Efficient Collision Detection Using a Dual Bounding Volume Hierarchy

Jung-Woo Chang[1], Wenping Wang[2], and Myung-Soo Kim[1]

[1] Seoul National University, Korea
[2] University of Hong Kong, Hong Kong

Abstract. We perform collision detection between static rigid objects using a bounding volume hierarchy which consists of an oriented bounding box (OBB) tree enhanced with bounding spheres. This approach combines the compactness of OBBs and the simplicity of spheres. The majority of distant objects are separated using the simpler sphere tests. The remaining objects are in close proximity, where some separation axes are significantly more effective than others. We select 5 from among the 15 potential separating axes for OBBs. Experimental results show that our algorithm achieved favorable speed up with respect to the existing OBB algorithms.

Keywords: Collision detection, Bounding volume hierarchy, OBB, Sphere.

1 Introduction

Collision detection is one of the most important geometric queries, with diverse engineering applications in areas such as robotics, computer graphics, animation, computer games, virtual reality, simulation and haptic rendering [1]. Because of their importance, collision detection algorithms have been studied for decades [12].

Among many different approaches, the bounding volume hierarchy has proved to be the most successful in contemporary systems [1]. The computation time of this approach can be formulated as follows [3]:

$$T = N_v \times C_v + N_p \times C_p, \tag{1}$$

where
T: total execution time
N_v: number of bounding volume pair overlap tests
C_v: time for testing a pair of bounding volumes
N_p: number of primitive pair overlap tests
C_p: time for testing a pair of primitives

This formula clearly shows that the performance mainly depends on two factors: the tightness of the bounding volumes and the simplicity of overlap test for a pair of bounding volumes. The first factor is related to N_v and N_p, whereas the second factor is related to C_v.

Spheres [6,14] and axially aligned bounding boxes (AABBs) [2] allow the simplest overlap tests. On the other hand, oriented bounding boxes (OBBs) [3] and discrete orientation polytopes (k-DOP) [8] fit volumes more tightly. In this paper, we propose a dual scheme that combines the simplicity of spheres and the compactness of OBBs to produce an efficient algorithm. The experiments show that the performance improvement over conventional algorithms is favorable.

Van den Bergen [19] suggested a simple but rough separation test for OBBs known as SAT lite. By using only 6 of the 15 potential separating axes, SAT lite demonstrates a better performance than the original OBB algorithm that uses all 15 axes [3]. In this paper, we use a sphere test followed by 5 separation axis tests. These 6 tests may look the same as the 6 tests of the SAT lite. However, the main difference is that two objects that have passed the sphere test can be expected to be in close proximity. In this stage, the choice of which 5 axes among the 15 possible axes becomes very important.

The main contribution of this work can be summarized as follows:

– A dual OBB-sphere tree is proposed, where each OBB node is enhanced with a bounding sphere.
– The more efficient sphere test is applied first to eliminate distant objects.
– We propose a selection of five separation axes that are effective in separating objects which are very close but not overlapping.
– For a wide range of experiments, the performance improvement has been observed over conventional OBB algorithms.

2 Related Work

In this section, we will briefly review related work on collision detection. The most basic type of collision detection algorithm deals with rigid bodies in static poses. But many recent studies have looked at more complicated problems, including detecting collisions between deformable models rather than rigid models [7,9,10,11,17,22], collision detection in the continuous time domain rather than static pose [15,16] and collision detection algorithms which can run on graphics hardware [4,22]. Even though these complicated problems deal with more general cases, the basic type collision detection algorithm is still quite important because the algorithms for the basic type problem are much more efficient than the algorithms for complicated problems. In this paper, we focus on the algorithm for the basic type problem.

The most widely used algorithms for detecting collisions between static, rigid bodies are based on a hierarchy of bounding volumes. As formulated in Equation (1), the performance of this approach is governed by the tightness of the bounding volumes and the simplicity of the overlap test for a pair of bounding volumes. Because the selection of the bounding volume is closely related to the performance of an algorithm, many different kinds of bounding volumes have been suggested.

