Efficient Approach Based on Hybrid Bounding Volume Hierarchy for Real-time Collision Detection

ZHOU Yuan-feng, MENG Jun

Abstract: Oriented bounding box (OBB) hierarchy is a popular and efficient approach for accelerating collision detection (CD) among complex rigid models, but it can’t determine the contact status among objects far away as fast as other bounding volumes (BV), such as axis-aligned bounding boxes (AABB) or sphere BV, because each intersection test of OBBVs is much more complex than that of others. Through a method of real-time collision detection using hybrid bounding volume hierarchy (HBVH) with OBB and sphere BV, a fast intersection test was used to overlap test among two objects far away by using sphere vs. sphere test or sphere vs. OBB test, and also accurately determine the contact status among objects in closer proximity by OBB vs. OBB test. It is implemented and compared with other competitive algorithm of OBB. Experimental results prove that the HBVH algorithm is a precision and efficient approach for complex models in real-time collision detection.

Key words: collision detection; OBB; AABB; hybrid bounding volume hierarchy

Introduction

In the last years, new computer technologies have been developed to achieve more realism and enhance the level of interactivity supported by virtual environments. Collision detection (CD), the goal of which is to automatically report a geometric contact when it is about to occur or has actually occurred, is an important one of these technologies.

A popular approach for accelerating CD is bounding volume hierarchies (BVHs) which have been proved to be a successful acceleration data structure for CD between rigid bodies [1-4], such as sphere, axis-aligned bounding box (AABB), OBB, k-dop [5] and so forth. However, they emphasize on different aspects between detection speed and accuracy for some special situations. To speed up CD, we use different bounding volumes (BV) and methods of intersection tests; to increase accuracy of CD, we use different detection depths and thresholds of leaf nodes. Our algorithm, called hybrid bounding volume hierarchy (HBVH), divides the process of CD into two phases: pre-processing and real-time CD. During pre-processing we use different types of BVs to approximate HBVH tree nodes, namely using sphere BVs in the upper layer (Layer X), using a new hybrid BV structure (OBB-Sphere BV) in the middle layer (Layer Y), and using OBBs in bottom layer (Layer Z). Approximated by sphere BVs in Layer X or OBB-Sphere BVs in Layer Y, two objects far apart can determine disjunction quickly with the sphere vs. sphere intersection test. Approximated by sphere BVs in Layer X or OBB-Sphere BVs in Layer Y, two objects far apart can determine disjunction quickly with the sphere vs. sphere intersection test. Approximated by sphere BVs in Layer X and OBB in Layer Z, the objects in close proximity, can determine the contact status with OBB vs. OBB test. Approximated by OBB-Sphere BVs in Layer Y or OBBs in Layer Z, two objects in closer proximity can accurately determine contact status using OBB vs. OBB method.

The main advantages of HBVH algorithm, being a compromise between sphere BVH and OBB hierarchy, are shown as follows. First, it has characteristics of sphere BV to quickly eliminate the possibility of collision between objects far apart in the complex scene which can’t collide with each other. Second, it has characteristics of OBB, so it can build a BVH tree with less height than AABB or sphere tree, and make accurate CD between close objects, which can’t be judged
contact status by sphere BVs. Based on our previous work [6], this paper provides more detail of supporting theories and new experimental data on demonstration why HBVH algorithm is feasible and efficient.

1 Previous Work

A BV of a model is a primitive shape that encloses the model, and classic examples of BV type are sphere, AABB, OBB, K-dop which have different desired properties. We will discuss the characteristics of following common BV briefly.

1.1 AABB

The construction and intersection of AABB are simple, and also the need of storage is small. For CD between the same objects, the depth of the AABB tree is larger than OBB tree, so it will cost more intersection tests. Due to the simplicity in testing two AABBs for overlap, it has been used for CD of rigid [7, 8] and deformable body [9].

1.2 Sphere Bounding Volume

Sphere BV [10-13] uses a sphere to approximate object, and intersection tests are just to compute whether the distance of centers of spheres bounding objects is greater than the sum of both radiuses. Sphere BVs are easy to update when translating and update nothing when rotating. When the object has some special shape such as slab-sided shape, it will be approximated rather weak by sphere BV. To improve accuracy for CD, the depth of constructing tree would be increased; accordingly, the quantity of computing will be increased. Recently, the algorithm [14] makes use of sphere-tree construction hierarchical representations along with frame to frame coherence to rapidly detect collisions, and can accurately detect contacts between large geometries composed of thousands of polygons in real time.

