2010 8th IEEE International Conference on
Control and Automation
Xiamen, China, June 9-11, 2010

Efficient Collision Detection Based on AABB Trees and Sort Algorithm

Yi-Si Xing, Xiaoping P. Liu, Shao-Ping Xu

Abstract –Efficient collision detection is a fundamental and very challenging problem in real-time surgery simulation. This paper presents a novel collision detection algorithm to detect collision and self-collision, between complex models undergoing rigid motion and deformation. The algorithm relies on a hierarchical model representation using axis-aligned bounding boxes (AABBs) and sort algorithm. Two main advantages of this algorithm is showed: (a) apply layer-by-layer intersection test based on AABB tree rather than traversing binary tree. (b) apply lots of AABBs sorting instead of one to one test. This algorithm reduces the time complexity of collision detection performed.

Keywords- AABB tree, collision detection

I. INTRODUCTION

Collision detection has received much attention in robotics, computational geometry, and computer graphics. Some researchers have investigated the problem of interference detection as a mechanism for indicating whether object configurations are valid or not. Others have tackled the problems of computing separation or penetration distances, with the objective of applying collision response in simulated environments.

Collision detection has been thoroughly studied in fields like computer graphics [1-4,8], robotics [5], computational geometry [6,7] , computational mechanics and can be considered as one of the robotics fundamental pillars. In animation, whenever some objects representing a scene are provided of movement, it is necessary to determine if the mobile objects collide with others. In robotics or path planning this information is used to avoid collisions. In other applications the aim can be the opposite, being necessary to know the contact surface in order to simulate objects deformations or exit trajectories. The results are widely applied to virtual reality, computer-aided design and physical simulation in general.

However, in many applications collision detection is considered to be a major computational bottleneck. Bounding volume hierarchies(BVHs) have been widely adopted in these relevant areas, e.g., BVHs of spheres, axis-aligned bounding boxes (AABBs), oriented bounding boxes, discrete orientation polytopes (k-DOPs), and convex pieces, organize the objects in a hierarchical (tree) manner, and these methods differ by the type of bounding volume at each level of the tree or by adopting different techniques to build, update and balance the tree. The idea behind the approaches using a hierarchy of bounding volumes is to approximate the objects (with bounding volumes) or to decompose the space they occupy (using decompositions), to reduce the number of pairs of objects or primitives that need to be checked for contact. Two main advantages of these approaches must be highlighted: (a) in many cases an interference or a non-interference situation can be easily detected at the first levels in the hierarchy, and (b) the refinement of the representation is only necessary in the parts where collision may occur. Space and object partitioning representations for collision detection are surveyed next[12].

In this paper, we present a collision detection scheme that relies on a hierarchical model representation using axis-aligned bounding boxes (AABBs). In the AABB trees as we use them, the boxes are aligned to the axes of the model's local coordinate system, thus, all the boxes in a tree have the same orientation. The AABB bounding box of an object is defined as the smallest hexahedron which contains the collision object and the edges parallel to the axis. Therefore, there is only need six float value to describe an AABB. When constructing the AABBs, it needs to go along the axial (X, Y, Z) of the objects local coordinate system. Therefore, there is only need six float value to describe an AABB. Because its construction difficulty is low and its storage space is small enough , the AABBs algorithm is widely used. However, existing methods mostly applied the intersection between AABBs and double-traverse AABB tree to carry out collision detection, it is difficult to reduce the time complexity. In this paper, we use segment sort algorithm based AABB trees instead of traversing binary tree to speed up overlap testing between relatively oriented boxes of a pair of AABB trees.

The authors are with School of Mechatronics Engineering and School of Information Engineering, NanChang University, PR China(e-mail: linyanhu78@163.com). X. P. Liu is with the Department of Systems and Computer Engineering, Carleton University, Canada. This work was partially supported by National Natural Science Foundation of China (No.: 6087020), the Department of Education of Jiangxi Province under the grant (No.:GJJ09015) and the Department of Science and Technology of Jiangxi Province under the grant (No.: CB200920364).

