An Efficient Collision ueries in Parallel Close Proximity Situations

Dae-Hyun Kim*, Han-Soo Choi**, and eong-Dong Kim**

*Department of Photoelectronic Information Engineering, Chosun College of Science & Technology, Gwangju, Korea
(Tel : +82-230-8036; E-mail: daehkim@chosun.ac.kr)
**Department of Information, Control and Instrumentation Engineering, Chosun University, Gwangju, Korea
(Tel : +82-62-230-7185, 7032; E-mail: hschoi, ydkim@chosun.ac.kr)

Abstract: A collision query determines the intersection between given objects, and is used in computer-aided design and manufacturing, animation and simulation systems, and physically-based modeling. Bounding volume hierarchies are one of the simplest and most widely used data structures for performing collision detection on complex models. In this paper, we present hierarchy of oriented rounded bounding volume for fast proximity queries. Designing hierarchies of new bounding volumes, we use to combine multiple bounding volume types in a single hierarchy. The new bounding volume corresponds to geometric shape composed of a core primitive shape grown outward by some offset such as the Minkowski sum of rectangular box and a sphere shape. In the experiment of parallel close proximity, a number of benchmarks to measure the performance of the new bounding box and compare to that of other bounding volumes.

Keywords: Collision detection, Proximity, Bounding volume, Top-down, parallel proximity

1. INTRODUCTION

Collision detection problems and their variants are of vital importance in many fields, such as computer animation, physical simulation, computer simulated environments, solid modeling and robot planning, especially with the emergent fields of virtual reality [1-3]. The problems concern that fact that two impenetrable objects cannot share a common region. In computer animation, object simulated in the environments change motions according to the contact constraints and impact dynamics. It is critical to computer the response in time when object collide. In physical simulation, complex interactions of hundreds of parts in the virtual prototyping system are simulated based on physics and geometry. It is important to locate the intersection points when parts collide in order to provide proper reaction. In virtual reality, a physical environment is simulated such that humans can readily visualize, explore and interact with the virtual objects in the environment. The virtual world will seem more believable if objects can receive expectable natural behavior presented as feedback from the objects in the virtual environment such as push, pull and grasp. In order to create a sense of touch between the user s hand and a virtual object, contact or restoring forces are generated to prevent penetrating into this virtual object. These forces are computed by first detecting if a collision or penetration has occurred, then determining the contact point on the object surface.

Typically the proximity query problems have two parts. The collision query algorithm and separation distance computation. A collision query determines intersection if geometric contact has occurred between given two or more object. A distance query computes the minimum Euclidean distance between two objects. An architectural design system may perform distance queries to verify that certain functional components are separated by some minimum distance at all points [4-5]. Algorithms for such queries have been extensively studied in the literature. While a number of specialized algorithms have been designed to handle a pair of a special class of primitives, the most general algorithms are based on bounding volume hierarchies. A bounding volume hierarchy is a tree of bounding volumes whose collective leaf nodes spatially enclose all the model geometry, and in which each parent spatially encloses all the geometry covered by its descendent leaf nodes as shown in Fig. 1.

Different bounding volume hierarchies are primarily categoryized by the choice bounding volume type at each node of the tree. Typical examples of bounding volumes include axis-aligned boxes [6] and spheres, and they are chosen for to the simplicity of finding collision between two such volumes. Hierarchical structures used for collision detection include cone trees, discretely-oriented polytopes and octree [7], sphere trees [8-9], R-trees and their variants [10], tree based on S-bounds, object oriented bounding box [11] etc. Other spatial representations are based on BSP tree [12] and its ex- tensions to multi space partitions [13], spatial representations based on space-time bounds [14-15] and many more. All of these hierarchical methods do very well in performing contact test, whenever two objects are far apart. However, when the two objects are in very close proximity, parallel close proximity and multiple contacts, these algorithms either use sub- division techniques or check very large number of bounding volume pairs for potential contacts.

