Research on Collision Detection Algorithm Based on AABB

WANG Xiao-rong WANG Meng Li Chun-gui
Department of Computer Engineering, GuangXi University of Technology

Abstract

An improved collision detection algorithm based on AABB is presented. During the global search, each axis is cut into a series of segments containing the same number of AABBs' projection intervals, and Shell sort is adopted to sort projection lists, not insertion sort.This will avoid needless intersecting test of AABB. During the local detection, the amount of byte of AABB bounding-volume for internal node is reduced according to the constructing process of AABB tree, and leaf nodes are wiped from tree structure The storage of AABB tree is compressed. This method can save a large amount of space and speed up the algorithm. Experiments indicate that the improved algorithm reduce detection time for the same models.

1. Introduction

Collision detection is a fundamental problem in a wide range of fields. Various collision detection applications are found in robotics, computer graphics, and 3D computer games[1]. The aim of collision detection is to automatically report one or more geometric contacts which are about to occur or have actually taken place between objects. Ensuring that objects interact in the correct manner is very computationally intensive and much research has addressed the issues involved with trying to reduce the computational requirements.

All major collision detection algorithms can be classified into three categories: (i) geometry, (ii) bounding volume hierarchy (BVH) and (iii) spatial subdivision. The geometric approach [2] is to analyse the topology of each separate model, track and then capture the closest features between two or more models. BVH-based methods 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. Spatial subdivision [3,4] method is first divided into equal cells, and at every instant the objects are assigned to one or more cells. Collisions are then checked between all object pairs belonging to a series of related cells. Since each space cell may need a storage location in the computer memory, the total memory requirement and the CPU time will inevitably be influenced by the overall space size.

Bounding-Volume hierarchies algorithm is the most common collision detection algorithm. The algorithm usually includes two steps: a global search and a local detection. By approximating each object with a simple bounding volume, the global search aims to reduce the O(N2) running time needed to perform intersection tests on all possible contact pairs of the N objects; the local detection then undertakes more detailed intersection calculations only for those potential contact pairs obtained in the global search. The method speeds up the collision detection process because the intersection test between bounding volumes is easier than between objects. In many applications collision detection is considered to be a major computational bottleneck. A lot of papers have focused on the efficiency of collision queries: various ways of building the tree have been explored, experiments have been made with hybrid hierarchies, and almost the whole BV-spectrum has been implemented in at least one available collision detection library.

Recently, more and more researchers use bounding volume hierarchy to resolve collision detection problem. Bin Yang using bounding volume hierarchy defined in the context of a binary tree, is built for each contact surface based on the geometry of the surface. A global contact searching procedure based on these bounding volume trees is first performed to find all candidate contact element pairs, and then a local searching procedure is done to find all the mortar segments having contributions to the mortar integrals that define the contact formulation[6]. Jung-Woo Chang performs collision detection between static rigid objects using a bounding volume hierarchy which consists of an oriented bounding box tree enhanced with bounding spheres [7]. Pan [8] adopts memory compression technology to improve bounding volume AABB tree for reducing memory requirements.

The rest of the paper is organised as follows. Section 2 describes the improved collision detection algorithm in the global search, including axial cut, Shell sort. Section 3 describes the compressed storage of AABB tree. Section 4 gives the results of experiments and analysis. Finally conclusions are drawn in Section 5.

2. Improvement in the global search

During the global search, each object is described with a simple AABB (axis-aligned bounding box). The method speeds up the collision detection process because the intersection test between bounding volumes is easier than between objects. 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. The tradition collision detection algorithm adopts insertion sort for AABBs' one-dimensional projections lists. This paper will adopt Shell sort and improves the procedure of sort.

Collision is a local behaviour, 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. So, during the sorting procedure each axis is cut into a series of segments containing the (nearly) same number of projection intervals, and following the axial cut the objects are divided into a series of subsets which are subject to the overlap checking. Overlap checking is limited within each subset, and objects which are in different subsets,will not interfere with each other. In general, the more segments an axis is cut into, the faster the algorithm runs. But, it is not necessary to cut an axis into tiny segments to achieve good performance. An axis should not be cut unlimitedly, and the limit is that an AABB must not be cut more than once on any direction. If a few objects are very large, a pre-processing procedure can be applied to divide each large object into several smaller objects before the contact detection takes place [9].

The above axial cut method can be regarded as an equivalent partition strategy, but is different from space subdivision. A traditional space subdivision method divides a simulation space into cells of equalsize, however there is no best size for all situations. The axial cut method consider for objects, not for space.

3. Improvement in the local detection

3.1 Construction of AABB tree

This paper constructs a balanced AABB tree. This can reduce the time of search of AABB tree. The algorithm is represented as follow:

(1) Computing representation point of each element according to coordinate-value of basic geometrical elements contained of root node V;
(2) Computing AABB bounding volume of root node V;
(3) Along the longest axis of the current AABB， partition it into two subsets in term of cancroids of triangles;
(4) Regard two subsets as root node respectively, go to (2), until each AABB of basic geometrical element is leaf node.

We can obtain an AABB tree that is a complete binary tree, namely each leaf node only contain one triangle.

3.2 Wipe off leaf nodes

Pan [8] adopts memory compression technology to improve bounding volume AABB tree for reducing memory requirements. The paper presents improved method based on Pan's algorithm. We optimize storage of leaf node based on two fast algorithm [10][11] about triangle presented by *Möller* ,which make computing time adopting triangles as the same as adopting bounding box.

