Fast Hierarchical Methods to Detect Collisions between Deformable Objects: A Comparison

F. A. Madera A. M. Day S. D. Laycock

School of Computing Sciences, University of East Anglia, Norwich NR4 7TJ, UK

Abstract

The time complexity of a collision detection algorithm can be reduced to logarithmic in the number of tests performed when the objects are decomposed into a Bounding Volume (BV) tree hierarchy. It is well known that the Sphere Bounding Volume (SBV) and the Axis Aligned Bounding Box (AABB) hierarchies require inexpensive overlapping tests. However, we present a method called the Distance Hierarchy (DH) which is more suitable for deformable objects and is very simple to implement. It uses a hierarchical tree to decompose each object into regions. Each region requires just one parameter, its radius. In this work we compare three methods, AABB hierarchy, Sphere hierarchy, and DH, where the focus is on the intersection test, computation time, and the use of memory.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modelling

1. Introduction

Real time interaction in virtual environments requires rapid and accurate collision detection algorithms. Often these can be optimised for rigid objects by precomputing data structures holding their geometry. However, for many applications the deformations of the objects should be incorporated and in fact it is frequently a fundamental requirement for the acheivement of an accurate simulation. This is particularly the case for cloth and surgical simulations. This paper presents a novel approach to collision detection for deformable objects and compares it to two fast bounding volume approaches. The Distance Hierarchy (DH) algorithm adopts a Bounding Volume Hierarchical approach and a feature-based method to obtain a simple and fast technique for collision detection. The intersection test utilises the Euclidean distance and only requires one parameter to be updated per region during deformation.

The algorithm proceeds in two stages. Firstly, in the precomputation stage a region hierarchy is constructed using an octree. Each region contains vertices from the original object which are located in the same node of the octree. The centroid of these vertices is calculated and the closest vertex to the centroid is identified as the central vertex for the region. The maximum distance between the central vertex and a given vertex is termed as the radius of the region. The second stage takes place during the running of the simulation and involves comparing the distance hierarchies for each object in the environment. The overlap test for two regions from separate hierarchies must determine the distance between the two central vertices. An intersection occurs if this distance is less than the sum of the regions' radii.

Our contribution is a collision detection method which is suited to deformable models since it can follow the mesh deformation. The method can be adapted for efficient computation by easily altering the number of levels in each hierarchy and more subtely by differing the values for the radii of the regions. The algorithm detects collisions in several regions and therefore is able to compute multiple points of contact, which is essential for deformable objects. It can be utilised in scenes with many complex objects and it is independent of the method used to deform the geometry. A comparison of our method with the Sphere Bounding Volume (SBV) hierarchy and the Axis-aligned bounding box (AABB) hierarchy is presented, with respect to the time, spatial complexity, and the intersection test. This paper is organised as follows: previous work is described in the next section, Section 3 gives details about the three parts of the method and discusses the experiments made to test the three algorithms. Section

4 presents the overall time for the approaches and finally, in Section 5 we present our conclusions and future work.

2. Previous work

Many algorithms have arisen to detect collisions between deformable models, and a recent survey with details about many important methods can be found in [TKH*05]. Bounding Volumes are utilised in many applications to approximate complex objects in an environment in order to efficiently compute an approximate collision. Spheres have been used in a wide range of applications since they are easy to define, have a computationally cheap overlap test and are rotationally invariant [SBT06, JP04]. An AABB also has an efficient overlap check, which is accomplished via a simple comparison of its coordinate values [vdB97, LAM01]. The Oriented Bounding Box (OBB) can bound the object more tightly than an AABB since it is oriented to best align with the underlying geometry. However, it does require a more expensive overlap test [GLM96]. Other volumes include Discrete Oriented Polytopes [KHM*98, FF03], Sphere-Swept Volumes [LGLM00, RKL*04], and Convex Hulls [PLM95].

For polygonal meshes, feature-based method working on the vertices, edges, and faces can be utilised. A well known collision detection algorithm based on the tracking of closest features is the Voronoi-Clip algorithm [Mir98], which operates on a pair of polyhedra. Three representative algorithms that work with rigid convex polyhedra are as follows: firstly, a feature based incremental method that walks on the boundary of polyhedra [Mir98, LC91]. Secondly, the Dobkin-Kirkpatrick algorithm [DK90] starts from the innermost layers of hierarchies and tracks the closest feature-pair from layer to layer. Thirdly, the H-Walks [GHZ99] which starts on the boundaries of polyhedra, walks on the same layers for a few steps, and then drops to inner layers to take shortcuts.

