http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 13 (10): 1723-1729, 2014
ISSN 1812-5638 / DOI: 10.3923/itj.2014.1723.1729

Hybrid Bounding Volume Hierarchy Tree (HBVHT) Algorithm for Collision Detection of Articulated Model Robot

[1,2]Lin Chen, [1,2]Jun Dai, [1,2]Junjie Feng, [1,2]Bing Fu and [1,2]Haihong Pan
[1]College of Mechanical Engineering, Guangxi University,
[2]Guangxi Key Laboratory of Manufacturing System and Advanced Manufacturing Technology,
Nanning, 530004, People's Republic of China

Abstract: To reduce time for collision detection among articulated models, the collision detection algorithm of Hybrid Bounding Volume Hierarchy Tree (HBVHT) was proposed to accelerate the speed of culling away triangles. The HBVHT was composed of two phases: A broad phase and a narrow phase. The broad phase consisted of Axis-Aligned Bounding Box (AABB) and the Bounding Volume (BV) was used to build a Multi-Level Hierarchy Tree (MLHT); the narrow phase was made up of the Oriented Bounding Box (OBB) hierarchy trees and triangles. Furthermore, according to the characteristic of hierarchical structure of the HBVHT, an improved cost function was given to analyze the performance of the HBVHT. Experiments were performed between two 6-DOF robots under the Open GL environment. Two robots with the same number of triangles moved with the same trajectory for the collision experiments. Experimental results show that the efficiency of HBVHT algorithm is higher than that of the RAPID and the other two HBVHTs with different structure. The results indicate that the HBVHT algorithm can effectively improve the efficiency of collision detection among the articulated model robots.

Key words: Hybrid bounding volume hierarchy tree, collision detection, articulated model, cost function

INTRODUCTION

The purpose of collision detection is to detect whether there are intersections among moving geometric objects in space and to identify overlap features of the intersecting objects. Therefore, collision detection has become a vital technology which is widely applied in the engineering fields, such as robotics, computing graphics, CAD/CAM, virtual reality, haptic rendering and so on. In the past two decades, many researchers have performed extensively research on collision detection algorithms (Zhang and Kim, 2007; Rashid *et al*., 2012). Some distinctive bounding volumes, such as AABB (Terdiman, 2001), OBB (Gottschalk *et al*., 1996), bounding sphere (Bradshaw and O'Sullivan, 2002), discrete orientation polytopes (K-dops) (Klosowski *et al*., 1998) and Swept Sphere Volume (SSV) (Larsen *et al*., 1999) have been proposed for collision detection, in which both Opcode (Terdiman, 2001) and RAPID (Gottschalk *et al*., 1996) are suitable algorithms for collision detection between two separate rigid objects. In order to speed up collision detection for rigid body, Bounding Volume Hierarchy Tree (BVHT) was put forward and has been widely applied (Zhang and Kim, 2007; Gottschalk *et al*., 1996; Larsen *et al*., 1999). Meantime, with the development of various algorithms, the detection objects have been extended from the rigid models to multiple rigid models, deformable models and articulated models, etc.

Collision detection of among articulated models is a hot but tough topic in present research (Redon *et al*., 2005; Yoon *et al*., 2010). Typical articulated models usually refer to industrial robots and virtual arms, kinematical or molecular chains, *etc*. Links of an articulated model are rigid, series connection and/or parallel connection. Multiple rigid models and deformed models are different in some way. Their adjacent links are not separated and there is a certain relative position between them when they move in space. The multiple rigid models have a loose positional relationship between those rigid models (Avril *et al*., 2012). The deformed models have a large number of dense primitives (Mendoza and O'Sullivan, 2006) and placements and shapes of primitives change over time within the local coordinate system of a model (Van den Bergen, 1997). However, the actual motion of articulated model links is the synthesis movement of its front-end links. With the increasing number of the links, the complexities of the actual trajectory of the end links as well as the model are all increased, thus results in much longer time for calculation and collision detection.

