when they are separated.

The above algorithms can be extended to cater for non-convex polytopes by using hierarchies of convex components [16]. Although this technique works well for slightly non-convex objects, it becomes very inefficient as the level of non-convexity increases [6]. To overcome this drawback, implicit surfaces [17][18] are commonly used to model deformable objects. Their advantages are to find and maintain the precise contact zone over time. However polygonal approximation methods are still too slow for real time performances.

1.2. Problem requirements

The focus is principally put on telerobotic process involving a rigid tool and static deformable workpieces. For example, in drilling operation, the operator manipulates in a three dimensional environment the tool in order to perform his task. This operation first consists in positioning the tool extremity upon the workpiece and then in applying a normal force to the surface. Movements of the tool are non predetermined and can be both rotation and translation. In such simulation, requirements for collision detection have not yet been all met by any of the methods surveyed in last subsection. These requirements are the ability to achieve real time performances, to handle deformable and concave objects, and to perform dynamic and predictive detection.

A system runs in real time if it is able to absorb all input information while they have interest, and to react quickly enough so that this reaction makes sense. This definition is right but not sufficient because in dynamic simulation, real time performances are often related to the operator immersion. Indeed depending on the information feedback, the operator does not have to feel discontinuities in the related sense. Consequently our collision detection method must provide at least a 15 frames per second rate to allow a visual real time simulation.

In robotic process simulation, it is very important to be able to perform collision detection for deformable objects that can either be convex or concave. As the association of auxiliary data structures (bounding boxes, octrees, etc.) for each object requires computationally expensive updates, our collision detection technique does not achieve any special processing of deformable objects.

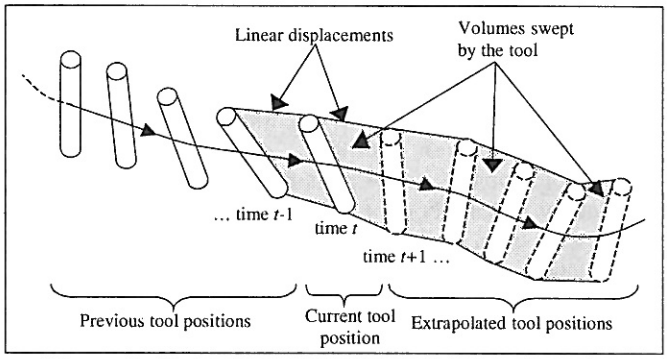

Figure 1. *Dynamic and predictive collision detection.*

It is common to distinguish between three kinds of collision detection: static, pseudo-dynamic and dynamic [19]. The first two can lead to critical situations when geometric interference are not detected because they consider the objects only at discrete times. The method introduced in this paper takes into account the volume swept by the whole tool (not only its extremity) between two consecutive instants to achieve a dynamic detection (cf. Figure 1). The assumption that the tool follows a straight line displacement between two positions is made. Indeed most of teleoperation applications require a limited displacement speed of the tool for safety reasons. The displacements undergone by the tool between two successive times are thus short length and defines small swept volumes, considering that system sample frequency is high.

Another characteristic of this method is its prediction aspect. The aim of prediction is not related to the computation of the nearest collision time [10] but to the operator response time – 1Hz for considered acts and 10Hz for reflex actions. Indeed, when a collision occurs, the operator must be able to react quickly and in a good way. Thus a collision which happens at time t must be predicted so that the operator have assimilated this information at time t. Figure 2 illustrates this principle. Suppose that the system sample time is smaller than the operator response time Δt and that a collision occurs at time t. Without a predictive detection, the operator is informed of the collision at time t and reacts at time $t+\Delta t$ while the system is running during this interval. In the other case, the collision is predicted at time $t-\Delta t$ so that the operator reacts to the collision when it happens (cf. Figure 2.). The predictive collision detection consists in achieving several tests with extrapolated tool positions at each discrete time instant (cf. Figure 1).