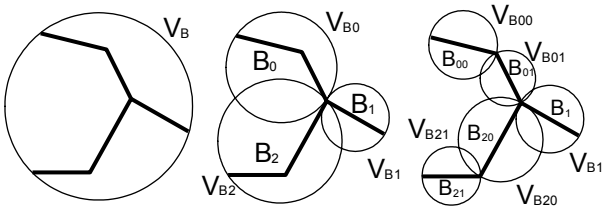
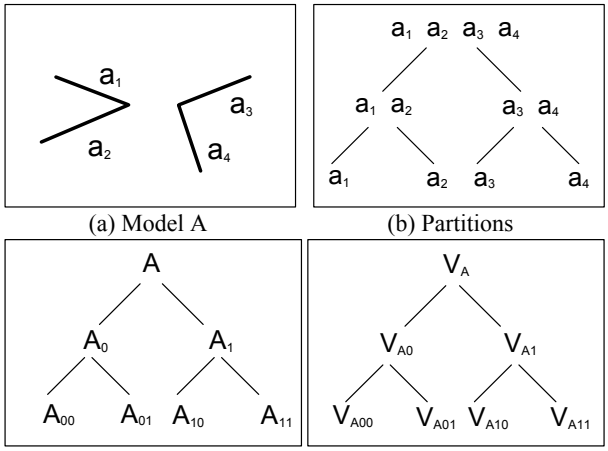


Fig. 1 A bounding volume hierarchy of spheres

In this paper, we present hierarchy of oriented rounding volume for fast proximity queries. Designing hierarchies of new bounding volumes, we use to combine multiple bounding volume types in a single hierarchy. A major motivation in the design of hybrid type hierarchies is to include simple shapes like sphere for fast overlap tests and tight fitting bounding volumes like oriented bounding box to reduce the number of tests. A bounding volume corresponds to geometric shape composed of a core primitive shape grown outward by some offset such as the Minkowski sum of rectangular box and a sphere shape. The resulting new bounding volume is like a rounded box. We represent it using the rectangle, its center, and a radius. In the parallel proximity query test, we examine the performance comparisons among types of bounding volumes.

2. OUNDING VOLUME IERARC IES

Considering the model consists of four line segments named a_1 through a_4, as shown in Fig. 2(a). The model is not completely connected. It has two disconnected pieces. The model can be recursively partitioned into a hierarchy, shown in Fig. 2(b), such that the top-level node is the entire model, the leaf nodes are the individual primitives elements, and each node equals the union of its children. The example shown in Fig. 2(b) is just one of many ways to partition the set of primitives. These nodes can be labeled as shown in Fig. 2(c). The top-level mode is the symbol for the whole model, the name of any first child is the name of the parent with a 0 subscript appended, and the name of second child has a 1 subscript appended. Thus, the sequence of 0 s and 1 s describes a path of left and right choices, starting at the root, which leads to the given node. This naming convention extends to any number of children using the digits 2 , 3 , etc. These nodes are named subsets of the original model. The sets A, A_0, A_1, A_{00}, A_{01} and so forth can be assigned bounding spheres. The bounding volume for a point set A_x is given the symbolic name V_{Ax}. Thus the bounding spheres form their own hierarchy, shown schematically in Fig. 2(d). We can redraw the original model A along with each level of the bounding volume hierarchy. Fig. 3(a) shows A with V_A, the sphere covering the entire model. Fig. 3(b) shows A_0 and A_1 with their bounding spheres. Finally, Fig. 3(c) shows the leaf nodes of the hierarchy and the model primitives they enclose. Now consider model B. The partitions of model B are given names in manner similar to that of model A, and the structure of the bounding volume hierarchy is also similar to that of model A as shown in Fig. 1.

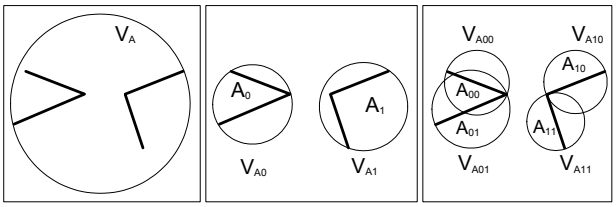


(a) Model A (b) Partitions

(c) Named partitions (d) Bounding volumes

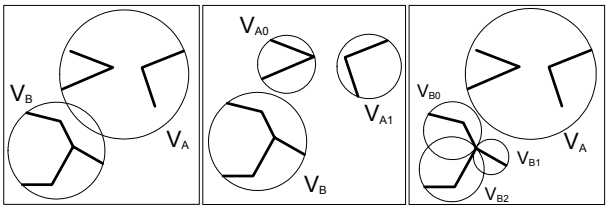
Fig. 2 Model A and symbolic representation