Beckmann et al. [2] proposed the AABB tree and Palmer et al. [14] and Hubbard [6] put forward the sphere tree to solve the problem. By introducing OBBs, having additional rotational degrees of freedom, Gottschalk et al. [3] constructed a new and efficient collision detection algorithm. Klosowski et al. [8] suggested the k-DOP tree which tightly bounds the underlying mesh and efficiently tests the overlap of two bounding volumes. To preserve the geometric feature of k-DOP, the k-DOP-based algorithm should tumble and recompute the k-DOP for rotational motion. There are some efficient methods that solve this k-DOP tumbling problem [5,21]. Larsen et al. [13] proposed the rectangular swept sphere as a bounding volume to solve the distance measure problem, which is closely related to collision detection.

Two convex objects are disjoint if and only if there exists a separating axis such that the projections of the two objects on to that axis are separate. If the objects are both convex polytopes, it is possible to enumerate a finite set of axes that can guarantee their separation, if they are separate. Two convex polytopes are separate if and only if the projections of the two polytopes are separate on at least one of these potential separating axes. The number of such axes is determined by the geometry of the polytopes. For instance, AABBs have 3 potential separating axes, which correspond to the x, y and z axis of coordinate system. Two k-DOPs have k potential separating axes and two OBBs have 15 potential separating axes. The SAT lite algorithm of van den Bergen [19] uses 6 of the 15 axes. By this rough separation test, the overall performance of the SAT lite algorithm is better than the original algorithm based on the exact OBB test.

3 Collision Detection Algorithm

We will now present an efficient collision detection algorithm for static rigid bodies. We will first describe the construction of a dual OBB-sphere tree and show how the problem can be reduced to a situation of close proximity using a sphere test. Then we will go on to describe the selection of an effective set of axes to deal with the remaining problem.

3.1 Dual OBB-sphere Bounding Volume Tree

To combine the relative advantages of OBB and sphere trees, we construct a dual OBB-sphere tree which keeps both an OBB and a bounding sphere for each node of the tree. The basic structure of the dual OBB-sphere tree is the same as the OBB tree proposed by Gottschalk et al. [3]. For every node of an OBB tree, the dual OBB-sphere tree also contains a sphere which bounds the elements of the polygonal mesh that are at that node.

There are two ways commonly used for the construction of a bounding sphere. The first method is to construct the smallest enclosing sphere [20]. The second is to fix the center of the bounding sphere at the center of the OBB. In the latter case the centers of the two bounding spheres under a separation test can be reused for the subsequent OBB separation test, so the second method provides a simpler overlap test. But the first method naturally guarantees a tighter bounding sphere. From a series of experiments, we found that the tightness of the first method is more important than the simplicity of the second for the improvement of overall performance. The dual OBB-sphere bounding volume tree therefore keeps a smallest enclosing bounding sphere and an OBB at each node.

The test for separation of a pair of nodes in the dual OBB-sphere trees is as follows. The bounding spheres of two corresponding nodes are tested whether they overlap or not. If the bounding spheres are separate, the two nodes are separate too. If the bounding spheres overlap, then the OBBs are tested for overlap using the method to be described in the following subsection.

Testing the bounding spheres has two advantages. The first is enhanced tightness. Although OBBs are generally tighter than spheres, there are some cases in which the spheres are separate when the OBBs overlap. The second advantage is a more general reduction of the problem. OBBs are only tested when they are quite close because remote OBBs are eliminated by the sphere test. An overlap test can be designed especially for the case of OBBs in close proximity, as described in the following subsection.

3.2 The Selection of a Set of Separating Axes

Gottschalk et al. [3] proved that the separation test for two OBBs can be reduced to separation tests with respect to 15 potential separating axes. Van den Bergen [19] then suggested SAT lite, which is a looser but simpler separation test that uses a subset of 6 of the potential separating axes. However, any subset of the potential separating axes can provide a viable separation test. In this subsection, we suggest a near-optimal subset of the potential separating axes for OBBs which are already known to be in close proximity.