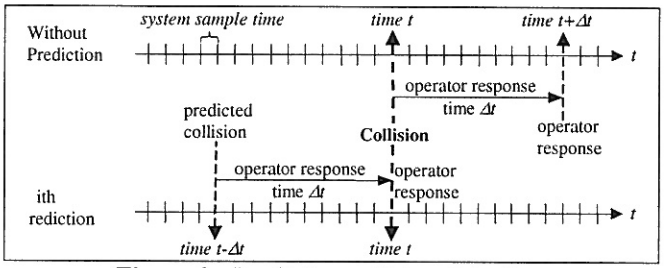

Figure 2. *Predictive collision detection.*

The remainder of the paper is organized as follows. Section 2 focuses on the OpenGL functions which allow the development of the method. Sections 3 describes the main features of the collision detection technique. Section 4 both presents simulation results and method's validation.

2. A graphics library: OpenGL

OpenGL (Open Graphics Library) [20] is a software interface to graphics hardware. This interface consists of about 120 distinct commands which allow the specification of objects and operations, needed to produce interactive three dimensional applications. The method takes benefit of its graphics capabilities by associating the visualization process with the collision detection [21].

Besides the objects positioning and orientation in the virtual scene, OpenGL offers facilities to define a viewing volume (objects outside are clipped out) and to specify the way objects are projected on the screen. There are two kinds of projection: perspective and orthographic (cf. Figure 3). The perspective projection is similar to our vision mode: the further an object is, the smaller it appears, and two parallel straight lines seem to converge in the distance. The orthographic projection draws object without affecting their relative size. In both cases, viewing volumes (in gray on Figure 3) are hexahedra, respectively a truncated pyramid and a box, defined by six parameters. The user can also specify up to six additional clipping planes to further restrict the viewing volume, and define the camera position and orientation thanks to the function *gluLookAt()*.

While most graphics software are content with drawing static objects in a simple way, OpenGL allows the user to identify objects that appear on the screen. This procedure cannot be easily performed because of the multiple three dimensional transformations undergone by the objects. OpenGL provides two mechanisms for identifying the objects drawn in a specified region of the scene. These two modes, select and feedback, that do not necessarily require to update the display, substitute for the usual rendering mode, *GL_RENDER*, thanks to the function *glRenderMode()*, which takes the arguments *GL_SELECT* and *GL_FEEDBACK*. Typically, when using the select mechanism and displaying the scene, OpenGL returns the list of primitives that are located in the viewing volume (function *glSelectBuffer()*), do well as their number (function *glRenderMode()*).

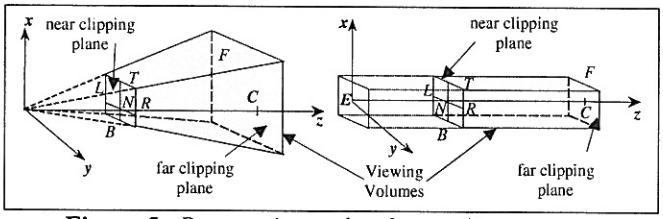

Figure 3. *Perspective and orthographic cameras.*

3. Collision detection

3.1. Dynamic detection

The main feature of the approach is to perform dynamic rather than static collision detection. The volume swept by the whole tool between two instants is taken into account instead of the tool shape at discrete time instant. The virtual tool manipulated by the operator is considered as a rigid cylinder undergoing non predetermined motions. Its positions and orientations at discretized times are the only known data.

The basic idea of this technique consists in associating the test of geometric interference with the OpenGL visualization process. At each time step the mapping is established between the viewing volume and the volume swept by the tool between two instants. On the one hand, if the OpenGL select mechanism does not detect the presence of objects, no contact is said to have occurred. On the other hand, if part of the deformable object is present in the viewing volume, a collision is detected between the tool and this object. The main mapping constraint is to minimize differences between the two volumes in order to avoid unlikely collisions detection. The solution chosen consists in computing a tight

bounding box (Oriented Bounding Box or OBB) of the volume swept by the tool between two consecutive positions at times t-1 and t. Then an orthographic camera (cf. Figure 3) is associated with the parallelepipedic OBB, and so to the swept volume.