Corresponding Author: Haihong Pan, College of Mechanical Engineering, Guangxi University, Nanning, 530004, People's Republic of China

Previous BVHT is usually applied to perform collision detection between rigid models (Terdiman, 2001; Gottschalk *et al*., 1996; Bradshaw and O'Sullivan, 2002; Klosowski *et al*., 1998; Larsen *et al*., 1999; Redon *et al*., 2005) and it needs to be pre-computed typically. Because the BVHT can not directly be applied to articulated models with multiple moving links, (Redon *et al*., 2005) studied the continuous collision detection between a single articulated model and the work environment. He used the SSV of links to calculate the AABB for building a hierarchy which was used to cull away the links that are not in close proximity to the environment. Zhang *et al*. (2007) studied continuous collision detection among multiple articulated models to remove non-collision links and they applied Taylor models of links to calculate the links' surrounding bounding box AABB for building a dynamic hierarchy. Nowadays, how to improve the efficiency of collision detection for articulated mode has become an eye-catching research subject. However, the difficulties are obvious, this problem involves which type of Bounding Volume (BV) to be used for building the BVHT of an articulated model and how to construct a suitable structure BVHT for the articulated model and so on.

To improve the efficiency, a Hybrid Bounding Volume Hierarchy Tree (HBVHT) structure which is based on the idea of BVHT is proposed. It contains two phases: A broad phase and a narrow phase. In the articulated model, the HBVHT can be built and updated quickly even when the locations of the links change with time and it can also eliminate the non-collision links rapidly. Hence, the process of the collision detection among articulated models is accelerated. The simulation of HBVHT collision detection was performed with two 6-DOF industrial robots under the OpenGL environment.

HYBRID BOUNDING VOLUME HIERARCHY TREE (HBVHT)

The HBVHT structure contains two phases: (a) A broad phase is a Multi-level Hierarchy Tree (MLHT), which is built with AABB; (b) A narrow phase is made up of multiple Oriented Bounding Box (OBB) BVHTs and triangles of links. Both the MLHT of the broad phase and the OBB BVHTs of the narrow phase use the binary tree data structure. Furthermore, the HBVHT not only increases the building and updating speed of the HBVHT, but also enhances the speed of collision detection by taking advantages of different BVs.

In this section, we firstly describe how to build BVs in the broad phase; secondly introduce the construction method of the broad phase; thirdly expound the construction process of the narrow phase; lastly display the executing processes of the HBVHT collision detection.

Building the BVs in broad phase: The overlap test of BVs in the broad phase requires less cost and the hierarchy tree needs to be built and update quickly. It is well known that the cost of overlap test of AABBs is less than that of other BVs (Ericson, 2005); Nevertheless, BVs of LSS have merits of tighter approximation and lower updating cost, etc. Therefore, AABB is employed to construct the broad phase and LSS is used as building primitives to build the leaf node BVs.

Two steps for the calculation of AABBs in the broad phase: (1) Compute the leaf node AABB., the building primitive of leaf node AABB is LSS; (2) Calculate the parent node AABB. Each parent node AABB uses the two child nodes AABBs as its building primitive.

Compute the leaf node AABB: Calculation of an LSS has been introduced in (Larsen *et al*., 1999). Here, we mainly describe how to calculate the leaf node AABBs:

- Calculate the center coordinates of the AABB:

$$\begin{cases} X = (X_A + X_B)/2 \\ Y = (Y_A + Y_B)/2 \\ Z = (Z_A + Z_B)/2 \end{cases} \tag{1}$$

- Acquire the radii of the AABB:

$$\begin{cases} R_X = ((\mathrm{Max}(X_A, X_B) + R) - (\mathrm{Min}(X_A, X_B) - R))/2 \\ R_Y = ((\mathrm{Max}(Y_A, Y_B) + R) - (\mathrm{Min}(Y_A, Y_B) - R))/2 \\ R_z = ((\mathrm{Max}(Z_A, Z_B) + R) - (\mathrm{Min}(Z_A, Z_B) - R))/2 \end{cases} \tag{2}$$

where, O_1 (X, Y, Z) is the center coordinate of the AABB; R_X, R_Y, R_Z are three radii of the AABB. Point A (X_A, Y_A, Z_A) and Point B (X_B, Y_B, Z_B) are two endpoints of the LSS; R is the radius of the LSS.

