INSTITUCIÓN UNIVERSITARIA SALAZAR Y HERRERA
Una decisión que se nota

Quid, N°. 19, pp. 29-34, Jul-Dic, 2012, ISSN: 1692-343X, Medellín-Colombia

DEVELOPMENT OF A HIERARCHY COLLISION DETECTION ALGORITHM IN ORDER TO IMPLEMENT A LAPAROSCOPIC SURGICAL SIMULATOR

DESARROLLO DE UN ALGÓRITMO DE DETECCIÓN DE COLISIONES JERÁRQUICA PARA LA IMPLEMENTACIÓN DE UN SIMULADOR DE CIRUGÍA LAPAROSCÓPICA

Est. Juan Sebastián Muñoz
Universidad EAFIT,
Laboratorio de Realidad Virtual.
Carrera 49 N° 7 Sur — 50,
Medellín, Colombia

PhD. (c) Christian Andrés Diaz León,
Universidad EAFIT,
Laboratorio de Realidad Virtual.
Carrera 49 N° 7 Sur — 50,
Medellín, Colombia
cdiazleo@eafit.edu.co

PhD. Helmuth Trefftz Gómez,
Universidad EAFIT,
Laboratorio de Realidad Virtual.
Carrera 49 N° 7 Sur — 50,
Medellín, Colombia
htrefftz@eafit.edu.co

(Recibido el 12-05-2012. Aprobado el 20-06-2012)

Resumen: el sistema de detección de colisión de un simulador quirúrgico es una de las partes más críticas para ser desarrolladas, debido a que encontrar cuál sección de un órgano choca con un instrumento laparoscópico tiene que hacerse en tiempo real y con la mayor precisión posible. En este trabajo se utiliza una aproximación denominada "más cerca de los dos triángulos más lejanos", puesto que con ello, no existe la preocupación por el problema de delimitación y superposición en la aproximación de los cientos de objetos de delimitación (esferas, cajas, etc.).

Palabras claves: simulador quirúrgico, detección de colisión, aproximación de jerarquías.

Abstract: the collision detection system of a Surgical Simulator is one of the most critical parts to be developed, because finding which section of an organ collides with the laparoscopic instrument has to be done in real time and as accurately as possible. In this paper we use a "most near from the two farthest triangles" approximation because we do not have to worry about the bounding-overlap problem in the bounding objects (spheres, boxes, etc.) approximation.

Keywords: surgical simulation, collision detection, bounding hierarchies.

1. INTRODUCCIÓN

In a surgical simulation an important component of the system is the collision detection algorithm, it provides the surgeon the interactivity with the virtual environment, allowing him/her to execute the common tasks of surgical simulation. Typical applications in computer graphics normally detect collisions between rigid bodies (concave and convex), but surgical simulators impose somewhat different requirements. Firstly, the objects (organs and tissues) are deformable. Secondly, the number of polygons of each object is very high in order to achieve acceptable realism. Hence, very efficient algorithms are required.

In such an application, the following interactions of pairs of objects that need testing can be found:

- Between surgical tools (Rigid objects).
- Between a surgical tool and a virtual organ (Rigid and deformable objects).
- Between virtual organs (Deformable objects).

The first phase is a broad phase where the idea is to select pairs of objects which are probably colliding, this can be handled using spatial decomposition algorithms. In this area several approaches aiming at decomposing the space have been proposed implementing voxels (García, et al., 1994) and (Dickheiser, 2000), binary space partition trees (Thibault, W. C. and Naylor, 1987) and sweep and prune approach (Cohen et al., 1995).

The second phase is a narrowing phase. In this phase, the exact colliding point between two objects is not computed, but broader areas of collision, which are called the "zone of collision". In this area several works have been proposed depending of which object is going to be evaluated. For objects with convexed geometry, algorithms like GJK (Gilbert et ál, 1999) LC (Lin and Canny, 1991) or EGJK (Cameron, 1997) are the most used. These algorithms determine the distance between two objects; this distance can be negative or positive depending if the objects are or not overlapped, respectively. On the other side, when objects are concave, the idea is to divide the object into convex sub objects and then to apply these algorithms. However, on deformable objects like the ones present in surgical simulators, the convex division of the object has to be updated and this can usually not be done in an iterative rate. Finally, in deformable objects, the best way to proceed is the bounding volumes Hierarchies that can be spheres (Benitez et ál, 2005), AABBs (Zhang and Kim, 2007) or OBBs (Gottschalk, 1996). These hierarchies offer a fast test finding the collision zone that can be easily updated. The problem is that these hierarchies can have some precision issues finding the correct fitting of the volumes to the actual object geometry.