The principle of OBB is to approximate an object, considered as a collection of polygons, with a box of similar dimensions and orientation. The OBB computation algorithm presented in [7] makes use of first and second order statistics summarizing the vertex coordinates. They are the mean, μ, and the covariance matrix, C. If the vertices of the i'th triangle are the points p^i, q^i and r^i defined in the reference frame, then the mean and covariance matrix can be expressed in vector notation as:

$$\mu = \frac{1}{3n}\sum_{i=0}^{n}(p^i + q^i + r^i)$$

$$C_{jk} = \frac{1}{3n}\sum_{i=0}^{n}(\bar{p}_j{}^i \bar{p}_k{}^i + \bar{q}_j{}^i \bar{q}_k{}^i + \bar{r}_j{}^i \bar{r}_k{}^i), \quad 1 \le j,k \le 3$$

where n is the number of triangles, and $\bar{p}^i$, $\bar{q}^i$, $\bar{r}^i$ are the vectors defined by $\bar{p}^i = p^i - \mu$, $\bar{q}^i = q^i - \mu$ and $\bar{r}^i = r^i - \mu$. C_{jk} are the elements of the 3×3 covariance matrix.

The eigenvectors of the C symmetric matrix are mutually orthogonal, and after normalizing them, they are use as the basis of the OBB. The extreme vertices along each axis of this basis define the size of the bounding box.

Considering that the collection of polygons consists of the triangles that define the tool at times t-1 and t, the minimal box bounding the volume swept by the tool is built. The OBB basis vectors orientation are turned so that axis Z corresponds to its length, axis Y with its width and axis X with its height (cf. Figure 4). This is an arbitrary choice which allows a homogeneous reasoning. Then the reference frame of the camera is mapped on the OBB basis vectors.

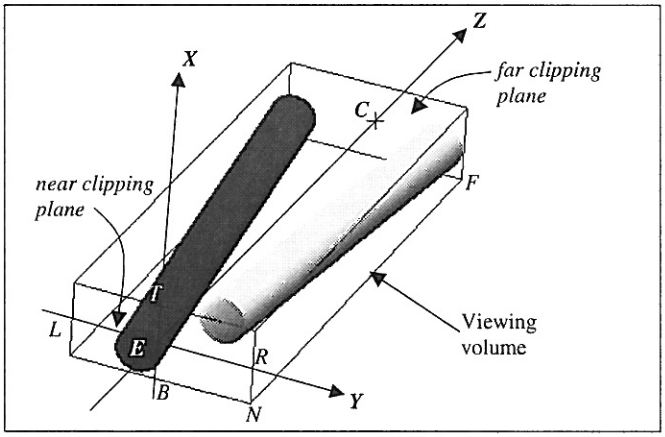

Figure 4. *OBB representation and camera specification.*

The setting of the OBB enables to define the viewing volume dimensions thanks to the function *glOrtho()*. The camera is positioned at point E and its direction is defined by the vector EC thanks to the function *gluLookAt()*. However Figure 4 shows that the volumic approximation carried out using the OBB is not tight enough to approximate the swept volume. It can be improved by using extra clipping planes.

These planes restrict the viewing volume by refining the convex hull defined by the OBB. The determination of the clipping planes is based on a geometrical reasoning into the plane (Y,Z) (cf. Figure 5). This plane is defined by the two greatest directions of the OBB which are usually associated with the largest differences of the volumes mapping. Four additional clipping planes can be defined in this plane: two side and two frontal.

Both side clipping planes (SCP1 and SCP2) are defined by two points (Y_{max3} and Y_{max4} or Y_{min1} and Y_{min2}) and by the X axis of the camera frame. These points are obtained after the computation along the Y axis of the two min-max values of the vertices lying into each face of the tools (rear and front) (cf. Figure 5).

To compute the two frontal clipping planes (FCP1 and FCP2), the following steps are taken. For FCP2, the two front faces of the tools are considered. For each front face, the min-max values are computed along the Y axis, and two points are obtained for each face: (Y_{max1}, Y_{min1}) and (Y_{max3}, Y_{min3}). Then the aim is to find the straight line which contains two points (Y_{max1} and Y_{min3} on Figure 5), and which satisfies the following property: the two other points belong to the left (resp. right for FCP1) halfspace created by the straight line. This straight line associated with the X axis of the camera frame defines a frontal clipping plane.

Figure 5 shows that the use of clipping planes leads to a new tight viewing volume (hatched zone).