Compute the parent node AABB: Each parent node AABB is built by two child nodes AABBs (Fig. 1), calculation steps of the parent node AABB are as follows:

Step 1: Project the surfaces of the two child nodes AABBs to each axis of the world frame and find the maximum and the minimum values of the projections on each axis:

$$\begin{cases} X_{max} = \mathrm{Max}(X_A + R_X^A, X_B + R_X^B); & X_{min} = \mathrm{Min}(X_A - R_X^A, X_B - R_X^B) \\ Y_{max} = \mathrm{Max}(Y_A + R_Y^A, Y_B + R_Y^B); & Y_{min} = \mathrm{Min}(Y_A - R_Y^A, Y_B - R_Y^B) \\ Z_{max} = \mathrm{Max}(Z_A + R_Z^A, Z_B + R_Z^B); & Z_{min} = \mathrm{Min}(Z_A - R_Z^A, Z_B - R_Z^B) \end{cases} \tag{3}$$

Step 2: Calculate the center coordinates and radii of the parent node AABB by using the maximum and minimum values of the projections on each axis:

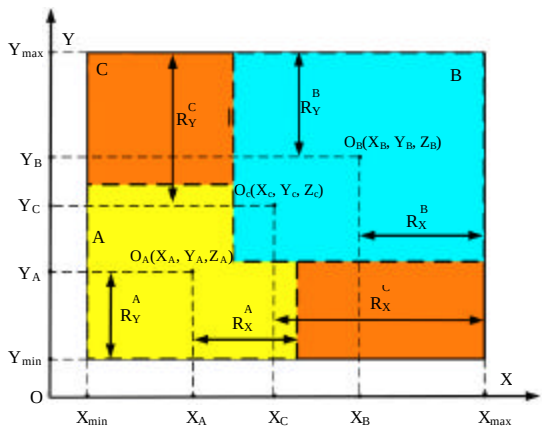

Fig. 1: Building a parent node of AABB by two child nodes

$$\begin{cases} X_C = (X_{max} + X_{min}) / 2 \\ Y_C = (Y_{max} + Y_{min}) / 2 \\ Z_C = (Z_{max} + Z_{min}) / 2 \end{cases} \quad (4)$$

$$\begin{cases} R_X^C = (X_{max} - X_{min}) / 2 \\ R_Y^C = (Y_{max} - Y_{min}) / 2 \\ R_Z^C = (Z_{max} - Z_{min}) / 2 \end{cases} \quad (5)$$

where, O_c (X_c, Y_c, Z_c) is the center coordinates of the AABB 'C'. R_X^C, R_Y^C, R_Z^C are radii of the parent node AABB, O_A (X_A, Y_A, Z_A) and O_B (X_B, Y_B, Z_B) are the center coordinates of the two child nodes AABB 'A' and 'B', respectively. R_X^A, R_Y^A, R_Z^A are radii of the AABB 'A', R_X^B, R_Y^B, R_Z^B are the radii of AABB 'B'.

Building the broad phase in the HBVHT: The broad phase is built as a MLHT structure. The MLHT is constructed with the bottom-up approach. For articulated models, the binary tree data structure is applied to construct the broad phase of the HBVHT and the entire building processes for the articulated models with the series or parallel connection are almost similar. Here, the series connection articulated model is chosen as the research object and details of its building process are particularly described.

For series connection articulated model, if the number of its links is 2^n (n is an integer), an even number nodes will be generated to calculate their parent nodes in the MLHT. The construction of MLHT starts from the left side of a leval and then from the bottom level to the top level with similar process, during which two adjacent child nodes are combined into one unit (parent node) of the upper level. If the number of links is non-2^n, odd number nodes will be generated. Under this condition, a left single node

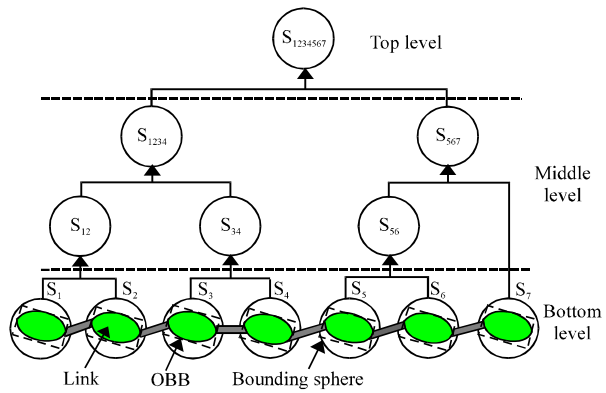

Fig. 2: A multi-level hierarchical tree (MLHT) structure of the broad phase

will be treated as the child node of its upper level to construct the parent node of the next upper level.

Figure 2 is an example of a MLHT structure of broad phase. The number of links is seven (non-2^n). Each link is surrounded by a building primitive of LSS and these LSSs are used to construct the AABBs which are labeled from S_1 to S_7 in the bottom level. The last leaf node S_7 and the node S_{56} are combined into the node S_{567} in the middle level. The top level has a root node $S_{1234567}$.