Finally, on the third phase, the exact point of collision is determined and the entities colliding are determined. In this phase very simple collision tests are applied (like sphere-sphere, triangle-triangle and box-box test). In the above mentioned approaches, the one described in this paper is the one that detects collision in the presence of deformable objects; in specific, the narrowing phase. One of the best known techniques for collision detection in deformable objects is the bounding-spheres techniques, which consists in wrapping the parts that compose the object to be collided with spheres, because of the invariance of the spheres in the presence of rotations, it makes searching a fast and simple process. The problem is that the spheres that wrap the triangles are overlapped which make that searches can fail. Bounding spheres can be changed to bounding boxes, but the box-box test is more complex than the sphere-sphere test. It is because of this that we propose an algorithm, which uses a hierarchy. For this reason, the algorithm is fast. It does not use bounding volumes, instead uses a "most near from the two farthest triangles" approach to build the hierarchy.

The rest of the paper is organized as follows: Section 2 describes similar projects. Section 3 gives an overview of the proposed collision detection Algorithm, the hierarchy creation, searching in the hierarchy and the experimental setup. Section 4 present the obtained results and finally, section 5 describes the conclusions reached so far and the future work.

2. RELATED WORKS

Several algorithms have been developed to detect collisions in a surgical environment, but the most widely used method to detect collision between surgical instruments and organs (this interaction is composed for a rigid and deformable object) is the bounding volume hierarchy. It is commonly used because the hierarchy allows an easy update of the data structure. This is the key point, because the features of the deformable objects change their position during collisions.

Mainly three different bounding volumes to build hierarchies have been used (Fares and Hamman, 2005). The most simple are the spheres-bounding volumes, but fitting the sphere to the object geometry is very poor and therefore the precision is not very good and in some cases the search can diverge if the hierarchy has not been built correctly. On the other hand, speed is very high because the sphere-sphere test has a low complexity. In (Hubbard, 1996) we can find a description of the building and searching of sphere hierarchies. On the other hand, the bounding box hierarchies can be of two different ways, using AABB (Zhang and Kim, 2007) or OBB (Gottschalk, 1996). When we use a bounding box hierarchy, two important decisions have to be made. The first is regarding the orientation of the box, because it provides a good fitting and the second is how to divide the nodes. The OBBs are normally oriented using the statistical principal components as the axis, on the other hand AABBs are oriented parallel to the local axis frame. The OBBs hierarchies provide a better fitting to the object geometry than the ABBs hierarchies, however the construction of the OBBs hierarchy is more complex than the ABBs hierarchies. In the Fig. 1 we can observe the adaptation achieve with everyone of the bounding volumes to a simple object.

In a surgical simulation, the objects (organs and tissues) are subject to continuous deformation inside the virtual environment. This implies that the hierarchies have to be updated in order to adapt the deformed object. In the literature two different techniques of modified hierarchies have been proposed:

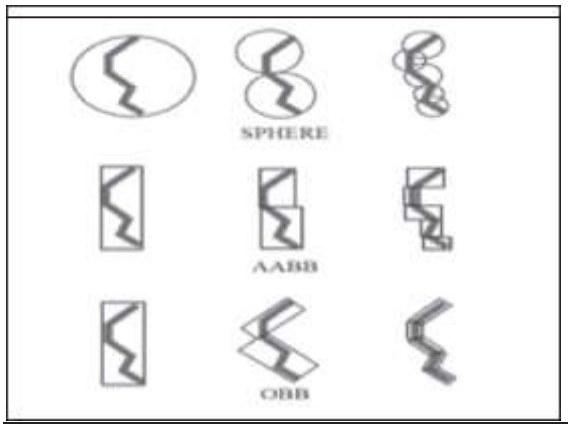

Fig. I. Comparison of the 2D urinary protein profile acetone and ethanol precipitation.

- Rebuild: This process is the entire reconstruction of the tree from parent to child (top-bottom) or child to parent (bottom-top).
- Refitting: This process only changes the affected bounding volumes. The changes are the propagated from node to node until root or leaf is reached.