978-1-4244-5196-8/10/$26.00 ©2010 IEEE

II. AABB Hierarchical Volume Bounding tree

The BVH is a tree constructed with bounding volumes. Each node of BVH is a volume that bounds the corresponding set of primitives. The leaf nodes of BVH generally bound single primitives. An AABB tree is constructed top-down, by recursive subdivision. At each recursion step, the smallest AABB of the set of primitives is computed, and the set is split by ordering the primitives with respect to a well-chosen partitioning plane. This process continues until each subset contains one element. Thus, an AABB tree for a set of n primitives has n leaves and n - 1 internal nodes.

At each step, we choose the partitioning plane orthogonal to the longest axis of the AABB. We position the partitioning plane along the longest axis, by choosing β, the coordinate on the longest axis where the partitioning plane intersects the axis. We then split the set of primitives into a negative and positive subset corresponding to the respective half spaces of the plane. A primitive is classified as positive if the midpoint of its projection onto the axis is greater than β, and negative otherwise. Fig.1 shows a primitive that straddles the partitioning plane depicted by a dashed line. This primitive is classified as positive, because its midpoint on the coordinate axis is greater than β.It can be seen that by using this subdivision method, the degree of overlap between the AABBs of the two subsets is kept small.

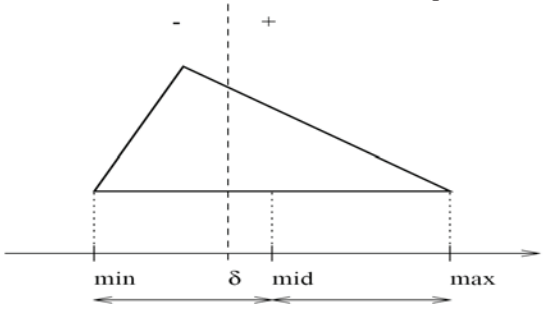

Fig. 1 A primitive is split by plane β

For choosing the partitioning coordinate β we tried the best performance is achieved by simply choosing β to be the median of the AABB, thus splitting the box in two equal halves. Using this heuristic, it may take $O(n^2)$ time in the worst case to build an AABB tree for n primitives, however, in the usual case where the primitives are distributed more or less uniformly over the box, building an AABB tree takes only $O(n \log n)$ time[11].

Occasionally, it may occur that all primitives are classified to the same side of the plane. This will happen most frequently when the set of primitives contains only a few elements. In this case, we simply split the set in two subsets of (almost) equal size, disregarding the geometric location of the primitives. Take a sample example showed in Fig.2. A bounding volume hierarchy of five simple objects. Here the bounding volumes used are AABBs.

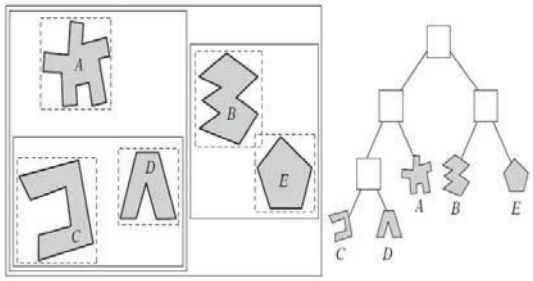

Fig. 2 AABBs bounding volume hierarchy of five simple objects

Several desired properties for bounding volume hierarchies have been suggested:

- The nodes contained in any given sub-tree should be near each other. Without explicitly defining nearness, the lower down the tree the nodes are the nearer they should be to each other.
- Each node in the hierarchy should be of minimal volume.
- The sum of all bounding volumes should be minimal.
- Greater attention should be paid to nodes near the root of the hierarchy. Pruning a node near the root of the tree removes more objects from further consideration than removal of a deeper node would.
- The volume of overlap of sibling nodes should be minimal.
- The hierarchy should be balanced with respect to both its node structure and its content. Balancing allows as much of the hierarchy as possible to be pruned whenever a branch is not traversed into.

III. Intersection Testing by Segment Sort Algorithm

In recent work, the test between models is done by recursively testing pairs of nodes. The time of traversing binary tree has become the main bottleneck of improve efficiency. In this paper , Segment Sort Algorithm is used to test AABB against AABB rather than traversing binary tree. This algorithm is based on different ways to incrementally update the efficiency during simulation by exploiting temporal coherence.