Building the narrow phase in the HBVHT: Because the OBB that surrounds the slender link is tighter than the AABB, the OBB is employed as the BV to construct the BVHT of narrow phase. The triangles of each link are used to build an OBB BVHT with a top-down manner (Gottschalk *et al*., 1996) and each leaf node OBB contains only one triangle. The binary tree data structure is used to build the BVHT. The number of OBB BVHTs is the same as the number of links. The root node of each OBB BVHT corresponds to a leaf node BV of link in the broad phase. Thus, only position and orientation of the OBBs need to be updated when the position and the orientation of links changed. The size of OBBs doesn't need to be re-calculated. Therefore, the narrow phase of the HBVHT can be updated quickly.

After all overlapped pairs of leaf node OBBs are found, the vertex coordinates of the two triangles which are contained in the overlapped pairs are converted to world frame and then the overlap tests of the converted triangle pairs are performed.

Implementation of HBVHT for collision detection: Considering engineering applications and to meet the real-time detection, a continuous trajectory is divided into some interpolation points. Since, the construction processes of various parts in an HBVHT have been

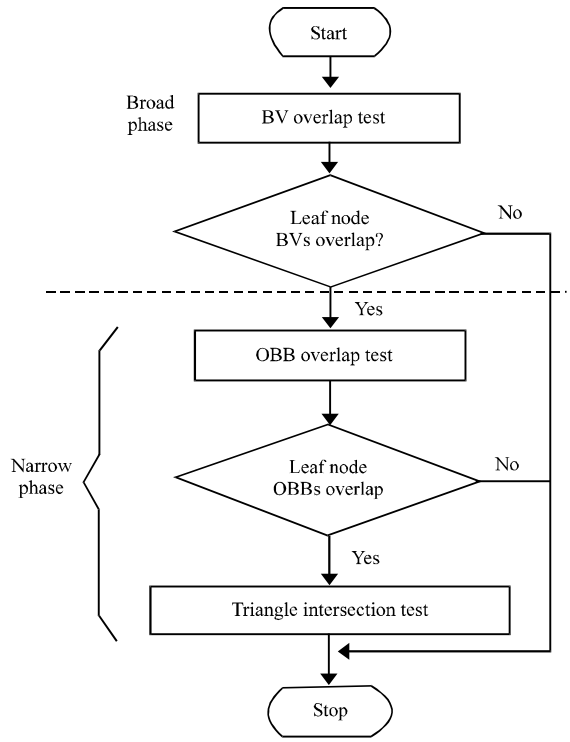

Fig. 3: Flow chart of implementation of HBVHT for collision detection

introduced, now we present the process of HBVHT for once collision detection (one interpolation point) is present as shown in Fig. 3. The procedures are described as follows:

- In the broad phase, the collision detection of BVs is performed to obtain the overlapped leaf node BVs,.Then the root node OBBs of OBB BVHTs is acquired
- In the narrow phase, the overlap tests of the OBBs are executed to search for overlapped leaf node OBBs of the OBB BVHTs and then used them to find the corresponding triangles, which need to carry out intersection test. At last the intersection test is implemented among these found triangles for acquiring the intersecting triangle pairs

If there is no leaf nodes BVs overlapped in the broad phase, the collision detection for narrow phase will be skipped; if there is no leaf nodes OBBs overlapped in the OBB BVHTs, it does not need to carry out the collision detection of triangles.

The whole process of the collision detection for a trajectory follows 5 steps: (a) Build the primitive LSS which surrounds the link for HBVHT; (b) Build the HBVHT; (c) Carry out the HBVHT collision detection at one interpolation point (it has been described above); (d) Recalculate the broad phase of the HBVHT; (e) Go to step (c) and continue executing (d), until the collision detections of all interpolation points are finished.

COST FUNCTION FOR HBVHT

Some researchers have presented formulas to analyze the impacts of the hierarchical structures of a BVHT on the cost time of collision detection. Weghorst *et al.* (1984) firstly presented a basic cost function Eq. 6 when he used the hierarchical method to analyze intersection test between light and BVs in ray tracing. This function is suitable for analyzing the collision detection between rigid objects (Gottschalk *et al.*, 1996; Chang *et al.*, 2010). But it merely considered the cost of overlap test for BV pairs and intersection test for primitive (triangle) pairs. For dynamic collision detection, the cost of nodes updating in the BVHT needs to be taken into account when the model's position and orientation change. Hence, (Klosowski *et al.*, 1998) revised the cost function as Eq. 7 and applied it to analyze K-Dops BVHT. The researcher He (1999) furtherly modified and revised the cost function as Eq. 8 by adding the cost of transformation between the coordinates of models.