Several research projects that involve bounding volumes, as mentioned before, have been made. Vand der Bergen (Bergen, 1997) proposed a method to update an AABBs hierarchy using a bottom-up refitting technique. On the other hand, Larsson and Moller (Larsson, and Akenine-Moller, 2001) enhanced the algorithm proposed by Van der Bergen in AABBs hierarchies. The basic idea is similar to Van Der Bergen's but instead of doing a bottom-up update sequence, an intermediate level in the hierarchy is chosen as the starting point for the update process. Hence, the levels up of the intermediate level were updated using the bottom-up strategy and the remaining levels were update using a top-town strategy. Finally, an approach to update sphere hierarchies was developed by Brown et al. in (Brown et ál, 2001) applied in a surgical simulation, where the radius and position of the sphere center taking the nodes that have been modified are updated. It uses a bottom-up strategy.

3. COLLISION DETECTION ALGORITHM

The algorithm that we propose in this paper is divided into two parts, the first one is the building process of the collision hierarchy, where a tree structure is built. Sets of tridimensional points are stored in non-leaf nodes of the tree, and tridimensional points that reference each triangle of the object (organ) are stored in the leaves of the trees.

On the other side, the second part consists on how to manipulate the hierarchy in order to find the triangle that can be colliding with the laparoscopic instrument.

3.1 Hierarchy creation

The minimum element that the object (organ) is made from is a triangle, but in order to achieve simplicity we do not use triangles to perform a search (that is three points), instead we use a central point that is the average of the sums of the vertex of the triangle, then we make a reference (In the programming context) for each 3d point calculated to each triangle (see Fig. 2).

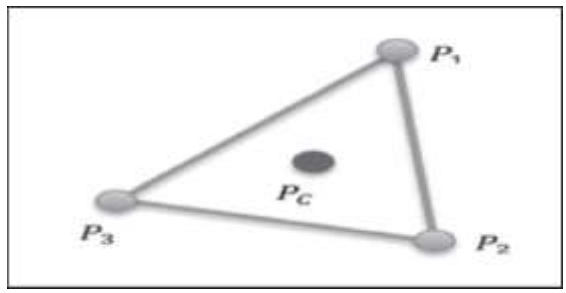

Fig. 2. Triangle with the vertex and the average point.

$$Pcenter = \frac{P_1 + P2 + P_3}{3} \quad (1)$$

Where P1, P2 and P3 are the vertices of the triangle. For the hierarchy building process we like this

- The set u_0 denotes the finite universe of valid triangles that compose the organ Ni = |Ui| is the size of a group of elements named |Ui| which is a subset of Uo
- *Tj* is the point j within *Ui*
- Ui_t and U_k two sub-groups from Ut where is father of Ui and U_k in which Ui u Uk = Ut and Ui n Uk = 0
- The function *d: X x X - R* denotes a measure of "distance" between two points (the smaller distance, the closer are the points).

The next algorithm describes how we build the hierarchy.

LoadHierarchy (Ud
If N_t > 1 then
Divide U_t in Uk and U_f
Does U_t father of Uk and U_1
LoadHierarchy (U_k)
LoadHierarchy (U_1)
End if
End LoadHierarchy

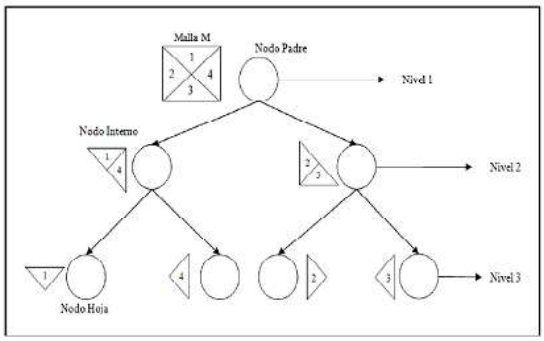

Fig. 3. In the figure a geometric mesh composed by four triangles, and the hierarchical structure associated with this, composed by three levels, and binary in nature is observed.

In this way to divide U_t in ***Uk*** and U_i the algorithm searches the two point Ck and C_i farthest between in U_t using this definition $d(C_k, c_i) = \max(d(U3)$ (2).