1.3 OBB

OBB [15] is defined to be a smallest hexahedron whose orientations are discretionary to contain the object. The obvious characteristic of OBB is that orientations are discretionary, and this makes OBB approximate the object more tightly than AABB or sphere BV; hence, compared with AABB or sphere BV, OBB can improve the efficiency of detection; however, each intersection test of OBB is much more complex than that of AABB or sphere BV.

2 HBVH Algorithm

2.1 Summary of Algorithm

During pre-processing, efficient CD algorithms are accelerated by spatial data structures, including BVHs, distance fields, or alternative ways of spatial partitioning. We use the algorithms of HBVH to construct a BV tree (BVT) in a top-down manner. The BVT falls into three successive layers, which are Layer X, Layer Y and Layer Z. Layer X uses sphere as its BV; Layer Y uses OBB-Sphere hybrid structure; Layer Z uses OBB.

The methods of intersection tests for BV overlap are named Sphere-Sphere, OBB-Sphere and OBB-OBB.

2.2 Tree Traversal Algorithm

We respectively define \(a\) and \(b\) as the nodes on the BVTs of object A and B, and define \(box(a), box(b)\) as the BVs. Our kernel algorithm CDTraverse\((a, b)\) is described as Fig.1. CDFlag is the flag to judge overlapping between BVs. \(Pa\) is one of the elements in set \(Sa\) of geometric primitives, a similar expression is used for \(Pb\) and \(Sb\).

Function CDSphere_Sphere(): When searching the nodes between Layer X or Layer Y of object A and Layer X or Layer Y of object B, we use the test of sphere vs. sphere (Sphere-Sphere), defining function CDSphere_Sphere() as the description of Sphere-Sphere. Because the tightness of sphere BV in Layer X or Layer Y is not good enough and the BVs in Layer Y are constructed by hybrid BVs, so we can use the method of Sphere-Sphere to speed up intersection tests for overlap and quickly exclude the possibility of CD between two objects far away.

Function CDSphere_OBB(): When searching the nodes between Layer X of object A and Layer Z of object B, we use the test of OBB vs. sphere (OBB-Sphere) to increase the performance of CD, defining function CDSphere_OBB() as the description of OBB-Sphere. If we use Sphere-Sphere at this time, we would search more nodes although it is quite fast for each pair of nodes, and we may also fail to determine overlapping accurately even searching the bottom nodes of BVT of object B. We also have another choice to use the method of OBB-OBB, but it seems not work well. First, it may cost long time to judge the situation of CD, because using OBB-OBB based on the theory of "separating axes" will cost 15 tests at the worst situation, but OBB-Sphere based on the theory of "separating axes" only need 3 tests (see Section 4.3) at the weakest situation. Second, the BV of Layer X of object A is weakly tightened, so OBB-OBB has to cost 15 times to judge overlapping when both of the objects are in close proximity, which will cost much time on each intersection test.

Function CDobb_OBB(): When searching the nodes between Layer Y or Z of object A and Layer Z of object B, because the tightness of BV in Layer Y is better and the tightness of BV in Layer Z is much nicer after the contraction of Layer Y, we use the test of OBB vs. OBB (OBB-OBB) to increase the accurateness of CD, defining function CDobb_OBB() as the description of method. If we used OBB-Sphere again at this time, we would search more nodes although it is faster than OBB-OBB for each pair of nodes; furthermore, we would fail to judge overlapping accurately even if the searching nodes of object B were at the bottom of the BV tree. Although each OBB-OBB test for overlap may cost more time, the accuracy of it is more important than time.

Function PrimTest(): When searching the leaf nodes we use PrimTest for overlap and determine the contact status between any primitives in BVs.
3 Construction of BVs

3.1 Choices of Tree Degree

To traverse the BVT in a small number of steps, it is usually a desirable quality to minimize the height of BVT. A tree with a high degree will tend to be shorter, but more work will be expended per node of the search, so there is a trade-off between tree height and node degree. We used binary trees for all of the experiments reported herein, for two reasons. First, it is simpler and faster to compute. Second, analytical evidence as in [5] suggests that binary trees are better than d-ary trees, for d > 2. For our limited investigation of some typical searches, we have found that our choice of d = 2 is justified.

3.2 Construction Sequence of HBVH

There is top-down or bottom-up manner to construct BVT(S). A bottom-up approach begins with the input primitives as the leaves of the tree and attempts to group them together recursively until reaching a single root node representing set S. A top-down approach starts with root node representing set S, and recursively divide the nodes until reaching the leaves. We build our HBVH tree in a top-down approach, because it is one of the most important strategies that we build the HBVH tree so that we can speed more time than traditional algorithm OBB when searching the foremost nodes and also because we have more experience with top-down method.