Considering the hierarchical features of the HBVHT, we give a cost function as Eq. 9 for analyzing the efficiency of HBVHT collision detection. The Eq. 9 is composed of two parts: The cost consumed in the broad phase and the narrow phase. Particularly, the cost consumed in the broad phase includes the cost of overlap test of BV pairs ($N_t \times C_t$) and updating ($N_u \times C_u$). The cost of overlap test of BV pairs ($N_v \times C_v$) in the narrow phase includes the cost of coordinate system transformation of the model and BV and the cost of updating for nodes is considered as well:

$$T_1 = N_v \times C_v + N_p \times C_p \tag{6}$$

$$T_2 = N_v \times C_v + N_p \times C_p + N_u \times C_u \tag{7}$$

$$T_3 = N_v \times C_v + N_p \times C_p + N_u \times C_u + C_o \tag{8}$$

where, T_1, T_2 and T_3 denote the total cost function of three different cost evaluating method for collision detection, respectively. N_v is the number of BV pairs in the overlap test of BV pairs, C_v is the cost of testing a pair of BVs for overlap, N_u is the number of the primitive pairs

interference tests, C_p is the cost of testing a pair of primitives for interference, N_u is the number of nodes updating in the collision detection, C_u is the cost of updating each node, C_o is the cost of the transformation between the coordinates of the models:

$$T = N_t \times C_t \times + N_u \times C_u + N_v \times C_v + N_p \times C_p \quad (9)$$

where, T is the total cost function for collision detection of HBVHT, N_t is the number of pairs of broad phase BVs overlap test, C_t is the cost of testing a pair of BVs of broad phase for overlap, N_u is the number of broad phase BV updating, C_u is the cost of broad phase BV updating, N_v is the number of pairs of narrow phase BVs overlap test, C_v is the cost of testing a pair of BVs of narrow phase for overlap, N_p and C_p are defined as Eq. 6.

RESULTS AND DISCUSSION

We implemented the HBVHT collision detection for articulated models and used C++ for programming. The computer runs a Windows XP operating system with an intel (R) Core (TM) i5-2300 2.8 GHz CPU and 4GB main memory. OpenGL was used for graphic processing. The research object is an industrial robot TA1400. It is a multi-link serial connection articulated model which has seven joints with six degrees of freedom. A workstation including two robots with public workspace was designed. The bottom planes of the two robots are placed parallelly and the distance between them is 2400 mm. The total number of triangles in a single robot is almost 11K. The number of triangles in the each link is shown in the Table 1.We used the construction method described in section 2 to build the HBVHT for the collision detection of the two robots and apply the cost function Eq. 9 to analyze the efficiencies of collision detection.

Results: The structures of broad phase of the HBVHTs for a robot are shown in Fig. 4. In Fig. 4, it is an AABB MLHT structure (four levels) based on LSS building primitives.

The HBVHT is performed for collision detection with two robots as shown in Fig. 5, the joint 4 of robot 1 collides with joint 4 of robot 2 at that time.

We also use the classic algorithm RAPID which is based on OBB and compare it with the HBVHT for robots collision detection. It has been found that the numbers of the triangle intersecting pairs detected by the HBVHT are the same as that by RAPID. They are both 6172 pairs in the same location of the robots. These results indicate that the proposed HBVHT collision detection algorithm can achieve accurate collision detection and the structure of HBVHT is valid.

The broad phase of the HBVHT is updated in the light of the position and orientation of robot joint model on the next interpolation point when collision detection

Table 1: No. of triangles in the each link

	Foundation bed	Link 1	Link 2	Link 3	Link 4	Link 5	Link 6
No.	1014	3326	726	2292	1344	1158	956