3. The DH Algorithm

This work is motivated by the need to achieve a simple and efficient algorithm to detect collisions between deformable objects. Bounding Volume Hierarchy (BVH) algorithms consist of two stages: the hierarchy construction to decompose the object into smaller parts and the hierarchy traversal to determine the overlap status of two bounding volumes in their respective hierarchies.

Let $P = \{O_1, ..., O_N\}$ be a set of N objects in an environment, where object i, O_i, is represented by a triangular mesh, whose features are faces, edges, and vertices. A BVH algorithm calculates a collision between two objects using volumes which bound a subset of these features to return the closest pair of primitives between them. The following sections discuss the design of the DH algorithm with direct comparison to SBV and AABB.

3.1. First Stage: Hierarchy Construction

In this phase, each object's mesh is divided into regions. A top-down approach is employed using an octree as a tool for the subdivision. In order to compare the three methods the same number of regions or bounding volumes will be created. This process takes $O(nlogn)$ time for an object of n primitives. The centre, C, of the SBV is computed as the average of all the vertices within the bounding volume, and the centre for DH corresponds to one of the vertices in the region termed the central vertex, C'. The vertex is chosen such that for all v within the bounding volume $|C - C'| < |C - v|$. The radius is computed in the same way for both methods using $R = max|C - v|$ and $R' = max|C' - v|$, for the SBV and for the DH respectively.

In the case of the AABB, boxes are obtained from the minimum and maximum coordinates of the underlying geometry defined in the region. The AABB approach requires one loop to enclose the primitives while the SBV method requires two loops, one for the centre and the other for the radius. The DH method requires three loops during construction, the first to compute the average of all the vertices within the region, the second computes the closest vertex to the average termed the central vertex and finally the third loop determines the radius. A bounding volume in the hierarchy has one of the following representations:

$$Sphere(C, r) \tag{1}$$

$$DH(C', r) \tag{2}$$

$$AABB(Corner_{min}, Corner_{max}) \tag{3}$$

In the previous equations, the SBV requires a vector of three floats and a scalar float (16 bytes in total), a DH region requires an integer and a float (8 bytes in total), and the AABB requires two vectors of three floats (24 bytes in total). Therefore, DH requires less memory to represent a bounding volume.

3.2. Second Stage: Hierarchy Traversal

In order to obtain the collisions between the objects in real-time, the hierarchies must be traversed efficiently. To compare the processes involved, the algorithms were tested on eight instances of the same model, a pawn with 6,976 polygons, subdivided into five hierarchical levels. Figure 1 illustrates the objects in collision. An experiment was carried out to investigate the performance characteristics of the three methods under the same conditions. The program was implemented in C++ and executed on a computer with 3 GHz Pentium 4 CPU and 1 GB RAM. To compute the deformations during the simulation a mass-spring network was incorporated. Two main procedures are required at each stage: the intersection test and the recalculation of the bounding volumes.

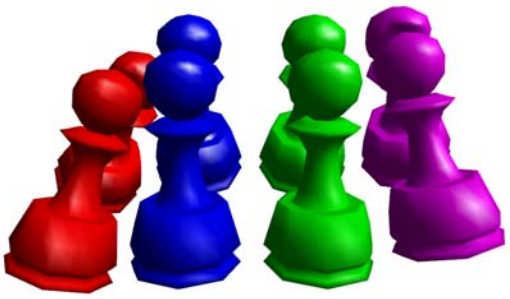

Figure 1: *The scene for the experiment including eight deformable pawns.*

3.2.1. Intersection test

In the intersection test, the SBV and DH approaches require 6 additions, 4 multiplications, and one comparison. In worst case the AABB requires 12 comparisons and 3 boolean operators. Since we are employing the same hierarchical subdivision, we could expect to have the same number of intersection tests, but this is incorrect because it depends on the tightness of the bounding volume. Figure 2 shows a graph of the number of bounding volume pairs tested in every frame against the simulation time steps. The DH approach requires a higher number of tests than the other approaches with the AABB requiring significantly less. However, a collision is reported only in the leaf bounding volumes of the hierarchy resulting in a similar number of collisions for each of the methods.