3.3 BV Choices

Based on the characteristics of different kind of BVs, we choose different BVs to approximate tree nodes for different layers of BVT. For our algorithm HBVH, we expect the intersection test of the top nodes is simple to compute so that we can speed up CD of the objects far from each other; therefore, we use sphere to approximate the node in the top Layer (Layer X) of the tree. Considering that the BVs are gradually approximating the nodes, we choose a new BV structure (OBB-Sphere) to approximate the tree nodes in the middle Layer (Layer Y), so that it can facilitate participation in deferent methods of intersection tests between the nodes in deferent layers.

After traversing Layer Y, we need a kind of BV, which can approximate tree node more tightly and accurately detect all of the contacts between complex objects composed of thousands of triangles at interactive rates. Therefore, we use OBB boxes as the bottom nodes in Layer Z.

3.4 Construction of BVs

Our method of construction of BVs is mainly based on [16], which can help us get a tight fitting BV. For n triangles in the convex hull, the k’th triangle \( \triangle p_{i}q_{i}r_{i} \) has vertices \( p_{i}, q_{i}, \) and \( r_{i} \). The area \( a^{k} \) of \( \triangle p_{i}q_{i}r_{i} \) and the surface area \( a^{ll} \) of the convex hull are calculated as in (1):

\[
a^{k} = \frac{1}{2} \left( (p^{k} - q^{k}) \times (p^{k} - r^{k}) \right)
\]

\[
a^{ll} = \sum_{k=1}^{n} a^{k}
\]  

(1)

The centroid \( m^{l} \) of \( \triangle p_{i}q_{i}r_{i} \) and the centroid \( m^{ll} \) of the entire convex hull are calculated as in (2) and (3):

\[
m^{l} = \frac{1}{3} \left( p^{l} + q^{l} + r^{l} \right) 
\]

\[
m^{ll} = \frac{1}{a^{ll}} \sum_{i=1}^{n} a^{i} \cdot m^{i}
\]  

(2)  

(3)

The elements of the covariance matrix \( C \) are defined as in (4):

\[
C_{ij} = \frac{1}{2a^{i}a^{j}} \left[ 9m_{i}m_{i}p_{i}^{j} + p_{j}^{i}p_{j}^{i} + q_{i}^{j}q_{i}^{j} + r_{i}^{j}r_{i}^{j} - m_{i}^{ll}m_{i}^{ll} \right]
\]  

(4)

Where \( 1 \leq i, j \leq 3 \). The three eigenvectors of \( C \) will be mutually orthogonal. After normalization, the three eigenvectors is used as axes of OBB and \( m^{ll} \) is the centre of OBB. Find the maximum and minimum extents of the original triangle set along each axis, then size the OBB, the dimensions of which are represented as \( a, b, c \). To build a new BV structure OBB-Sphere or a sphere BV. We build a tight fitting OBB to approximate the tree node first, and then set \( m^{ll} \) as the centre of BVs to make a OBB’s circumscribed sphere, the radius R of which is calculated as in (5):

\[
R = \sqrt{a^{2} + b^{2} + c^{2}} / 2
\]  

(5)

In this way, we can avoid arbitrary influences from interior vertices of the model, so we can not only get a tight fitting OBB, but also a tight fitting OBB-sphere hybrid BV or a sphere BV.
4 Real-time Collision Detection

During the period of real-time CD for two objects, our algorithm will proceed upon three different methods, namely Sphere-Sphere, OBB-Sphere and OBB-OBB, described as follows.

4.1 Sphere-Sphere Test

Due to the simplicity of sphere BV, two spheres A and B, given in the first picture of Fig. 2, intersect if and only if the square of distance between spheres centres $o_A$ and $o_B$ is at most the square of the sum of their radiiuses $r_A$ and $r_B$, calculated as in (6):

$$|o_A - o_B|^2 \leq r_A^2 + r_B^2,$$

(6)

4.2 OBB-OBB Test

For two objects with the OBBs A and B, given in the second picture of Fig. 2, $a_i$ and $b_i$ are respectively the dimensions of A and B, for $i = 1, 2, 3$; $A_i$ and $B_i$ are respectively the axis unit vectors of A and B; $T$ is the distance between the centres of A and B; $L$ is the separating axis unit vector, $r_A$ is the sum of the image which we obtain from projecting the box radius onto the axis. The intersection test using OBB-OBB is shown in Fig. 2. $r_A$ and $r_B$ are calculated as in (7):

$$r_A = \sum_{i=1}^{3} |a_i A_i \cdot L| \quad r_B = \sum_{i=1}^{3} |b_i B_i \cdot L|$$