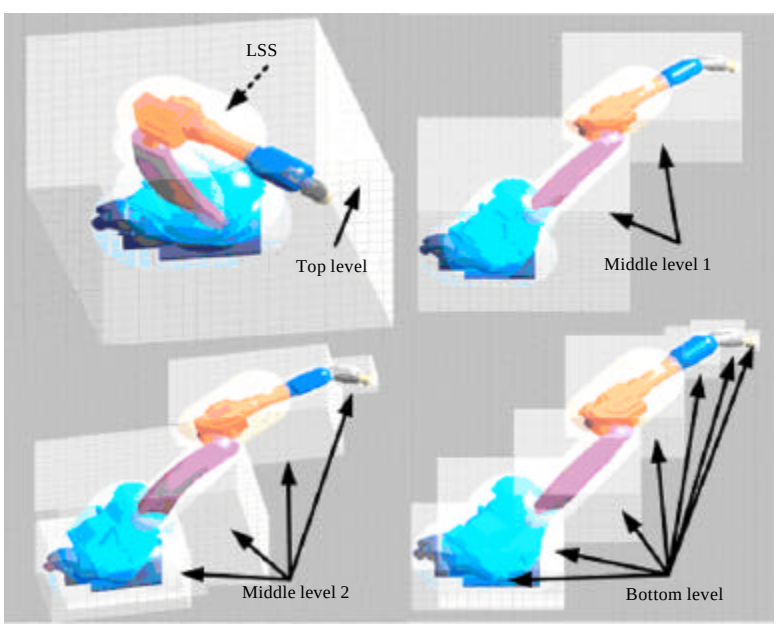

Fig. 4: Multi-level hierarchical tree (MLHT) structures of AABB in the broad phase

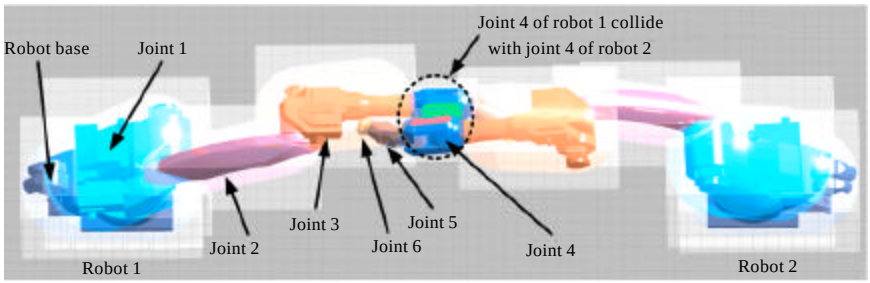

Fig. 5: Using HBVHT for collision detection between two robots

Table 2: Average time consuming of each interpolation point for 4 algorithms

	AABB (LSS)	AABB (OBB)	Sphere (OBB)	RAPID
Average time-consuming (μs)	4.611	9.069	6.358	53.856

AABB: Axis aligned bounding box, OBB: Oriented bounding box, SSV: Swept sphere volume, LSS: Line swept sphere, K-dops: Discrete orientation polytopes, BVHT: Bounding volume hierarchy tree, HBVHT: Hybrid bounding volume hierarchy tree, BV: Bounding volume, MLHT: Multi level hierarchy tree

of an interpolation point is finished. Collision detection continues till the last interpolation point of the trajectory. Two robots in the workstation move along the same trajectory composed by 5032 interpolation points. Moreover, the same triangle intersection test algorithm (projection method) is used to perform the triangle intersection test in the narrow phase for the HBVHTs and the RAPID.

The broad phase of our HBVHT is built with the AABB based on LSS, and then the experiment is performed to compare our algorithm with other three algorithms. They are the HBVHT whose broad phase built with the AABB based on OBB, or with the Sphere based on OBB and RAPID. The experiment results of collision detection are shown in the Table 2.

Discussion: From the experiment results we can know that the efficiency of our HBVHT is higher than that of other three algorithms. The reasons mainly include: (1) Considering the total amount of calculation for building the Bvs, our HBVHT is less than the others, (2) Considering the tightness of BVs, the total volume of our HBVHT is less than that of the others and (3) Considering the updating speed of BVs, The update cost of our HBVHT is the least as well.

These experiment results imply that: Firstly, the efficiency of collision detection is related to the structure of HBVHTs. Secondly, the AABB MLHT based on LSS has the highest efficiency. Finally, the proposed HBVHT has been proved to be better than RAPID in term of efficiency.

However, the HBVHT algorithm still has some limitations. If intensive collisions occured among the articulation models, the increase of efficiency with HBVHT will be found limited. Despite this, this study obviously indicates that the HBVHT is a good algorithm to improve efficiency for articulated models. The structure of HBVHT will be very useful to distribute the task to make good load balance for parallel computing, which can increase the efficiency substantially.