The main feature of the algorithm is the full exploitation of the temporal coherence of the objects exhibited in a dynamic environment. In the algorithm, the AABBs are projected to each Cartesian axis. The projected intervals on the axes are separately sorted by the diminishing increment sort (DIS) and further divided into subsections. By processing all the intervals within the subsections to check if they overlap, a complete contact list can be built. The only assumption made is that the sorted list at one time step will remain an almost sorted list at the next time step, which is valid for most

applications whose movement and deformation of each AABB and the dynamic change of the total number N (the total number of objects) are approximately continuous[9].

A. *Temporal coherence*

Temporal coherence is the property that some features of an application do not change significantly between two consecutive time steps. The objects move and deform only slightly from last time step to the current time step and, in the meantime, the change of the total number of objects is within an acceptable level. Supposed the time interval of sampling points is small enough, the object has not undergo a big move, and (or) in the 2 sampling points the total number of the objects has not experienced a large increase or decrease ,thus the applications features in the two consecutive sampling points would not be changed significantly.

Such a temporal coherence is present in most dynamical applications. However, a traditional collision detection algorithm always consumes the same CPU resource every time step even if nothing in the environment has changed. This paper addresses this issue by utilizing temporal coherence information in order to speed up collision detection. Temporal coherence makes the collision detection on the adjacent sampling points great similar, so we could use this feature to improve the efficiency of collision detection greatly. In the virtual reality technology in order to achieve the purpose of real-time interaction, the time interval of sampling points is very small (to ensure the vision natural flow , in generally, refresh rate is no less than 20fvs), complete satisfy the application premise of the temporal coherence.

B. *Segment Sort Algorithm*

Two AABBs intersect each other if, and only if, their orthogonal projections onto the axes of the coordinate system are all overlapping. This means, that the problem can be reduced to one dimension since the overlapping intervals can be obtained from the ordered lists after sorting the AABBs' one-dimensional projections. This observation further suggests a special sorting algorithm that may be effective in handling this situation: the diminishing increment sort (DIS). It proposed by Donald L. Shell in 1959 improves on Insertion Sort by reducing the number of inversions and comparisons made on the elements to be sorted. It sorts an array A with n elements by dividing it into subsequences and sorts the subsequences. As an extension of the insertion sort, DIS can achieve a O(N) speed for an almost ordered input, and also behave much more robust than the quick sort or the heap sort[10]. The concept of temporal coherence has been introduced to speed up the sorting process of AABBs on the system axis. The sorting sequence of the current sampling points is regarded as a ordered sequence for the next sampling point. Therefore, DIS algorithm is used to sort the AABBs' projection value on the three coordinate axis. During the sorting process, the projection list of the current time sampling point would be saved so that there is only need to modify the projection value of moving objects on the next sampling point. Meanwhile, there is also need to track and record the overlapping states of the projection intervals so as to determine whether the AABBs intersect.

Similarly to other mechanical phenomena such as deformation and movement, collision is also a local behavior, i.e., an object can only collide with objects in its proximity and is hardly affected by objects far away from where the object is. In Fig.3, object B is non-local to object A and therefore should not be considered when dealing with the collision of the latter. However, if all the projections are sorted together, objects A and B will overlap on the x-axis and therefore inevitably be recorded once. With the objects increased, this makes the overlap checking procedure increasingly more expensive than the sorting procedure.

In order to overcome this, during the sorting procedure each axis is cut into a series of segments containing the (nearly) same number of projection intervals. If the projection intervals could not be cut into equivalent partition ,an equal number of items is arranged for all the sub-lists except for the last one . For example, if there are 4 sub-lists and 110 items all together, the first 3 sub-lists will each have 30 items and the forth sub-list will have 20 items. In general, the more segments an axis is cut into, the faster the algorithm runs. However, an axis should not be cut unlimitedly, and the limit is that an AABB must not be cut more than once on any direction.