The 15 potential separating axes for two OBBs are $\{a_0, a_1, a_2, b_0, b_1, b_2, c_{00}, c_{01}, c_{02}, c_{10}, c_{11}, c_{12}, c_{20}, c_{21}, c_{22}\}$. The axes $\{a_0, a_1, a_2\}$ are defined by the orientation of the first OBB, and the subscripts are defined by the extents of the OBB, such that the extent corresponding to a_0 is smaller than the other extents, and the extent corresponding to a_2 is larger than the other extents. The axes $\{b_0, b_1, b_2\}$ are defined by the orientation of the second OBB and the subscripts are defined as before. The axis c_{ij} is the cross product of a_i and b_j. The exact separation test for OBBs uses all 15 axes but SAT lite uses $\{a_0, a_1, a_2, b_0, b_1, b_2\}$ only. On the other hand, we propose a sphere test followed by the separation test using 5 axes $\{a_0, b_0, c_{22}, c_{12}, c_{21}\}$.

Our selection of the separating axes is based on the fact that the extent corresponding to a potential separating axis is much more important for OBBs in close proximity than it is in the general case. Because this fact is hard to illustrate in 3D, we will consider the Minkowski sum of bounding volumes in 2D [1]. The separation test for two rectangles is closely related to the containment test for a reference point with respect to slabs, and it is also determined by the Minkowski sum of the rectangles. This relation is shown in Figures 1 and 2. The left images of Figures 1 and 2 denote the objects itself and the right images denote their configuration spaces defined by the Minkowski sum of two objects. In Figure 1, two rectangles are separated with respect to the y axis, and the relative center of the two rectangles is placed outside the slab, which is orthogonal to the y axis.

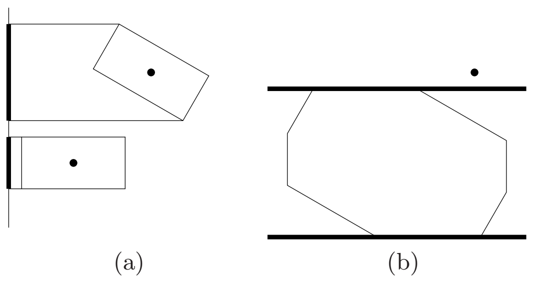

Fig. 1. The relation between the overlap test for an axis and the containment test for a slab - separation case

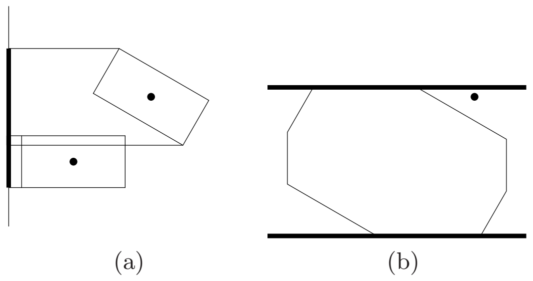

Fig. 2. The relation between the overlap test for an axis and the containment test for a slab - false positive case

Figure 2 shows a false positive case. Though the two rectangles are separated and their relative center is placed outside of the Minkowski sum, they are overlapping on the y axis and their relative center is located inside of the slab.

Figures 3 and 4 show the relation between the extent corresponding to an axis and the compactness achieved by that axis when the objects are in close proximity. In the case of 2D rectangles, the potential separating axes are defined by the orientation of each rectangle. The 4 potential separating axes can be categorized into 2 axes $\{a_0, b_0\}$ which correspond to smaller extents and 2 axes $\{a_1, b_1\}$ which correspond to larger extents. The axes $\{a_0, b_0\}$ can be called as minor axes and the axes $\{a_1, b_1\}$ can be called as major axes like the diameters of ellipse are commonly called. Diagram (c) in both Figures shows the importance of the extent corresponding to an axis when the objects are in close proximity. If the relative center of two objects is contained in the Minkowski sum of two bounding circles, the two bounding rectangles must be tested. Regarding the selection of separating axes, the Minkowski sum of two bounding circles can be subdivided to 4 regions - white region, light gray region, dark gray region and black region. The white region shows that the two rectangles overlap for all 4 potential separating axes. In the light gray region the two rectangles are separated with respect to minor axes and in the dark gray region the two rectangles are separated with respect to major axes. If the relative center of two objects is within the black