CONCLUSION

To speed up the efficiency of collision detection for articulated models, we have proposed a new algorithm of HBVHT and proposed a new cost function for analyzing the performance of the HBVHT algorithm. Experiments have been performed between two 6-DOF robots with OpenGL. The results show that the HBVHT could effectively improve the efficiency of collision detection among articulated models.

Based on this research, further work could be of interest and attractive in the areas of: (a) Applying HBVHT to proximity query among articulated models and (b) Combining HBVHT and parallel computing and explore the multi-core CPU architecture for parallel computing to improve the efficiency of collision detection.

ACKNOWLEDGMENTS

This study was supported by the National Natural Science Foundation of China (Grant No. 51065005), Natural Science Foundation of Guangxi (2012GXNSFAA053200); The Guangxi Key Laboratory of Manufacturing System and Advanced Manufacturing Technology (13-051-09S13, 11-031-12S03).

REFERENCES

Avril, Q., V. Gouranton and B. Arnaldi, 2012. Fast Collision Culling in Large-Scale Environments using GPU Mapping Function. In: Eurographics Symposium on Parallel Graphics and Visualization, Childs, H. and T. Kuhlen (Eds.). Eurographics Association, France, pp: 71-80.

Bradshaw, G. and C. O'Sullivan, 2002. Sphere-tree construction using dynamic medial axis approximation. Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation, July 21-22, 2002, San Antonio, Texas, USA, pp: 33-40.

Chang, J.W., W. Wang and M.S Kim, 2010. Efficient collision detection using a dual obb-sphere bounding volume hierarchy. Computer-Aided Design, 42: 50-57.

Ericson, C., 2005. Real-Time Collision Detection, Morgan Kaufmann Series in Interactive 3D Technology. Morgan Kaufmann, Philadelphia, PA.

Gottschalk, S., M. Lin and D. Manocha, 1996. OBB-tree: A hierarchical structure for rapid interference detection. Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, August 4-9, 1996, New Orleans, LA., USA., pp: 171-180.

He, T., 1999. Fast collision detection using QuOSPO trees. Proceedings of the 1999 symposium on Interactive 3D graphics, April 26-29, 1999, Atlanta, GA, USA, pp: 55-62.

Klosowski, J.T., M. Held, J.S.B. Mitchell, H. Sowizral and K. Zikan, 1998. Efficient collision detection using bounding volume hierarchies of k-dops. IEEE Trans. Vis. Comput. Graph., 4: 21-36.

Larsen, E., S. Gottschalk, M.C. Lin and D. Manocha, 1999. Fast proximity queries with swept sphere volumes. Technical report TR99-018, Department of Computer Science, UNC Chapel Hill. http://pdf.aminer.org/000/353/237/fast_distance_queries_with_rectangular_swept_sphere_volumes.pdf

Mendoza, C. and C. O'Sullivan, 2006. Interruptible collision detection for deformable objects. Comput. Graph., 30: 432-438.

Rashid, A.T., A.A. Ali, M. Frasca and L. Fortuna, 2012. Multi-robot collision-free navigation based on reciprocal orientation. Robotics Autonom. Syst., 60: 1221-1230.

Redon, S., Y.J. Kim, M.C. Lin and D. Manocha, 2005. Fast continuous collision detection for articulated models. J. Comput. Inform. Sci. Eng., 5: 126-137.

Terdiman, P., 2001. Memory-optimized bounding-volume hierarchies. http://www.codercorner.com/Opcode.pdf

Van den Bergen, G., 1997. Efficient collision detection of complex deformable models using AABB trees. J. Graph. Tools, 2: 1-14.

Weghorst, H., G. Hooper and D. Greenberg, 1984. Improved computational methods for ray tracing. ACM Trans. Graph., 3: 52-69.

Yoon, S., Y.J. Kim and T. Harada, 2010. Recent advances in real-time collision and proximity computations for games and simulations. http://www.siggraph.org/asia2010/content/attendees/Courses/18

Zhang, X., S. Redon, M.Y. Lee and Y.J. Kim, 2007. Continuous collision detection for articulated models using taylor models and temporal culling. ACM Trans. Graph., Vol. 26. 10.1145/1275808.1276396

Zhang, X.Y. and Y.J. Kim, 2007. Interactive collision detection for deformable models using streaming AABBs. IEEE Trans. Vis. Comput. Graph., 13: 318-329.