Given the two models and their placements in the world space, the simplest approach to perform a collision query is to test each of the four line segments in A against each of the five line segments in B, requiring 20 pairwise primitives overlap tests. While this approach is reasonable for relatively small models such as these, we cannot perform exhaustive pairwise testing on models which have millions of primitives. Instead, we perform a trivial test by checking the top-level bounding volumes for overlap. If the models are a little closer together, as shown in Fig. 4(a), the top-level spheres touch and more work is needed. There are several choices for our next step. We will choose to test the children of V_A, which are V_{A0} and V_{A1}, against V_B, as is shown in Fig. 4(b). In this processing, we say that we descend A to move from a comparisons between the pair (V_A, V_B) to two comparisons, between the pair (V_{A0}, V_B), and between the pair (V_{A1}, V_B). Since V_{A0} doesn t touch V_B, the point set A_0 doesn t touch the point set B, and likewise for A_1 and B. Since A_0 and A_1 together make up the entire model A, there is no contact between A and B. We could have chosen to test the children of V_B against model A, as shown in Fig. 4(c), with the same result that A and B are disjoint.



(a) Top level (b) Second level (c) Leaf level

Fig. 3 Graphical display of the Model A



(a) Overlapping (b) Children of V_A (c) Disjoints

Fig. 4 Situation of the models do not touch

When the models are actually in contact, we will not be able to use bounding volume tests to bound the models entirely apart. However, the bounding volume tests still enable us to eliminate whole groups of potential contacts from consideration. From this point, we could solve the contact leaf nodes of two models independently, and merge their results for the final answer. This formula can applied recursively, and is the basis for the divide-and-conquer approach used in bounding volume hierarchy based collision detection.

3. OUNDING VOLUME ALGORIT MS

3.1 Cost of proximity ueries

The fundamental operations of a proximity query are the bounding volume overlap test and the primitive overlap test. The time required to perform a collision query can be approximated as [16]

$$T = N_v \times T_v + N_p \times T_p \quad (1)$$

where N_v and N_p are number of overlap tests for bounding volumes and primitives, respectively, and T_v and T_p are the average times required to perform each such test. Queries using simple bounding volumes, such as axis aligned bounding boxes, exhibit large N_v and small T_v, while those using complex bounding volumes such as convex hull or oriented bounding boxes have smaller N_v and large T_v. Also, using simple bounding volumes tends to increase N_p, since the leaf nodes of simple bounding volumes are less likely to bound the models apart than those of complex bounding volumes. It is not immediately obvious from this equation, given the opposite tendencies of T_v and N_v, what is the best choice of bounding volume type. Several factors contribute to the cost of proximity query. Models with many polygons tend to be more expensive to query than models with fewer polygons. Proximity queries are more expensive when the

models are closer together than when they are farther apart. Thus the cost of a proximity query depends not only on the size of the number of polygons in the two models and the size of the number of touching primitive pair found, but also on the nature and degree of the proximity of the models. Therefore, no bounding volume yields optimum performance for proximity queries in all situations. A major motivation in the design of the oriented rounded bounding boxes is to include simple shape like sphere for fast overlap tests and tight fitting bounding volume like oriented bounding boxes to reduce the number of test.

3.2 Oriented rounded bounding box

Basic algorithm to fitting an oriented rounded bounding box to a model is first to choose an orientation for the bounding box, and then to choose a center, minimal edge lengths, and radius that enable it to cover the model. Suppose we have selected an orientation for our bounding box, Let the three vectors $\mathbf{v}_x$, $\mathbf{v}_y$, and $\mathbf{v}_z$ be aligned with the face normal of the bounding box. Also, let $\mathbf{p}_k$ be the kth vertex of the model being fitted, k ranges from 1 to n. Now project all the vertices of the model, $\mathbf{p}_k$, onto each of the vectors. The upper and lower extremes along each axis is given by

$$\begin{aligned} u_x &= \max_k(\mathbf{v}_x \cdot \mathbf{p}_k), \quad l_x = \max_k(\mathbf{v}_x \cdot \mathbf{p}_x) \\ u_y &= \max_k(\mathbf{v}_y \cdot \mathbf{p}_k), \quad l_y = \max_k(\mathbf{v}_y \cdot \mathbf{p}_x) \\ u_z &= \max_k(\mathbf{v}_z \cdot \mathbf{p}_k), \quad l_x = \max_k(\mathbf{v}_x \cdot \mathbf{p}_k) \end{aligned} \tag{2}$$