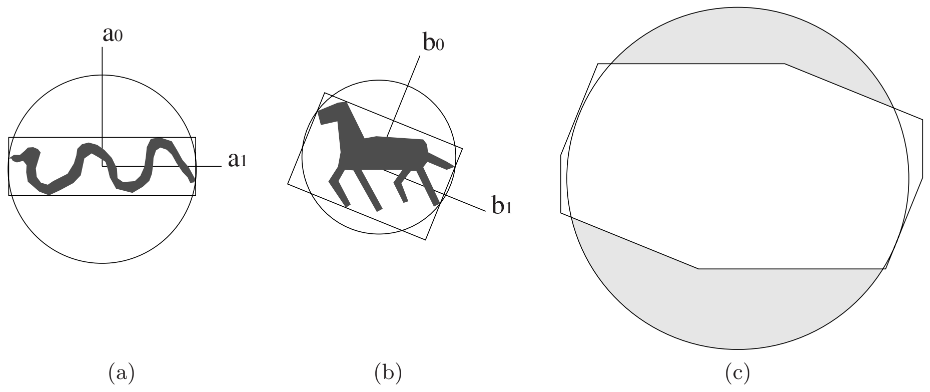

Fig. 3. Minkowski sum of bounding volumes (general case): (a) O_1 and its bounding rectangle and bounding circle; (b) O_2 and its bounding rectangle and bounding circle; (c) The Minkowski sum of the two bounding rectangles and that of the two bounding circles

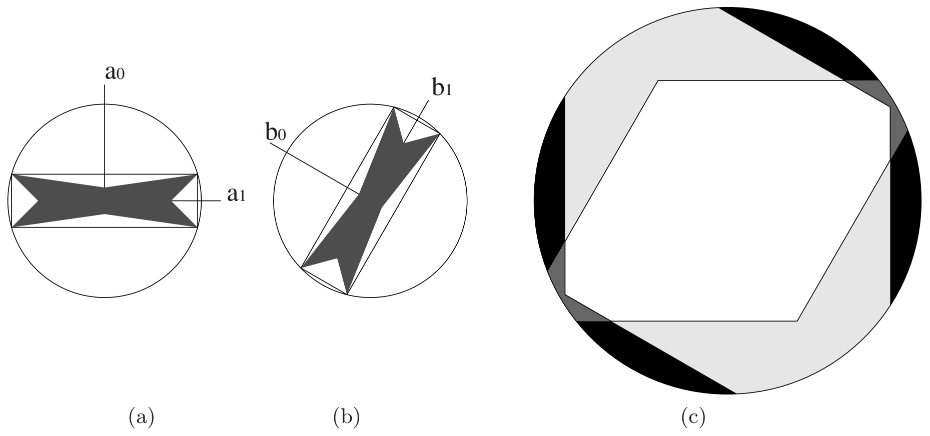

Fig. 4. Minkowski sum of bounding volumes (extreme case): (a) O_1 and its bounding rectangle and bounding circle; (b) O_2 and its bounding rectangle and bounding circle; (c) the Minkowski sum of the two bounding rectangles, that of the two bounding circles, and the clipped parallelograms defined by each set of separating axes

region, then either the separation test with respect to the minor axes or the separation test with respect to the major axes will detect the separation. In these figures, the reduction in discrimination that comes from eliminating the 2 major axes is denoted by the dark gray region, which is relatively small in Figure 4 and is absent in Figure 3. These examples support the elimination of the separation test for major axes. A more detailed discussion about the loss of discrimination region is presented in the Appendix.