Where it is the index of group analyzed in that instant. Based on this criterion, we proceed as follow to define the content of the hierarchy:

For each Ci in U_t - [C_k,Cii do
1f $d(C_i , C_i) > d(C_k,$ then
Include Ti in Uk
Else If
Include T_i in U
End if
End For

3.2 Searching in the hierarchy

After building the hierarchy, we can search the triangles with a possible collision, applying this algorithm:

SearchCollision(11O, T)
Usearch = UO
While Nsearch > 1 do
If *d(T , Ci) > d(T , Ck)*
Usearch = Uk
Else if
Usearch = U1
End if
End While
Find the triangle T_r refer to Usearch
Find the triangle T_c belong to T
Return if there are collision between T_r and T
End SearchCollision

Where Usearch is the group of points where the search is happening and $\mathbf{T_r}$ is the triangle reference of the point T.

3.3 Experimental Setup

In order to evaluate the algorithm, we added it as a module of the surgical simulator, which it has been developing in Virtual Reality lab at EAFIT University. The complexity of the algorithm was determined in order to compare it with other hierarchical algorithms. On the other hand we developed an experimental test to determine the efficiency of the algorithm which was looking for the primitives (triangle, edge or point), that is closer to the object. In our experimental configuration, one of the objects (the rigid one), was changed to a triangle. For our virtual development we used a triangle that was at the end of the surgical instrument. We did not do any simplification with the deformable object, all its triangles were considered during the process of collision detection. Several meshes of different sizes, describing the deformable objects were created, and the size was determined by the number of edges of the mesh. In the measurement taken during the experiment, we took the time that is necessary for the algorithm to find the two closest primitives.

The tests were run on a DELL XPS computer, with a dual core processor of 2.0 GHz, 2 GB of RAM and Windows XP.

4. RESULTS

As was mentioned in the experimental configuration, was determined the time it took the algorithm to find the closest primitives. Figure 4, describes the results and the behavior of the algorithm as the size of the mesh was increased.

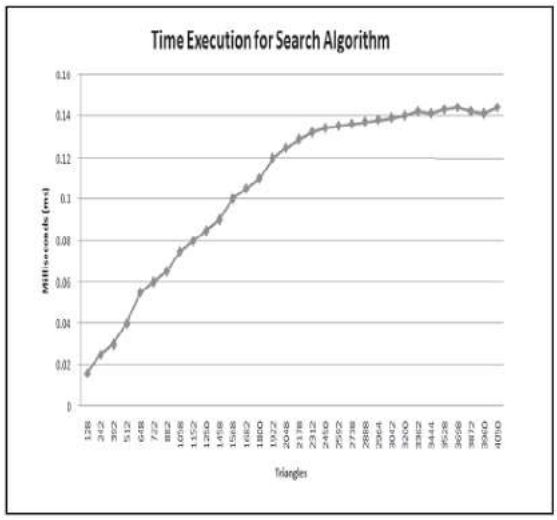

Fig. 4. Execution time with continuous translation between a triangle and a mesh to several size of mesh.

Table 1 shows the sizes of the meshes used and the time to find the two closest primitives.

Table 1. Features of the meshes used for the experimental test to evaluate the execution time of the search algorithm

	Triangles	Vertex	Edges
Mesh 1	128	81	207
Mesh 2	242	144	384
Mesh3	392	225	615
Mesh 4	512	289	799
Mesh 5	648	361	1007
Mesh 6	722	400	1120
Mesh 7	882	484	1364
Mesh 8	1058	576	1632
Mesh 9	1152	625	1775
Mesh 10	1250	676	1924
Mesh 11	1458	784	2240
Mesh 12	1568	841	2407
Mesh 13	1682	900	2580
Mesh 14	1800	961	2759
Mesh 15	1922	1024	2944
Mesh 16	2048	1089	3135
Mesh 17	2178	1156	3332
Mesh 18	2312	1225	3535
Mesh 19	2450	1296	3744
Mesh 20	2592	1369	3959
Mesh 21	2738	1444	4180
Mesh 22	2888	1521	4407
Mesh 23	2964	1560	6164

In Fig. 5 we can observe the visual representation of the algorithm implementation. Three models compose the surgical scene. The red one is a wireframe representation of the liver and blue and green ones are representations of the surgical instruments. The yellow and green triangles on the liver mesh are the nearest triangles to the tip of the blue and green surgical instruments, respectively.