The ith axis of the bounding box is aligned with $\mathbf{v}_i$, and the bounding box s width along this axis will be given by $u_i - l_i$, and the center point $\mathbf{c}$ for the bounding box is given by

$$\mathbf{c} = \{(u_x + l_x)\mathbf{v}_x + (u_y + l_y)\mathbf{v}_y + (u_z + l_z)\mathbf{v}_z\}/2 \tag{3}$$

Finally, the radius vector $\mathbf{r}$ for bounding sphere and the center point $\mathbf{cp}_z$ be aligned axis are given by

$$\mathbf{r} = (u_z - l_z)\cdot \mathbf{v}_z/2, \quad \mathbf{cp}_z = (u_z + l_z)/2 \tag{4}$$

Now repeat the upper and lower length, $u_{(x,y,z)}$, $l_{(x,y,z)}$ aligned with $\mathbf{r}$ and $\mathbf{cp}_z$. This is processing to find bounding box area more tightly.

$$u_i = u_i - \sqrt{MaxOfTwo(r^2 - dz^2, 0)} \tag{5}$$

$$l_i = l_i + \sqrt{MaxOfTwo(r^2 - dz^2, 0)} \tag{6}$$

where *MaxOfTwo* is macro to find the maximum number between the two numbers. Fig. 5 shows the algorithm codes for growing max. and min. points with corner compensation.

We use statistical techniques to computer orientation for the bounding box. Our approach is based on first and second order statistics summarizing the vertex coordinates, as used by [17]. They are the mean, μ, and the covariance matrix, $\mathbf{C}$, respectively. Let the vertices of the i th triangle be the points $\mathbf{a}_i$, $\mathbf{b}_i$, and $\mathbf{c}_i$. Our fitting algorithms use the eigenvectors of the covariance matrix, $\mathbf{C}$, to initially compute an bounding box that encloses the underlying geometry. For fitting rounding box, the smallest of the three dimensions of the bounding box becomes the rectangle normal direction. This direction is most likely to be the perpendicular to a nearly flat cluster of triangles, and will allow the flat shape of the rounding box to fit the geometry tightly. The other directions fix the orientation of the rectangle and the rectangle dimensions are grown appropriately to enclose all the geometry. The dimensions of the rectangle are initially determined so that they enclose triangles along the two side projections of the rounding box.

Algorithm : *min* , *max* for *x* or *y* axis with compensation

IINPUT : $P[k][\ ]$, C_z , r , ε
OUTPUT : *max_x* , *min_x* , *max_y* , *min_y*

```
1 : FOR k ← 0 TO np
2 :   IF P[k][x] > max_x
3 :   THEN dx ← P[k][x] - max_x
4 :     IF P[k][y] > max_y
5 :     THEN dy ← P[k][y] - max_y
6 :            u ← ε( dx + dy )
7 :            t ← (εu − dx)²+ (εu − dy)²
8 :                + (C_z − P[k][z])²
9 :            u ← u − √MaxOfTwo( r² − t , 0)
10 :           IF u > 0
11 :           THEN max_x ← max_x + εu
12 :                max_y ← max_y + εu
13 :    IF P[k][y] < min_y
14 :    THEN dy ← P[k][y] - min_y
15 :           u ← ε( dx − dy )
16 :           t ← (εu − dx)²+ (−ε u − dy)²
17 :               + (C_z − P[k][z])²
18 :           u ← u − √MaxOfTwo( r² − t , 0)
19 :           IF u > 0
20 :           THEN max_x ← max_x + εu
21 :                min_y ← min_y - εu
```

Algorithm : (contd.)