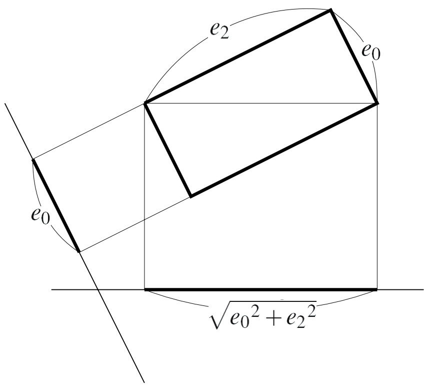

Fig. 5. The length of projection

The above suggests that an axis which corresponds to a smaller extent should be preferred. For three-dimensional OBBs, a_0 is likely to be a more discriminating axis than a_1 or a_2 and we would expect b_0 to be better than b_1 or b_2. In the case of axes defined as a cross product, we need to estimate the length of projection. This can be most easily shown by an example. The extents corresponding to each axis can be written $\{e_0, e_1, e_2\}$ and $\{f_0, f_1, f_2\}$, where e_0 corresponds to a_0 and f_0 corresponds to b_0. The axis c_{12} is orthogonal to a_1 and to b_2. Because of this orthogonality, the projection of the first OBB on to c_{12} can be reduced to the projection of a rectangle whose extents are e_0 and e_2. The length of the projection is bounded by the interval $[e_0, \sqrt{{e_0}^2 + {e_2}^2}]$ as shown in Figure 5. In the same way, the projection of the second OBB on to c_{12} is bounded by the interval $[f_0, \sqrt{{f_0}^2 + {f_1}^2}]$. Because the axis with the smallest length of projection is likely to be the most effective, c_{22} is chosen rather than c_{11} or c_{00}.

The foregoing discussion can be summarized as follows:

– In the case of axes defined by the orientation of OBBs, the axis with the smallest extent is preferred.
– An axis c_{ij} is preferred if the extents corresponding to a_i and b_j are relatively large.

By using these rules, the potential separating axes can be sorted. The remaining problem is to determine the size of the reduced set of axes. We used a greedy framework for a series of experiments. Each axis is added to the reduced set of axes in the order of preference and tested to see whether it improves the performance of collision detection. The experiments show that the optimal size of the reduced set of axes is 5 and the final set of axes is $\{a_0, b_0, c_{22}, c_{12}, c_{21}\}$.

4 Experimental Results

We have implemented our collision detection algorithm in C++ on an Intel Core Duo 1.66GHz PC with a 1.5GB main memory. To demonstrate the performance

Table 1. The execution time (in sec)

		RAPID	QuickCD	SAT lite	Dual
Scenario I	0%	27.2540	50.5709	25.3296	20.6053
	1%	14.0696	36.4444	13.2419	10.1924
	2%	8.6457	26.2898	8.1142	5.8939
	3%	6.2860	19.6893	5.8670	4.0741
	4%	4.9193	15.4700	4.5779	3.0381
	5%	4.0032	12.7353	3.7149	2.3816
Scenario II	10^{-1}	0.0012	0.0112	0.0010	0.0013
	10^{-2}	0.0138	0.4909	0.0113	0.0131
	10^{-3}	0.1754	0.8663	0.1418	0.1653
	10^{-4}	0.9418	0.9339	0.7621	0.7074
	10^{-5}	1.1858	0.9409	0.9682	0.8973
Scenario III		0.9640	0.5816	0.9935	0.8690

of our algorithm, we used three heterogeneous scenarios, each applied to different input meshes and position/orientation configurations.

In Scenario I, we employed two Happy Buddha models, each constructed with 15,536 triangles, and located the two models in 229,824 different configurations of relative position and orientation. The different configurations were generated by the sphere method of Trenkel et al. [18], which is a benchmark generation method for collision detection algorithms. We used 6 different relative distances: 0%, 1%, 2%, 3%, 4%, and 5% of the size of input models. Each distance is determined by the radius of a bounding sphere. A total of 266 different positions are generated on the sphere by sampling the spherical coordinates at every 15°. Moreover, a total of 144 different orientations are generated by sampling Euler rotation angles at every 60°. For each one of six distances, a total of 38,304 configurations are tested.