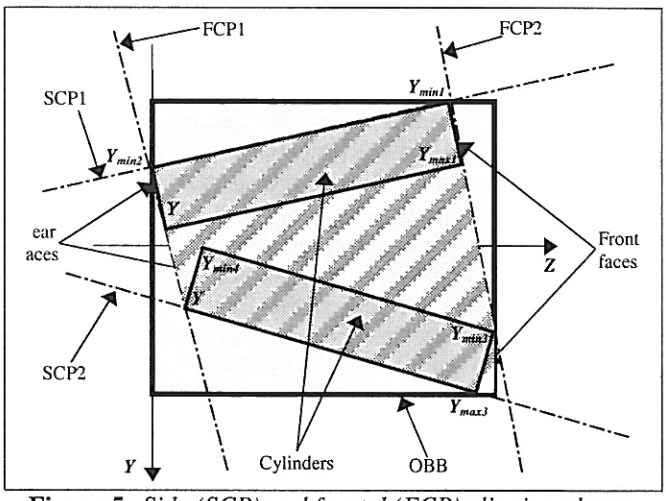

Figure 5. *Side (SCP) and frontal (FCP) clipping planes.*

The collision detection algorithm returns the number of primitives located in the specified viewing volume thanks to the select mechanism of OpenGL. If this number is equal to zero, there is no collision. Otherwise, there is a geometric interference and the triangles of the deformable object in contact with the tool can be identified.

3.2. Predictive detection

Collision detection's prediction aims at reporting to the operator a collision (warning light, highlighted contact zone) before its occurs. To take into account his response time, the operator must be informed sufficiently early to have time to assimilate the future situation and to consequently modify the tool's behavior. Collisions do not have to be processed when they are predicted but when the contact really happens.

The method consists in computing extrapolated tool's position-orientation couples at each discrete time instant; these couples are then used to perform dynamic interference tests. Studied system is composed of 6 state variables q_i associated with the 6 degrees of freedom of the tool (i.e. 3 translations and 3 rotations). Although some variables can be constrained by the process, they are considered mutually independent. The computation of extrapolated tool's position-orientation couples relies on two main steps: interpolation of the movement and determination of tool's estimated future configurations.

The first step consists in interpolating each of the non-constrained state variables with a polynomial function defined by the operator. Identification of the function parameters is performed thanks to a recursive least squares method on an observation period composed of tool's position-orientation samples.

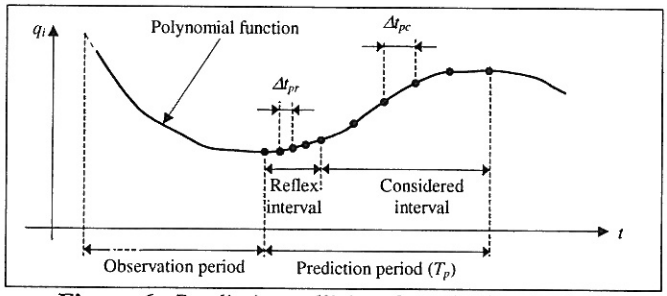

Figure 6. *Predictive collision detection's principle.*

Prediction is then performed computing future tool's configurations with the interpolated movement in the prediction's temporal interval T_p. During prediction period, the tool's behavior does not have to be modified; thus prediction's temporal interval must be smaller than operator's response time for considered acts (i.e. 1 sec). Furthermore, the system must inform the operator of a predicted collision so that he have assimilated the information when the collision occurs. Thus T_p is set equal to 1 sec. The prediction period is divided in two interval: the reflex interval and the considered interval (cf. Figure 6). The reflex interval corresponds to the operator's reflex response time (i.e. 0.1 sec), and the considered interval matches to the difference between this last interval and operator's considered response time (i.e. 0.1 sec to 1 sec). Indeed, the temporal interval between two predictions relies on the degree of interest puts on the current prediction period. Information related to the reflex interval takes an important part because it deals with what happens very soon and because it is associated with a reflex act. Thus prediction's sample time step Δt_{pr} associated with the reflex interval has to be small in order to provide the operator with numerous information. For the considered interval, the prediction's sample time step Δt_{pc} is greater because operator needs less information and has more time to react. Finally, prediction's samples number is limited by the computation time of prediction and collision detection. Predictive collision detection does not have to reduce the simulation system's sample frequency under 15Hz (i.e. visual real time).