IINPUT : $P[k][\ ]$, C_z , r , ε
OUTPUT : *max_x* , *min_x* , *max_y* , *min_y*

```
22 :   IF P[k][x] < min_x
23 :   THEN dx ← P[k][x] - min_x
24 :     IF P[k][y] > max_y
25 :     THEN dy ← P[k][y] - max_y
26 :            u ← ε( dx − dy )
27 :            t ← (−ε u − dx)²+ (εu − dy)²
28 :                + (C_z − P[k][z])²
29 :            u ← u − √MaxOfTwo( r² − t , 0)
30 :            IF u > 0
31 :            THEN min_x ← min_x - εu
32 :                 max_y ← max_y + εu
33 :     IF P[k][y] < min_y
34 :     THEN dy ← P[k][y] - min_y
35 :            u ← −ε( dx − dy )
36 :            t ← (−ε u − dx)²+ (−ε u − dy)²
37 :                + (C_z − P[k][z])²
38 :            u ← u − √MaxOfTwo( r² − t , 0)
39 :            IF u > 0
40 :            THEN min_x ← min_x - εu
41 :                 min_y ← min_y - εu
42 : RETURN max_x , min_x , max_y , min_y
```

Fig. 5 Algorithm for finding bounding box parameters

We use a top-down strategy to create the nodes of our hierarchy. This means that the hierarchy is build from the root node downward. The triangles in each node of the tree, starting with the root that contains all of the triangles, are split into two subsets that become the children nodes of this node as shown in Fig. 2. Nodes are recursively subdivided unless they contain only a single triangle, which corresponds to a leaf node of the hierarchy. Our splitting rule is the same used for an [17]. A splitting axis is chosen, and a plane orthogonal to the axis is used to partition the triangles into two sets, according to which side of the plane their center point lies on.

3.3 Implementation

We have implemented all our algorithms as part of a general purpose framework for performing proximity queries using bounding volume hierarchies. The framework has been implemented in C++ with OpenGL functions, and runs on top of PC s. The basic C++ struct types defining a model, a bounding volume node, a face, and a vertex. The model block is a simple C++ class which contains model-specific information and pointers to the aforementioned arrays. The hierarchies or tree data can be store the memory block for fast access time. Fig. 6 shows the graphic-user-interface program.

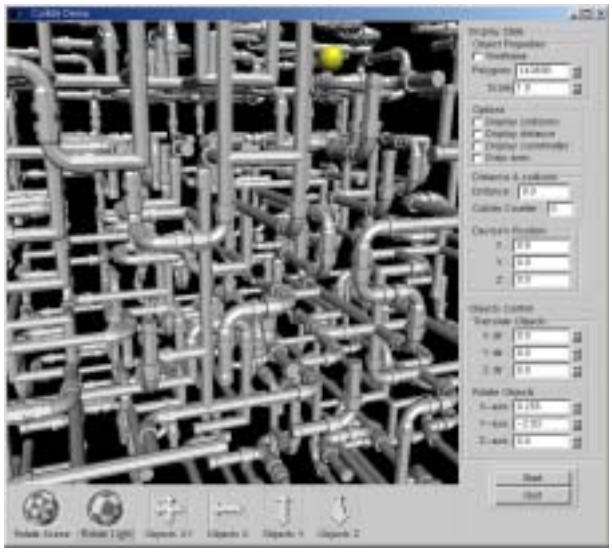
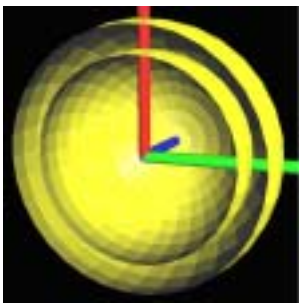
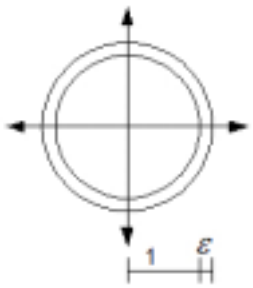


Fig. 6 Implemented program with graphic user interface

4. ENC MARKS AND ANAL SIS

We used number of benchmarks to measure the performance of our new bounding volume and compare them with other bounding volumes. Our goal was to include some real-world benchmarks that consist of a variety of configurations between the models. These include models in parallel close proximity as well as models moving away from each other. It is evident that lager models are generally more expensive to query against than smaller models. Also, proximity queries take more work when the model are close than when they are widely separated. These tendencies are diagrammed in Fig. 7(a). Every point on each surface is exactly a given ε distance from the other surface. This is one of the most challenging configurations that collision detection systems typically encounter. For this type of configuration, the amount of work to process a collision query is a sensitive function of the gap between the surfaces. The closer the surfaces are together, the more finely they must be approximated in order to bound them apart, requiring us to descend the bounding volume hierarchies of both models are deeply. Because every point on each model is close to the other model, a reduction in the gap requires a query to descend to a greater depth and every additional level descended doubles the number of nodes visited.