Scenario II considers two concentric spheres of radius 1 and $1+\epsilon$ respectively. (This test was also conducted in Gottschalk et al. [3] and Klosowski et al. [8].) Each sphere was approximated with 79,600 triangles, and five different values of ϵ were used: $10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}$, and 10^{-5}.

In Scenario III, we repeat one test of Klosowski et al. [8], where a hand model with 404 triangles moves along a path consisting of 2,528 frames in the interior of an airplane constructed with 169,944 triangles.

For comparison purpose, in addition to ours we have also tested three other collision detection packages: RAPID, QuickCD, and SAT lite. RAPID is an OBB-based collision detection package which is available from `http://www.cs.unc.edu/~geom/OBB/OBBT.html`. QuickCD is a k-DOP-based package available from `http://www.ams.sunysb.edu/~jklosow/quickcd/QuickCD.html`. By slightly modifying the source code of RAPID, we implemented the SAT lite as well as our own algorithm called Dual.

Table 1 shows the execution times of collision detection tests applied to each scenario using four different packages. Dual is faster than RAPID about 24-40% for **Scenario I** and about 10% for **Scenario III**. It is also faster than RAPID

Table 2. The number of overlap tests

		RAPID		SAT lite		Dual		
		Box	Triangle	Box	Triangle	Sphere	Box	Triangle
Scenario I	0%	56,228,398	2,301,207	65,976,730	3,141,544	59,980,606	47,996,655	2,353,391
	1%	32,878,322	253,521	39,583,946	435,413	34,891,432	26,718,220	272,417
	2%	20,601,212	10,717	24,899,942	22,324	21,004,422	15,453,244	12,507
	3%	15,053,514	1,842	18,128,188	3,670	14,671,214	10,563,698	2,122
	4%	11,830,858	834	14,215,850	1,586	11,018,180	7,823,461	972
	5%	9,656,554	504	11,568,194	752	8,691,784	6,105,813	546
Scenario II	10^{-1}	2,735	0	2,879	0	3,778	3,081	0
	10^{-2}	34,195	0	35,283	0	44,522	35,895	0
	10^{-3}	445,727	0	460,279	0	591,726	477,699	0
	10^{-4}	2,224,243	89,284	2,299,687	102,236	2,373,534	1,981,233	89,286
	10^{-5}	2,780,453	136,796	2,874,967	184,672	2,972,484	2,498,341	146,129
Scenario III		1,760,646	168,962	2,208,346	264,964	2,055,120	1,922,216	170,038

about 5-25% for 4 cases of **Scenario II** and slower than RAPID about 8% only for one case (corresponding to the largest value of $\epsilon = 10^{-1}$). Moreover, it is faster than QuickCD about 60-80% for **Scenario I** and 5-97% for **Scenario II**. Nevertheless, it is slower than QuickCD about 50% for **Scenario III**. Dual is also faster than SAT lite about 18-36% for **Scenario I** and about 13% for **Scenario III**. It is slower about 16-30% for 3 cases of **Scenario II** and faster about 7% for 2 cases (corresponding to the smaller values of $\epsilon = 10^{-4}, 10^{-5}$).

By comparing the execution times, we realized that the k-DOP-based algorithm and the OBB-based algorithms show different patterns. This means that there are some applications which best fit to k-DOP-based algorithm and some to OBB-based algorithms. For our experiments, the outperformance of k-DOP in **Scenario III** is based on the fact that the orientation changes and the corresponding recomputation of k-DOPs are needed only for the moving hand model which is considerably smaller (404 triangles) than the whole environment (170,348 triangles). Since our algorithm is based on OBB, it showed a pattern similar to the other OBB-based algorithms. Thus we concentrate on the performance of three OBB-based algorithms below.