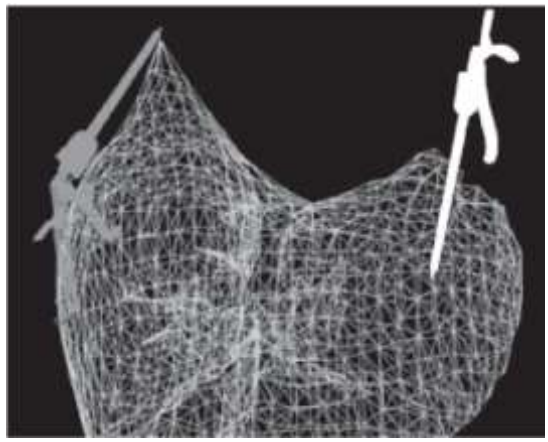

Fig. 5. Figure showing the detection algorithm proposed running with a wireframe representation of a liver model and two surgical instruments.

5. CONCLUSIONS

Current results suggest that, even as the size of the mesh is increased, the proposed algorithm shows a very stable behavior. We are currently working on tests with more complex objects that better resemble real anatomic structures.

We are also working on updating the hierarchy in order to determine collisions involving deformable objects as some organs and human tissue.

REFERENCES

Garcia, A., Serrano, N. and Flaquer, J. (1994) Solving the Collision Detection Problem. In IEEE Transactions on Computer Graphics and Applications, Volume **14,** Issue 3, pp. 36-43.

Dickheiser, M. (2000) Game Programming Gems 6, Cengage Learning.

Thibault, W. C. and Naylor, **B.** F. (1987) Set Operations on Polyhedra using Binary Space Partioning Trees. In Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, New York NY, USA, pp. 153-162.

Cohen, *J.* D., Lin, M. C., Manocha, **D.** and Ponamgi, M. (1995) I-COLLME: An Interactive and Exact Collision Detection System for Large-scale Environments. In Proceedings of the Symposium on Interactive 3D graphics, New York NY, USA, pp. 189-194.

Gilbert, E. G., Johnson, D.W. and Keerthi, S. (1999) A Fast Procedure for Computing the Distance between Complex Objects in Three-dimensional Space. In IEEE Transactions on Robotics and Automation, Volume 4, Issue 3, pp. 193 - 203.

Lin, M. C. and Canny, **J.F.** (1991) A Fast Algorithm for Incremental Distance Calculation. In Proceedings IEEE International Conference on Robotics ant Automation, Sacramento, CA, USA, pp. 1008-1014.

Cameron, S. (1997) Enhancing GJK: Computing Minimum and Penetration Distances between Convex Polyhedra. In Proceedings IEEE International Conference on Robotics and Automation, Albuquerque NM, USA, pp. 3112-3117.

Benitez, A., Del Carmen Ramirez, M. and Vallejo o, D. (2005) Collision Detection using Sphere-Tree Construction. In Proceedings of 15th International Conference on Electronics, Communications and Computers, Puebla, Mexico, pp. 286-291.

Zhang, X. and Kim, Y. (2007) Interactive Collision Detection for Deformable Models Using Streaming AABBs. In IEEE Transactions on Visualization and Computer Graphics, Volume 13, Issue 2, pp. 318-329.

Gottschalk, S., Lin, M. C. and Manocha, D. (1996) OBBTree: A Hierarchical Structure for Rapid Interference Detection. In Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, New York, NY, USA, pp. 171-180.

Fares, C. and Haman, Y. (2005) Collision Detection for Rigid Bodies: A State of the Art Review Proceedings of International Conference Graphicon 2005, Novosibirsk Akademgorodok, Russia, pp. 142-163.

Hubbard, P. M. (1996) Approximating Polyhedra with Spheres for Time-critical Collision Detection. In ACM Transactions on Graphics, Volume 15, Issue 3, pp. 179-210.

Bergen, G. (1997) Efficient Collision Detection of Complex Deformable Models usingAABB Trees. In Journal of Graphics Tools, Volume 2, Issue 4, pp. 1-14.

Larsson, T. and Akenine-Moller, T. (2001) Collision Detection for Continuously Deforming Bodies. In Eurographics 2001, Short Presentations, Manchester, England, pp. 325-333.

Brown, J., Sorking, S., Montgomery, K., Bruyns, C., Sephanides, M. and Latombe, J.C. (2001) Real-time simulation of deformable objects: tools and application Proceedings of the 14th Conference on Computer Animation, Seoul, South Korea, pp. 228-258.