(a) Parallel close proximity (b) Schematic of Exp.

Fig. 7 Schematic of parallel close proximity experiment

Fig. 7(b) shows a schematic diagram of an experiment involving two concentric spheres. In this experiment we placed a 20,000 polygon sphere of unit radius inside a 20,000 polygon sphere of radius $1+\varepsilon$. For each choice of ε, we performed 100 collision queries on these spheres. For each query, we gave the spheres random orientations. The smaller sphere begins at the origin, and grows in increments of 0.001 units until surface separation reaches $\varepsilon = 4$. Fig. 8 shows a plot, for each bounding volume type, of the number of bounding volume tests over a range of surface separations, ε. For the extremely large values of ε, there is a certain minimal depth to which we must descend the bounding volume hierarchies of the outer sphere in order to bound it away from the point at its center. However, once this situation is reached, it provides significant clearance for the object in the middle. In case of extremely small values of ε, tiny gap brings the leaf nodes of the bounding volume hierarchies have been descended to their leaves, and decreasing ε even further cannot cause more bounding volumes to be tested, because there are no additional children to be visited. The slope of oriented bounding box in the log-log plot is measured -1.26, whereas the slope for new bounding box is almost exactly -1.58.

Another benchmark is diagrammed in Fig. 9. In this experiment, the surface separation was used at $\varepsilon = 0.01$. The sphere begins at the $y = -0.3$, and travels along y-axis until its center reaches $y = 1.0$. The sphere is also given random orientation, and a collision query performed which finds all the contact pairs between them. In Fig. 10 we show the plot of the number of triangle contact pairs as function of the y-coordinate of the sphere. The general trend of the plot is to rise sharply at approximately $y = 0.0$. The numbers of contacts for each bounding volume type are zero, but the distributions of the number of bounding volume tests are qualitatively different, which are shown in Fig. 10. Examination of the parallel close proximity shows that these two distributions overlap a little, but the points for oriented bounding box are generally above the points for our new bounding volume, implying that our bounding volume are more efficient than oriented bounding box at pruning the bounding volume test.

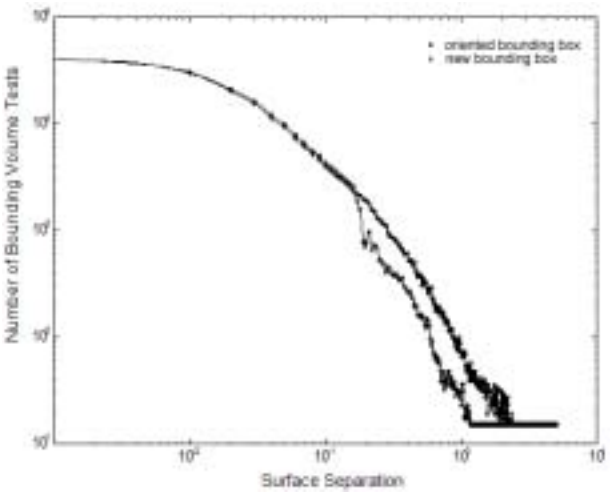
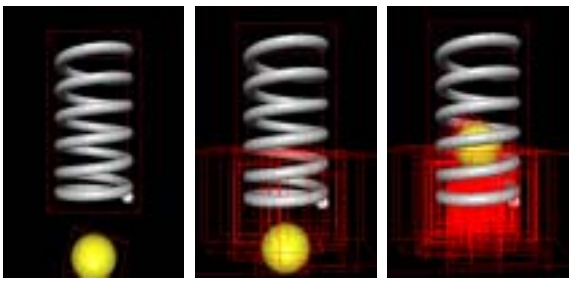


Fig. 8 Bounding volume tests vers. surface separation



(a) $y = -0.2$ (b) $y = -0.1$ (c) $y = -0.35$

Fig. 9 Bounding volume tests for parallel close proximity

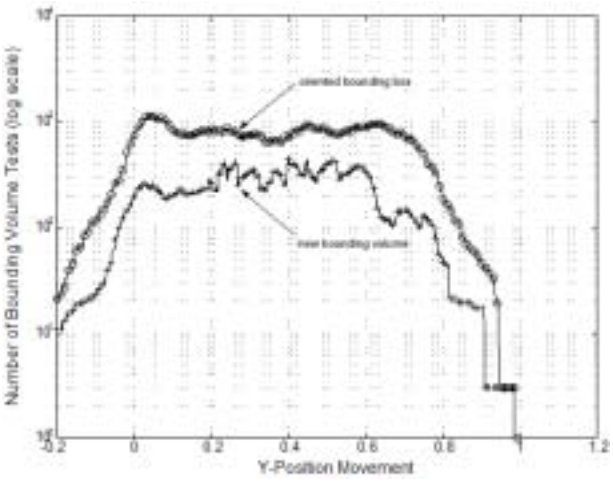


Fig. 10 Bounding volume tests vers. position movement

5. CONCLUSIONS

In this paper, we have introduced a new family of bounding volumes based on sphere and oriented bounding box, and used them to perform parallel close proximity. This bounding volume provides varying tightness of fit to the underlying polygonal model. We also examine some of the trade-offs of using new bounding volumes by analyzing benchmarks results comparing the performance of oriented bounding boxes, and show that new bounding volumes can significantly outperform the oriented bounding boxes as gap size decreases in the parallel close proximity. The benchmarks showed that our new bounding volume is 5 times faster than previously known method.

REFERENCES

[1] D. Baraff, Curved Surface and Coherence for non-penetrating rigid body simulation, ACM Computer Graphics 4(24):19-28, 1990.

[2] Gilbert, E. G. Johnson, D. W. and Keerthi, S. S. A fast procedure for computing the distance between objects in three-dimensional space, *IEEE Journal of Robotics and Automation* 4:193-203, 1988.

[3] R Moor, M. Wilhelms, J.. Collision detection and response for computer animation, ACM Computer Graphics 4(22):289-298, 1988.

[4] M. C. Lin, Efficient collision detection for Animation and Robotics, *Ph.D thesis, Department of Electrical Engineering and Computer Science*, University of California, Berkely, 1993.

[5] N. K Sancheti and S. S. Keerthi, Computation of certain measures of proximity between convex polytopes: A complexity viewpoint, *IEEE Int. Conf. on Robotics and Automation*, pp.2508-2513, 1992.

[6] Cameron, Stephen, Collision Detection by 4-dimensional intersection Testing, *IEEE Trans. On Robotics and Automation*, 3(6), 1990.

[7] H. Samet, *Spatial Data Structures Quadtree, Octrees and Other Hierarchical Methods* , addision Wesley, 1989.

[8] P. M. Hubbard, Approximating Polyhedra with Spheres for Time-Critical Collision Detection, *ACM Transac tions on Graphics 1* 179-210, 1996.

[9] S. Quinlam, Efficient Distance Computation Between Non-Convex Objects, *Proceedings of International Conference on Robotics and Automation* 3324-3329, 1994.

[10] N. Beckmann, The R*-Tree : An Efficeint and Robust Access Method for Points and Rectangles, *Proc. of SIGMOD Conference on Management of Data* 322-331, 1990.

[11] M. Ponamgi, D. Manocho, and M. Lin, Incremental algorithms for collision detection between general solid models, *In Proc. of ACM/Siggraph sym. On Solid Modeling*, pp.293-304, 1995.

[12] B. Naylor, J. Amanatides, and W. Thibault, Merging BSP trees yield polyhedral modeling results. *In Proc. of ACM Suggraph*, pp. 115-124, 1990.

[13] W. Bouma and G. Vanecek, Collision detection and analysis in a physically based simulation, Proc. *Euro graphics workshop on animation and simulation*, pp.191-203, 1991

[14] A Garica-Alonso, N. Serrano, and J. Flaquer, Solving the collision detection problem, *IEEE Computer graphics and Applications*, 13(3):36-43, 1994.

[15] J. Arvo and D. Kirk, A survey of ray tracing acceleration techniques , *In An Introduction to Ray Tracing*, pp.201-262, 1989.

[16] H. Weghorst, G. Hooper, and D. Greenberg, Improved computational methods for ray tracing, *ACM Trans actions on Graphics*, pp.52-69, 1984.

[17] Gottschalk, S. Lin, M. and Manocha, D. OBB-Tree : A Hierarchical Structure for Rapid Interfenece Detection, *International Proceeding of ACM GGRAPH96* 171-180, 1996.