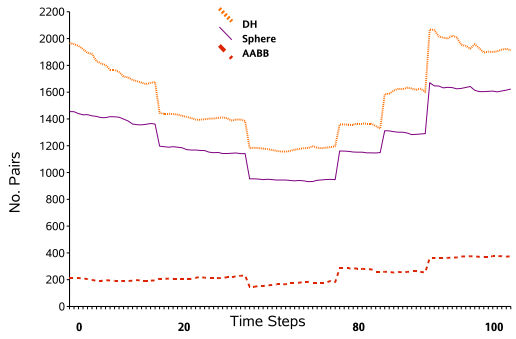

Figure 2: *A graph showing the number of bounding volume pairs tested in each of the methods during* 100 *timesteps.*

3.2.2. Recalculating the Bounding Volumes

As the objects deform, the bounding volumes will eventually no longer fit the underlying geometry correctly. For this reason the bounding volumes must be recalculated to follow the deformations of the object. This recalculation adopts a lazy-evaluation strategy where bounding volumes are only updated just before they are required in the intersection test.

To update an AABB, the minimum and maximum values of the box must be recomputed. This requires 6 comparisons per primitive inside a region, while the SBV requires the centre to be recomputed first and then the radius. This results in two loops over the number of vertices in the region. However, to update a DH region only the radius is recalcuted because the centre is defined by the central vertex, whose position is directly effected by the mesh deformation. As can be observed in Figure 2, DH needs to update more bounding volumes than the other methods. However, as explained the update procedure for DH is less involved.

There is a relationship between the number of intersection tests and the number of collisions reported. Figure 3 illustrates the number of collisions against the time steps of simulation for each of the three methods. The difference between the three methods comes from the tightness of the bounding volume, since the tighter the BV the smaller the number of intersection tests. For the objects tested in our experiment the AABBs bound the geometry tighter than the other two approaches. However, the cost of updating the AABBs is higher than the DH approach.

4. Time Complexity

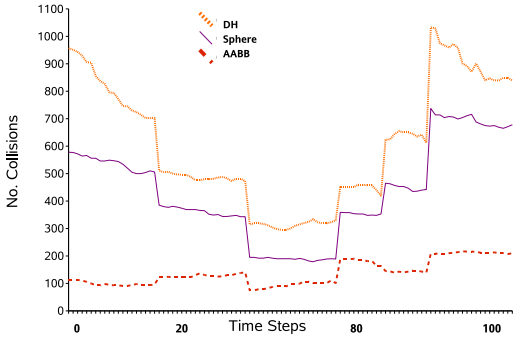

Figure 3: *A graph showing the number of collisions reported for each method over 100 time steps of the simulation.*

According to [KHM*98], the total cost of intersecting two hierarchies, T, is given as:

$$T = N_V C_V + N_P C_P + N_U C_U, \quad (4)$$

where N_V is the number of BV pairs tested for overlap, C_V is the cost of testing a pair of BVs for overlap, N_P is the number of primitive pairs tested, C_P is the cost of testing a primitive pair, N_U is the number of BVs that need to be updated, and C_U is the cost of updating such a BV. From here, the total time of the three methods can be computed and the results are shown in Figure 4, where we can see that DH is the faster method even though it makes more computations, due to its efficient updating process.

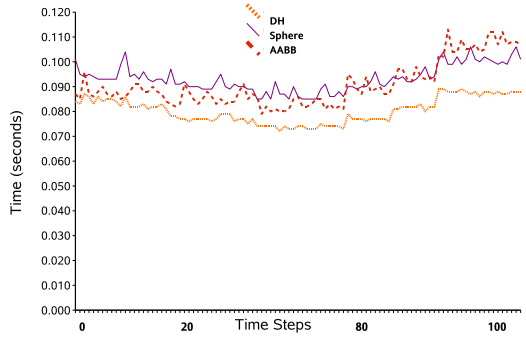

Figure 4: *A graph showing the time to compute the collisions between eight objects over 100 time steps of the simulation.*

5. Conclusions

To evaluate the BVH algorithms, the intersection test, the number of primitives involved, and the updating process required was considered. In a test scene with more than 55,000 polygons, the overall computation time for the DH algorithm is less than the AABB and the SBV. This is despite the fact that the number of intersecting pairs is greater for DH in comparison to the others. The DH algorithm is particularly well suited to deformable objects since the time to update the data structures is relatively small when compared to AABB and SBV. Future work will investigate more efficient ways of computing the radius for the DH algorithm and techniques for updating the bounding volumes during deformation.