Following the axial cut the objects are divided into a series of subsets which are subject to the overlap checking. As shown in Fig. 3, the x and y-axes are both cut into two segment, and the corresponding division is marked by dashed lines. Since overlap checking is limited within each subset, objects A and B, which are in different subsets, will not interfere with each other. So the number of the objects which should be detected was reduced greatly.

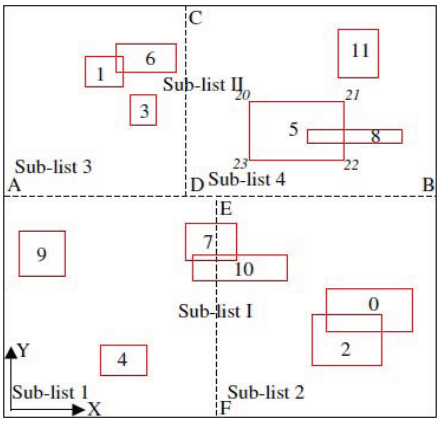

Fig. 3 Space locality and axial cut (2D).

For the sake of simplicity, the 2D case shown in Fig.3 is taken as an example to explain the SMB algorithm in detail. Twelve AABBs are represented by their 48 vertices. For example, AABB 5 is represented by vertices 20, 21, 22 and 23. In general, AABB n has the corresponding vertices 4n, 4n + 1,4n + 2, and 4n + 3 respectively and, vice versa. Also, if a sorted list at one time step remains as an almost ordered list at the next time step, which is often held by a system with temporal coherence, the SMB will become a linear algorithm whose computational cost is only proportional to N[9].

IV. Implementation and Analysis

The initial state of the projection list is based on the number of the orthogonal vertices gained from the object sequence. Starting from the root node, the every layer AABBs of the binary tree would be sort with the DIS in order to determine the collision pairs. For example, if the root nodes 0 and a are regarded as contact pairs after DIS sorting, then the next layer nodes 1, 2, b and c should be further detected. If the result shows the collision pairs are nodes 2 and b, the detection for the next layer should be go on until going to the leaves nodes. At last, the determined collision pairs are leaves nodes 6 and d which contain the only one basic element, then the more exact detection between the basic elements would be used to determine the precise collision position.

In this virtual environment, an assumption made is that only consider the collision between two objects except for multi-object collision. Set an overlapping state for every AABB bounding box, the state formed by the three Boolean variables, respectively, stands for corresponding axis(x, y in 2D system; x, y, z in 3D system). When the three Boolean variables are all set to true value, it shows that there are an collision pairs, and then put this into the overlapping list.

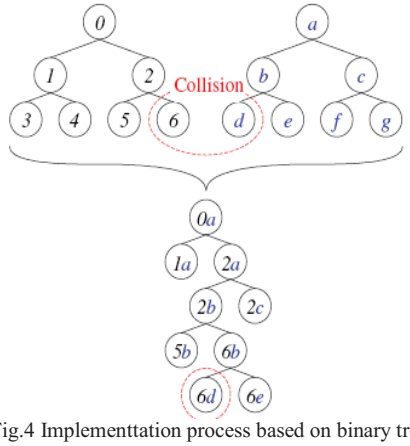

Fig.4 Implementtation process based on binary tree.

For the sake of simplicity, the 2D case showed in Fig. 4 is taken as an example to show the implementation process.

*1)*Building an AABB tree, sort the every layer AABBs;

*2)*Divided the y-axis and get the projection intervals sub-sequence;

*3)*Perform DIS on the sub-sequences of y-axis ;

*4)*Get the sequence list y1(the collision sequence list on y-axis);

*5)*Divide the x-axis, perform DIS on x-axis sub-sequens and get the list X1(the collision sequence list on x-axis);

*6)*Regroup list Y1 ad Y2 according to X1;

*7)*Loop over each sub-list of X1,Y2 and do
Check overlapping intervals in list X1;
Check overlapping intervals in list Y2 and report contact pairs;

The time cost of the sort algorithm in this paper could be obtained by adding up the costs of all the operations from step 1 to step 7. In the algorithm, steps 2, 3, 4 and 5 are linear operation, and the cost of each is approximately equivalent to one loop over a list. There are only one non-linear operations in the algorithm, i.e. step 7, but because of the axial cut their costs are approximately proportional to N as well. Hence, the performance of the algorithm in terms of CPU time is a function of $O(n)$, where N is the total number of objects. While, in previous algorithm when N objects need to be detected, the time complexity would be $O(n^2)$. This algorithm presented in this paper reduces the time complexity of collision detection. The complexity of collision detection time should ideally be of order $O(n)$, regardless of the AABB sizes, the packing density and the space size.