A canonical internal node of AABB tree contains an AABB bounding volume, an index to the left child and an index to the right child. Each value of node needs a float data (4 bytes), therefore a node requires 24 bytes to store. To advance the efficiency of queries, each internal node has two indexes to left child and right child, which needs 4 bytes. Therefore, an internal node needs 32 bytes to store. A canonical leaf node structure contains one AABB bounding volume and a pointer/index to the surrounded primitive [5]. During the intersection test of object A and object B, if the object of test includes a leaf node, we can directly adopt triangle to test. So we can get rid of the AABB bounding volume in leaf node. Only the index to triangle is remained in leaf nodes. We can get rid of them as well, by storing them in parent nodes, replacing previous index to the leaf nodes we just discarded. This method does not need more additional memory space. The structure of parent node is showed as table 1. In table 1, the values of last two bits are 1 which indicates its children nodes are leaf node.

Table 1. Parent node of leaf node structure

Scope-value ($4 \times 3 = 12$ bytes)		
Left-child node index (4 bytes)	Right-child node index (4 bytes)	symbol (1 byte)

4. Experiments and Results analysis

We detect collision for different models by adopting traditional algorithm and improved algorithm. The result is showed in figure 1 and figure 2.

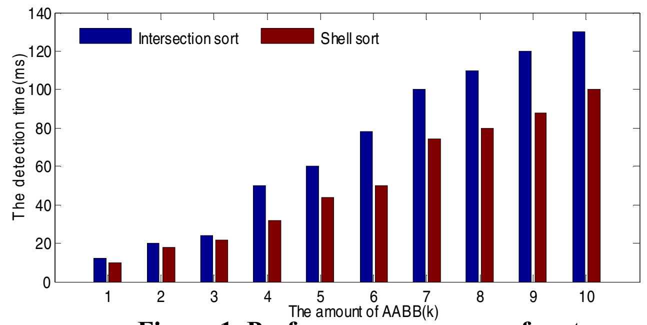

Figure 1. Performance compare of sort

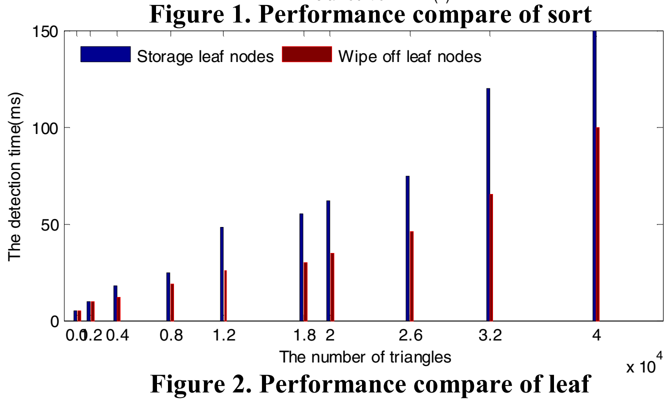

Figure 2. Performance compare of leaf node compressed storage

As showed in fig.1, improved sort method can reduce comptuing time of intersection test of AABBs'. The objects move 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. So, for most cases, these projection sorted lists of AABB would posses a temporal coherence, i.e. they will change only slightly between time steps. Shell sort can achieve a O(N) speed for an almost ordered input. So this method speeds up the detection during global search.As showed in fig.2, we compressed storage of AABB tree. This not only reduces the memory requirements, but also speeds up the algorithm. We construct balanced AABB tree, which saved search time for tree.

5. Conclusions

This paper speeds up the traditional collision detection algorithm based on AABB box. During the global search, we adopt Shell sort to sort projection lists of AABB on three axes, and cut axis into segments. This improve can limit detect within subset, and reduce the amount of AABB box detected. During the search of AABB tree , we compress the storage of AABB's nodes, not only internal nodes ,but also leaf nodes. This reduces the memory requirement, and speeds up the algorithm.

6. Acknowledgement

This paper is Supported by the Educational Commission of GuangXi Province of China (200808LX341, 200808LX338), Master Degree Foundation of Guang Xi University of Technology (0816219).

7. References

[1] Dan Albocher, Uzi Sarel, Yi-King Choi etc, Efficient Continuous Collision Detection for Bounding Boxes under Rational Motion, Proceedings of the 2006 IEEE International Conference on Robotics and Automation, Orlando, Florida - May 2006.

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

[3] A. Munjiza, K.R.F. Andrews, NBS contact detection algorithm for bodies of similar size, Int. J. Numer. Methods Engrg. 43 (1) (1998) 131－149.

[4] J.R. Williams, E. Perkins, B. Cook, A contact algorithm for partitioning N arbitrary sized objects, Engrg. Comput. 21 (2/3/4) (2004) 235－248.

[5] Pierre Terdiman, Memory-optimized bounding-volume hierarchies, 2001, http://www.codercorner.com/Opcode.htm.

[6] Bin Yang , Tod A. Laursen, A contact searching algorithm including bounding volume trees applied to finite sliding mortar formulations,Comput Mech (2008) 41:189–205

[7] Jung-Woo Chang, Wenping Wang and Myung-Soo Kim. Efficient Collision Detection Using a Dual Bounding Volume Hierarchy, LNCS, Advances in Geometric Modeling and Processing, Volume 4975/2008.

[8] PAN Zhen-Kuan, LI Jian-Bo, The Collision Detection Algorithm Based on Compressed AABB Trees， Computer Science, 2005, 33(2):213-215.

[9]C.F.Li, Y.T.Feng, D.R.J.Owen, SMB: Collision detection based on temporal coherence, Computer methods in applied mechanics and engineering, 2006, 195:2252~2269.

[10] Moller Tomas, A fast triangle-triangle intersection test,Journal of Graphics Tools,1997,2(2):25-30.

[11] Moller Tomas, Fast 3D Triangle-Box Overlap Testing[J], Journal of Graphics Tools ,2002,6(1):29-33