6. Acknowledgments

We wish to thank to the Universidad Autonoma de Yucatan and the Mexican program PROMEP for funding F. Madera's PhD program. We are also grateful to the reviewers for their feedback.

References

[DK90] DOBKIN D. P., KIRKPATRICK D. G.: Determining the separation of preprocessed polyhedra: a unified approach. In *Proceedings of the seventeenth international colloquium on Automata, languages and programming* (New York, NY, USA, 1990), Springer-Verlag New York, Inc., pp. 400–413.

[FF03] FUNFZIG C., FELLNER D. W.: *Easy Realignment of k-DOP Bounding Volumes*. Tech. rep., University of Technology Muhlenpfordtstr, Institute of Computer Graphics, 2003.

[GHZ99] GUIBAS L. J., HSU D., ZHANG L.: H-walk : Hierarchical distance computation for moving convex bodies. In *Symposium on Computational Geometry* (1999), pp. 265–273.

[GLM96] GOTTSCHALK S., LIN M., MANOCHA D.: Obb-tree: A hierarchical structure for rapid interference detection. In *Proceedings on SIGGRAPH 96* (New York, 1996), ACM, pp. 171–180.

[JP04] JAMES D., PAI D.: Bd-tree: Output-sensitive collision detection for reduced deformable models. *ACM Transactions on Graphics (SIGGRAPH 2004) 23*, 3 (2004).

[KHM*98] KLOSOWSKI J. T., HELD M., MITCHELL J. S. B., SOWIZRAL H., ZIKAN K.: Efficient collision detection using bounding volume hierarchies of k-dops. *IEEE Transactions on Visualization and Computer Graphics 4*, 1 (1998), 21–36.

[LAM01] LARSSON T., AKENINE-MÖLLER T.: Collision detection for continuously deforming bodies. In *Eurographics 2001, Short Presentations* (Manchester, September 2001), Eurographics Association, pp. 325–333.

[LC91] LIN M. C., CANNY J. F.: A fast algorithm for incremental distance calculation. In *IEEE International Conference on Robotics and Automation* (1991), pp. 1008–1014.

[LGLM00] LARSEN E., GOTTSCHALK S., LIN M., MANOCHA D.: Fast distance queries using rectangular swept sphere volumes. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)* (San Francisco,CA, April 2000), vol. 4, pp. 24–48.

[Mir98] MIRTICH B.: V-clip: fast and robust polyhedral collision detection. *ACM Trans. Graph. 17*, 3 (1998), 177–208.

[PLM95] PONAMGI M., LING M., MANOCHA D.: Incremental collision detection for polygonal models. In *Proceedings of the eleventh annual symposium on Computational geometry* (Vancouver, British Columbia, Canada, 1995), pp. 445– 446.

[RKL*04] REDON S., KIM Y., LIN M., MANOCHA D., TEMPLEMAN J.: Interactive and continuous collision detection for avatars in virtual environments. In *IEEE Virtual Reality Conference 2004 (VR'04)* (2004), IEEE, pp. 117–130.

[SBT06] SPILLMANN J., BECKER M., TESCHNNER M.: Efficient updates of bounding sphere hierarchies for geometrically deformable models. In *Third Workshop in Virtual Reality, Interactions and Physical Simulations VRIPHYS'06* (Madrid, Spain, Nov. 2006), EG.

[TKH*05] TESCHNER M., KIMMERLE S., HEIDELBERGER B., ZACHMANN G., RAGHUPATHI L., FUHRMANN A., CANI M., FAURE F., MAGNENAT-THALMANN N., STRASSER W., VOLINO P.: Collision detection for deformable objects. *Computer Graphics Forum 24*, 1 (2005), 61– 81.

[vdB97] VAN DEN BERGEN G.: Efficient collision detection of complex deformable models using aabb trees. *Journal of Graphics Tools 2*, 4 (1997), 1– 14.