More detailed discussions on the performance of three OBB-based implementations are in Table 2. SAT lite uses only a subset of potential separating axes; thus the number of bounding volume overlap tests and triangle overlap tests for SAT lite is larger than that for RAPID. Though SAT lite needs more overlap tests, SAT lite is faster than RAPID for **Scenario I** and **Scenario II** since each bounding volume overlap test is considerably simpler. By comparing the number of bounding volume overlap tests and triangle overlap tests for SAT lite and that for Dual, we can show the advantage of enhanced tightness which was discussed in Section 3.1. The number of triangle overlap tests for Dual is smaller than that for SAT lite in all three scenarios. Moreover, the number of bounding volume overlap tests for Dual is smaller than for SAT lite in **Scenario I** and **Scenario III**. For 3 cases in **Scenario I**, the number of bounding volume overlap tests for Dual is even smaller than RAPID. This result implies that the reduced set of one sphere and five axes in our algorithm is more effective than the six axes of SAT lite.

Dual is faster than RAPID for all cases except the case of 10^{-1} in **Scenario II**. It is also faster than SAT lite except the three cases of $10^{-1}, 10^{-2}, 10^{-3}$ in **Scenario II**. Because the performance for the worst case is more important, the case of 0% is the most important for **Scenario I** and the case of 10^{-5} is the most important for **Scenario II**. In other words, the more difficult cases where Dual show better performance are more significant than the cases where RAPID or SAT lite show better performance.

The above experimental results show that our algorithm is a good choice when a collision detection package is needed for static rigid bodies.

5 Conclusions

We have presented a fast OBB-based collision detection algorithm that uses both OBBs and spherical bounding volumes. We have shown how to combine the compactness of OBBs and the efficient overlap test for spheres. Out of the 15 possible separation axes for two OBBs, we have selected 5 axes which detect separation most effectively. Experimental results show that our scheme makes a favorable speed up with respect to existing algorithms based on OBBs.

References

1. Akenine-Moller, T., Hains, E.: Real-Time Rendering. A K Peters (2002)
2. Beckmann, N., Kriegel, H.-P., Schneider, R., Seeger, B.: The R*-Tree: an efficient and robust access method for points and rectangles. In: ACM SIGMOD Conf. on the Management of Data, pp. 322–331 (1990)
3. Gottschalk, S., Lin, M.C., Manocha, D.: OBB-Tree: a hierarchical structure for rapid interference detection. In: ACM SIGGRAPH 1996, pp. 171–180 (1996)
4. Govindaraju, N.K., Redon, S., Lin, M.C., Manocha, D.: CULLIDE: interactive collision detection between complex models in large environments using graphics hardware. In: Proc. Eurographics/SIGGRAPH Graphics Hardware Workshop, pp. 25–32 (2003)
5. He, T.: Fast collision detection using QuOSPO trees. In: ACM Symp. on Interactive 3D Graphics, pp. 55–62 (1999)
6. Hubbard, P.M.: Collision detection for interactive graphics applications. IEEE Trans. on Visualization and Computer Graphics 1(3), 218–230 (1995)
7. James, D.L., Pai, D.K.: BD-Tree: output-sensitive collision detection for reduced deformable models. ACM Trans. on Graphics 23(3), 393–398 (2004)
8. Klosowski, J.T., Held, M., Mitchell, J.S.B., Sowizral, H., Zikan, K.: Efficient collision detection using bounding volume hierarchies of k-DOPs. IEEE Trans. on Visualization and Computer Graphics 4(1), 21–37 (1998)
9. Kavan, L., Zara, J.: Fast collision detection for skeletally deformable models. Computer Graphics Forum 24(3), 363–372 (2005)
10. Larsson, T., Akenine-Moller, T.: Collision detection for continuously deforming bodies. In: Proc. Eurographics, pp. 325–333 (2001)