I-Collide algorithm is the open source database based on the one-dimensional space sorting method of the AABB bounding box. We have made some changes to this database, using axes splitting strategy, and sort the projection list with the algorithm in this paper. The experiment results are shown in Fig. 5.

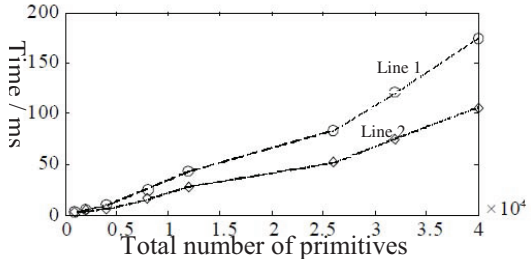

Fig.5 Compare result between two algorithms

Here we set some specific scene, all the privatives projections are artificial given. Then we compare the two algorithms in the same scene. In Fig. 5 the line 1 represents the improved algorithm; the line 2 represents

the traditional algorithms. As the total number's increase of the primitives, the improved algorithm in this paper is superior to the traditional algorithm .

V. CONCLUSIONS

In this paper, temporal coherence is fully used to improve the efficiency of collision detection. A novel algorithm based on this feature for multiple moving AABBs is presented. Different to most other collision detection algorithms, such as those geometric based, bounding volume hierarchy (BVH) and spatial subdivision, the algorithm presented here is based on a sequencing model. The performance of the algorithm is comprehensively investigated through theoretical analysis. The time complexity of collision is reduced.

REFERENCES

[1] S. Gottschalk, M.C. Lin, D. Manocha, "OBB: a hierarchical structure for rapid interference detection," in: Proc. ACMSIGGRAPH, 1996, pp. 171–181.

[2] A. Wilson, E. Larsen, D. Manocha, M.C. Lin, "Partitioning and handling massive models for interactive collision detection," Comput. Graph. Forum (Proc. Eurographics) 18 (3) (1999), p. C319-+ Sp.

[3] D.J. Kim, L.J. Guibas, S.Y. Shin, "Fast collision detection among multiple moving spheres," IEEE Trans. Visual. Comput. Graph. 4 (3) (1998) 230–242.

[4] S. Redon, A. Kheddar, S. Coquillart, "Fast continuous collision detection between rigid bodies," Comput. Graph. Forum 21 (3) (2002), p. 279-+ Sp.

[5] J. Canny, "Collision detection for moving polyhedra," IEEE Trans. Pattern Anal. Mach. Intell. 8 (2) (1986) 200–209.

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

[7] H. Edelsbrunner, "A new approach to rectangle intersections," Int. J. Comput. Math. 13 (3–4) (1983) 209–219.

[8] J. Cohen, M. Lin, D. Manocha, K. Ponamgi, "I-COLLIDE: an interactive and exact collision detection system for large-scaled environments," in: Proc. ACM Symposium on Interactive 3D Graphics, 1995, pp. 189–196.

[9] C.ELi，YT.Feng，D.R.J.owen, "SMB:Collision de1ect1on basedon temporal coherence," Computer methods in applied mechanics and engineering， 2006，195: 2252 ~ 2269

[10]Oyelami, M.O.," A Modified Diminishing Increment Sort for Overcoming the Search for Best Sequence of Increment for Shellsort," Journal of Applied Sciences Research, 4(6): 760-766, 2008

[11]Gino van den Bergen, "Efficient collision detection of complex deformable models using AABB trees," Journal of Graphics Tools, Volume 2 , Issue 4: 1 - 13 , April 1997

[12]P. JimeH nez. F. Thomas, C. Torras, "3D collision detection: a survey," Computers &Graphics ,25 (2001) 269}285