4. Results and validation

A series of tests to bench the collision detection method have been performed. The experimental platform is a 866 MHz Pentium III workstation running under Windows NT and using an ELSA Gloria II Pro graphics card. The programming environment is developed with Microsoft Visual C++. The operator manipulates a cylindrical tool thanks to a 6 d.o.f. Space Mouse in a 3D virtual workspace composed of a deformable hull (cf. Figure 7).

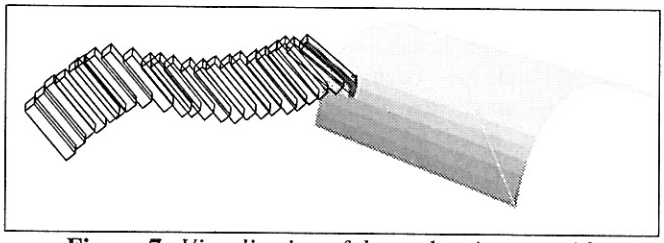

Figure 7. *Visualization of the tool trajectory with Oriented Bounding Boxes.*

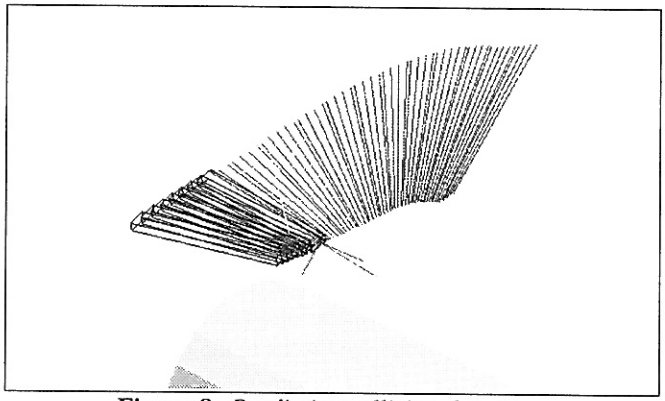

Figure 8. *Predictive collision detection.*

Table 1 shows results of simulations without prediction done with various geometric resolutions of the deformable objects, using only OBB or both OBB and clipping planes.

	400 triangles	1600 triangles
OBB + clipping planes	0.85 ms	2.21 ms
OBB alone	0.61 ms	1.48 ms

Table 1. *Computation times for collision detection.*

The collision detection method provides impressive results. Use of extra clipping planes damages the efficiency of the algorithm in a quite reasonable way.

Thanks to these efficient computation times, high sampling frequency of the operator inputs is defined. Consequently both mapping of the viewing volume with the volume swept by the tool and assumption of straight line displacements of the tool between two consecutive instants are validated.

Efficient computation times allow the realization of predictive collision detection. Several interference tests can be performed at each discrete time instant with extrapolated tool's configuration (cf. Figure 8).

5. Conclusion and future works

This paper has presented an efficient collision detection between rigid and deformable objects for a drilling telerobotic process. This technique is based on OpenGL graphics hardware and on oriented bounding boxes to perform a dynamic detection taking into account the volume swept by the tool between two successive times. Real time performances lead to achieve a predictive collision detection with extrapolated tool's position and orientation with respect to the operator response times. The method has been validated through simulations providing good real time performances. Unlike other techniques, since no pre-computation is required this method ideally fits dynamic simulation of elastic bodies.

This method can be extended to handle other telerobotic processes characterized by more complex tool shape. The technique consists in associating a hierarchy of oriented bounding boxes (i.e. OBBTree [7]) with the volume swept by the tool between two consecutive times. The collision detection method can also be used as an acceleration phase before a fine collision detection. The technique described in this paper can also be applied to the crucial issue of swept volume computation (cf. Figure 7).

The next step of our work is the collision processing which consists in computing the force collision thanks to the interpenetration volume between the objects. This force is then used to determinate the deformations of the elastic object according to the bar model described in [1]. Soon an haptic device will be interfaced with the system in order to provide force feedbacks to the operator.