11. Larsson, T., Akenine-Moller, T.: Efficient collision detection for models deformed by morphing. The Visual Computer 19(2-3), 164–174 (2003)
12. Lin, M.C., Gottschalk, S.: Collision detection between geometric models: a survey. In: Proc. IMA Conference on the Mathematics of Surfaces, pp. 37–56 (1998)
13. Larsen, E., Gottschalk, S., Lin, M.C.: Fast distance queries using rectangular swept sphere volumes. In: Proc. IEEE International Conf. on Robotics and Automation (ICRA), pp. 3719–3726 (2000)
14. Palmer, I., Grimsdale, R.: Collision detection for animation using sphere-trees. Computer Graphics Forum 14(2), 105–116 (1995)
15. Redon, S., Kheddar, A., Coquillart, S.: Fast continuous collision detection between rigid bodies. Computer Graphics Forum 21(3), 279–287 (2002)
16. Redon, S., Kim, Y.J., Lin, M.C., Manocha, D.: Fast continuous collision detection for articulated models. In: ACM Symp. on Solid Modeling and Applications, pp. 145–156 (2004)
17. Teschner, M., Kimmerle, S., Heidelberger, B., Zachmann, G., Raghupathi, L., Fuhrmann, A., Cani, M.-P., Faure, F., Magnenat-Thalmann, N., Strasser, W., Volino, P.: Collision detection for deformable objects. Computer Graphics Forum 24(1), 61–81 (2005)
18. Trenkel, S., Weller, R., Zachmann, G.: A Benchmarking Suite for Static Collision Detection Algorithms. In: International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG) (2007)
19. van den Bergen, G.: Efficient collision detection of complex deformable models using AABB trees. J. Graphics Tools 2(4), 1–14 (1997)
20. Welzl, E.: Smallest enclosing disks (balls and ellipsoids). New Results and New Trends in Computer Science 555, 359–370 (1991)
21. Zachmann, G.: Rapid collision detection by dynamically aligned dop-trees. In: Proc. of IEEE Virtual Reality Annual International Symposium (VRAIS), pp. 90–97 (1998)
22. Zhang, X., Kim, Y.J.: Interactive collision detection for deformable models using Streaming AABBs. IEEE Trans. on Visualization and Computer Graphics 13(2), 318–329 (2007)

Appendix

We present a statistical analysis of the loss of discrimination resulting from the elimination of the major axes in the two-dimensional case.

The overhead in discrimination corresponds to the dark gray region in Figure 4. A general formula for the area of this dark gray region would be very complicated; thus we have adopted a statistical analysis. By calculating the area for a number of samples, we provide an understanding of this overhead.

Because the space of all possible rectangles and circles is huge, we made the following assumptions:

– **Assumption 1:** The extents corresponding to a_1 and b_1 are fixed at 1.
– **Assumption 2:** To restrict the space of possible circles, each bounding circle surrounds the oriented bounding rectangle. The center of the bounding circle is fixed at the center of the bounding rectangle and its radius is determined by the size of the bounding rectangle. This assumption makes the calculation easy, and also makes the estimation very conservative.

– **Assumption 3:** Because the rectangle and the circle are both symmetrical, and the center of the bounding circle is fixed at the center of the bounding rectangle, the angle between a_0 and b_0 can be limited to the range of $0°\sim90°$ without loss of generality.

The extents corresponding to a_0 and b_0 are written h_1 and h_2, which are of course both less than or equal to 1. The angle is sampled at 10° increments and h_1 and h_2 are sampled with 0.1 increments. This gives 10 samples for each of 3 free variables, making 1,000 samples in total.

The ratio of the areas of two different regions is calculated as

$$\text{ratio} = \frac{\text{area of dark gray region}}{\text{area of white region} + \text{area of dark gray region}}.$$

The dark gray region denotes the overhead in discrimination. The white region and the dark gray region are the ones we test in our algorithm. This ratio represents the percentage of false positive case resulting from eliminating two major axes.

The average ratio for the 1,000 samples is 6.4%. For 80.7% of these samples, the ratio is less than 10%. In the worst case, the ratio is 22.2%. This worst case arises when the angle is 0° , and h_1 and h_2 are equal to 1.0.