(7)

To determine the contact status, we compare $|T \cdot L|$ with the sum of $r_A$ and $r_B$ as in (8):

$$|T \cdot L| > r_A + r_B = \sum_{i=1}^{3} |a_i A_i \cdot L| + \sum_{i=1}^{3} |b_i B_i \cdot L|$$

(8)

If (8) is true, we conclude that BV A intersects with BV B; otherwise, we need to use one of the other 14 separating axes as variable $L$ to determine the contact status. When 15 separating axes test are completed and every of them don’t satisfy (8), we can draw the conclusion that the two objects don’t intersect with each other.

4.3 OBB-Sphere Test

For two given objects OBB A and sphere B, given in the third picture of Fig. 2, the parameters $L$, $T$ and $r_A$ of OBB A are defined as mentioned above (see Section 2.4); $r_A$ and $r_B$ are calculated as in (9):

$$r_A = \sum_{i=1}^{3} |a_i A_i \cdot L| \quad r_B = r$$

(9)

To determine the contact status, first, we try to find at least one separating axis based on SAT as in (10):

$$|T \cdot L| > r_A + r_B = \sum_{i=1}^{3} |a_i A_i \cdot L| + r = a_i + r$$

(10)

If at least one of the three axis unit vector L is a separating axis, the two BVs are disjoined, or we will determine intersection between this pair of sphere and OBB iff for all of the eight vertexes of OBB as in (11):

$$\|x - o\|^2 \leq r^2$$

(11)

Where $x_i$ is any of the eight vertexes of OBB and $o$ is the centre of sphere. In fact, compared with classic sphere-box algorithm[16], we decreases precision in calculation of each test of OBB-Sphere to speed up the intersection test between sphere and box, but we can still accurately determine the contact status, which we will prove as follows, see Fig.3.

![Fig. 2 Intersection test for overlap respectively using Sphere-Sphere test, OBB-OBB test, OBB-Sphere test](image-url)

![Fig. 3 Closest points lie on the sphere surface and on the OBB surface. The first picture shows the closest feature combinations of face-sphere; the second and third pictures show the combinations of edge-sphere; the last two show the combinations of vertex-sphere.](image-url)

Proof: Fig. 3 shows two non-overlapping BVs, on each there is a point which is closest to the other. The closest point of OBB lies on a vertex, edge, or face; the closest point of sphere lies on the sphere surface. Therefore, there are three possible closest feature combinations: face-sphere, edge-sphere, and vertex-sphere. We analyse these three cases respectively as follows.

• Face-sphere: As rendered in the first picture of Fig. 3, the closest points lie on a face of sphere and a face of OBB. The separating axis L can be formed from a local unit axis vector of OBB, so we use (9) to determine the contact status.

• Edge-sphere: As rendered in the second and third picture of Fig. 3, the closest points lie on an edge of OBB and on a vertex. The separating axis can be formed from a line, which is orthogonal to the edge of OBB and in the plane which is constructed by the centre of sphere and the edge. We can use (10) to determine the contact status first, as rendered in the second picture of Fig. 3, if possible intersection, as rendered in the third picture of Fig. 3, we have two possible choices. The first one is to compute the distance between them directly, as described in [16], which cost much time. The second choice is that we do not compute it at all and leave it to its child node. Here we choose the second strategy, because compared with OBB, sphere doesn’t approximate BV tightly enough, but we can still accurately determine the contact status using the test (OBB-Sphere or OBB- OBB) among their child nodes.

• Vertex-sphere: As rendered in the fourth and fifth pictures of Fig. 3, the closest points lie on a vertex of OBB and on a vertex of Sphere. We can use (10) to determine the contact status first, as rendered in the third picture of Fig. 3; if possible intersection, as rendered in the forth picture of Fig. 3, we use (11) to determine whether the vertex of OBB is in the sphere; if not, we abandon the precise intersection test of OBB vs. sphere BV to speed up OBB-Sphere test, but we can still accurately determine

• 5102 •
the contact status using the test (OBB-Sphere or OBB-OBB) among their child nodes.