References

[1] C. Caby *et al.*, "Interaction Model for Programming Teleoperated Missions", 31st International Symposium on Robotics 2000, pp. 214-219, Montréal, Canada, May 14-17 2000.

[2] J. Vertut *et al.*, "Téléopération : Evolution des Technologies", Tome 3a-3b, Editions Hermès, Paris, France, 1984, in French.

[3] M.C. Lin *et al.*, "Collision Detection Between Geometric Models: A Survey", IMA Conference on Mathematics of Surfaces, Birmingham, Great Britain, August-September 1998.

[4] P. Meseure *et al.*, "Accélération de la Détection de Collisions entre Corps Rigides et Déformables", Actes des 6èmes Journées du Groupe de Travail Réalité Virtuelle, PRC-GDR Communication Homme-Machine, pp. 167-174, Paris, France, March 12-13 1998, in French.

[5] A. Smith *et al.*, "A simple and Efficient Method for Accurate Collision Detection Among Deformable Polyhedral Objects in Arbitrary Motion", IEEE Virtual Reality Annual International Symposium, pp. 136-145, March 1995.

[6] F. Ganovelli *et al.*, "BucketTree: Improving Collision Detection Between Deformable Objects", Spring Conference on Computer Graphics, Bratislava, Slovakia, May 4-6 2000.

[7] S. Gottschalk *et al.*, "OBBTree: a Hierarchical Structure for Rapid Interference Detection", SIGGRAPH'96, Computer Graphics, pp. 171-180, New-Orleans, USA, August 4-9 1996.

[8] A. Crosnier *et al.*, "Tribox Bounds of Three Dimensional Objects", Computer and Graphics, Vol. 23, pp. 429-437, 1999.

[9] S. Quinlan, "Efficient Distance Computation between Non-Convex Objects", IEEE International Conference on Robotics and Automation 1994, pp. 3324-3329, San Diego, USA, April 1994.

[10] P.M. Hubbard, "Collision Detection for Interactive Graphics Applications", PhD Thesis, Brown University, Providence, USA, October 1994.

[11] A. Garcia-Alonso *et al.*, "Solving the Collision Detection Problem", IEEE Computer Graphics and Applications, Vol. 14, No. 3, pp. 36-43, May 1994.

[12] D. Baraff, "Curved Surfaces and Coherence for Non-penetrating Rigid Body Simulation", SIGGRAPH'90, Computer Graphics, Vol. 24, pp. 19-28, Dallas, USA, August 1990.

[13] M.C. Lin, "Efficient Collision Detection for Animation and Robotics", PhD Thesis, Department of Electrical Engineering and Computer Science, University of California, Berkeley, USA, 1993.

[14] E.G. Gilbert *et al.*, "A Fast Procedure for Computing the Distance between Complex Objects in Three-Dimensional Space", IEEE International Conference on Robotics and Automation, Vol. 4, pp. 193-203, 1988.

[15] S. Cameron, "Enhancing GJK: Computing Minimum and Penetration Distances between Convex Polyhedra", IEEE International Conference on Robotics and Automation, pp. 3112-3117, Albuquerque, USA, April 22-24 1997.

[16] D. Baraff *et al.*, "Dynamic Simulation of Non-penetrating Flexible Bodies", SIGGRAPH'92, Computer Graphics, Vol. 26, pp. 303-308, Chicago, USA, July 1992.

[17] A. Pentland *et al.*, "Good Vibrations: Modal Dynamics for Graphics and Animation", Computer Graphics, Vol. 23, No. 3, pp. 215-222, July 1989.

[18] M.P. Gascuel, "An Implicit Formulation for Precise Contact Modelling Between Flexible Solids", SIGGRAPH'93, Computer Graphics, pp. 313-320, Anaheim, USA, 1993.

[19] M. Held *et al.*, "Evaluation of Collision Detection Methods for Virtual Reality Fly-throughs", 7th Canadian Conference Computational Geometry, pp. 205-210, Quebec, Canada, August 10-13 1995.

[20] J. Naider *et al.*, "OpenGL Programming Guide", Addison-Wesley, 1993.

[21] J.C. Lombardo *et al.*, "Real-time Collision Detection for Virtual Surgery", Computer Animation'99, pp. 33-39, Geneva, Switzerland, May 26-28 1999.