5 Results

The experimental environment were implemented on AM2 workstation (this has an AM2 64bit 4000+ CPU, a NVIDIA 7800GTX video adaptor and 4 GB DDR2 800 memory). To show that our algorithm is truly efficient, we directly compared the performance of our algorithm against other competing algorithm call. In general, it is difficult to make such direct comparisons, because authors of CD papers do not normally give out the code that they used to get experimental results. However, we get the C++ language code for the competing CD algorithm OBB, of which the experimental platform named RAPID, whose algorithm called OBB was one of the best standards to test the algorithm of rigid CD.

In the first experiments described in this section, two knots (either of them had 4560 faces and 2299 vertexes) were built, started at different positions (non-penetrating) and moved toward each other at moderate speed. Fig. 4, which has five sampling pictures, respectively shows the two knots intersection with different faces and our algorithm could accurately reports colliding triangles.

The left picture of Fig. 5 shows the computation cycles of HBVHS (HBVH System) and RAPID at the stage of the proposed CD between two identical knots against the number of contact faces. We can also see that HBVHS gives better performance than RAPID. Based on these experiments, it seems reasonable to perorate that HBVHS could perform quite well in many applications. However, we cannot assert that HBVHS is the fastest for all possible applications. There has already been much research on CD for rigid objects, and many of them are especially efficient in some applications. For the time being, we have provided a practical solution to the problem of real-time CD; but we need more memory requirements than other algorithms, such as AABB, OBB.

The right picture of Fig. 5 shows FPS of HBVHS at the stage of the proposed CD between two identical knots against the number of faces. As can be seen from the various graphs, our CD algorithm is rather efficient. FPS is short for frames per second, a measure of how much information is used to store and display motion. In general, the minimum FPS needed to avoid jerky motion is about 30. As the right picture of Fig. 5 proved, HBVHS could render the scene smoothly and perform real-time CD for objects having up to 4560 faces (triangular patches), 2299 vertexes, 1347 contact faces, and the minimum FPS is above 33 FPS. Fig. 6 proves that our algorithm could simulate other complex models composed of thousands of triangles at interactive rates when they intersected with each other.

Fig. 4 Two knots respectively intersect with 408 faces, 1191 faces, 4354 faces, 5231 faces, and 6857 faces when moving toward each other. It shows our HBVH could accurately reports colliding triangles rendered in different colours, namely green and red contact triangles respectively on the two knots.

Fig. 5 The left picture shows speed comparison between HBVHS and RAPID; the right picture shows the Lowest FPS of HBVHS is at least 33 FPS.

Fig. 6 Our HBVH algorithm is adapted to real-time CD between complex models. Another two knots (9600 faces each, 4800 vertexes each) respectively intersect with 1088 faces on 42.26 FPS, with 856 faces on 53.74 FPS, 1082 faces on 42.98 FPS. A bunny (902 faces, 453 vertexes) runs into a teapot (16256 faces, 8257 vertexes) respectively with 520 faces on 103.49 FPS, with 1190 faces on 54.26 FPS, with 1269 faces on 46.63 FPS.
6 Conclusion

We have proposed a HBVH algorithm for efficient CD among complex polygonal models. We build a BVT with sphere BVs, OBB-Sphere hybrid BVs and OBBs to approximate its tree nodes in different layers (Layer X, Layer Y and Layer Z) from top to bottom, and use some intersection test (Sphere-Sphere, OBB-Sphere and OBB-OBB test) for overlap. Especially for the validity of OBB-Sphere test, we give out the details of proof. Our algorithm have been implemented and tested for a variety of models and the design parameters. Our results have shown, compared with a leading system (RAPID) based on OBB for rigid CD, our methods provide more flexibility and better result compared with a leading system (RAPID) based on OBB for rigid CD, our methods provide more flexibility and better result compared with a leading system (RAPID) based on OBB for rigid CD, our methods provide more flexibility and better result compared with a leading system (RAPID) based on OBB for rigid CD, our methods provide more flexibility and better result compared with a leading system (RAPID) based on OBB for rigid CD, our methods provide more flexibility and better result.

References:

5 Conclusions

A preliminary open T&E architecture was built in this paper, which can be used for the effectiveness evaluation of aircrafts during the design process and under the battlefield environment. It can help to verify the reasonability of the design or the anticipated tactical/technical indexes. It can also help to find out the potential errors and correct them as soon as possible. The application of distributed architecture and database technology make it possible to extend and support the T&E of different systems and different indexes. The T&E system can also support the research of large-scale, distributed, digital and virtual information combat. Especially it can help to improve the nerve centers’ effectiveness of current warfare, such as command, control and communication.

References: