Research on hybrid bounding box interference check algorithm for robot offline simulation system

Jing Li1, Zhen Zhong1,4, ChangQi Fan2, NanYan Shen1,*, YiHao Lu1, ZeYuan Feng1, ZhenLiang Dai3 and Hong Yu1

1Shanghai Key Laboratory of Intelligent Manufacturing and Robotics, School of Mechatronic Engineering and Automation, Shanghai University, Shanghai 200444, China

2Shanghai Mobile Internet Industry Promotion Center, Room 614, 306 Taopu Road, Putuo District, Shanghai 200444, China

3Yanfeng Plastic Omnium Automotive Exterior Systems Company Limited, No.540 Moyu Road, Anting, Shanghai 200444, China

4Email: shuzon1996@163.com; *Corresponding author's Email: shny@shu.edu.cn

Abstract. Aiming at the problem that the traditional industrial robot may interfere with the workpiece when running in the offline simulation environment, a hybrid bounding box algorithm based on octree is proposed. First, AABB is used for rough interference checks, which quickly eliminates the pairs of impossible objects to intersect, then OBB hierarchical bounding box is used for precise interference checks. A new OBB construction method is proposed, which recalculates the semi-axis length and centre of the constructed traditional OBB to make it surround the object as much as possible and reduce the space gap. A traversal method based on distance priority is proposed, which traversed the closest node every time by maintaining a priority queue. And in the update strategy, "update on demand" is adopted, which discards updating the entire tree before each intersection and only updates the current node during the intersection traversal to achieve the purpose of partial update. Experiments show that the proposed hybrid bounding box algorithm can detect interference faster and perform better real-time performance and accuracy.

1. Introduction

Interference check algorithms have been widely used in various applications, including robotics, computer graphics, animation, computer games, virtual reality, and simulation. The simplest and most primitive interference check algorithm uses the brute force method, which checks the intersection of all triangles of the two object models. As the complexity of the model increases, although the brute force method can get the correct results, the time complexity is relatively high, generally \(O(N^2)\), which is challenging to use in actual production operations.

To improve detection efficiency, Francisco[1] proposed an Extended Bounding-Planes (EBP) octree algorithm to avoid calculating each triangle's normal and provide accurate intersection detection results. Stuevel[2] carried out discrete collision simulation for crowds and proposed a new bounding cylinder hierarchy (BCH) algorithm. Tang[3] takes advantage of AABB construction's simplicity, regards AABB as a special OBB, uses the rotation matrix to update, and uses SAT [4] during the intersection test. Huang[5] used the octree to divide the space and judged whether the number of point
sets in each octree node met the threshold, thereby establishing a tree structure. Although scholars have studied interference check from various aspects, it has improved interference check efficiency, but certain limitations still exist. The single bounding box method requires a trade-off between accuracy and real-time. Thus, based on previous algorithms, this paper presents an octree-based hybrid bounding box algorithm named AO0. AO0 aims at the industrial robot offline simulation system and solves the shortcomings of a single method in terms of accuracy and real-time.

2. Hybrid Bounding Box Algorithm Based on Octree

Traditional interference check methods between three-dimensional objects mostly perform the intersection operation between two entities: the intersection operation between all the primitives contained between the things. The surface of traditional industrial robots and curved workpieces is composed of many triangular facets. It is often time-consuming to directly conduct an intersection test on these thousands of triangular elements. Due to the particular geometric characteristics of complex surface or free-form surface workpiece, there will be a significant gap near the surface in three-dimensional space. Although the traditional bounding box algorithm can quickly eliminate the object pairs that are unlikely to interfere, the number of triangle face pairs that need to be detected cannot reduce in the precise detection stage. The motion simulation of industrial robots has a high demand for real-time performance, so an octree-based bounding box structure is used in this paper to solve this problem.

2.1. Algorithm Description

The octree-based hybrid bounding box algorithm named AO0 (AABB-OBB-Octree) proposed in this paper uses a two-layer structure. The outer AABB can eliminate the object pairs that are unlikely to interfere as early as possible. Simultaneously, the lower OBB hierarchical bounding box can perform an accurate interference check, essentially using a simple geometrically shaped bounding box to surround the object. Before carrying out the intersection test of triangles between objects with complex shapes, AO0 performs a simple intersection test between bounding boxes. Simultaneously, improvements were made in constructing the bounding box, the hierarchical tree structure, and the hierarchical tree's traversal and update. The algorithm flow chart of the mixed bounding box based on the octree is shown in figure 1 below.

![Algorithm flow chart](image-url)
2.2. Improvement of bounding box construction method

The centre position of the traditional OBB[6] bounding box is the mean value of the sum of the vertices of all the basic geometric primitives of the object. The basic geometric primitive of the object in this paper is a triangular face. The OBB bounding box's centre position is the average coordinate of all the triangular face vertices. However, this method is only a straightforward geometric mean and does not consider the deviation problem caused by the triangular faces' uneven distribution. The obtained bounding box locally generates gaps, as shown in figure 2(a) below.

An improved recalculation method is proposed in this article, which solves the traditional bounding box algorithm's inaccurate calculation results. This method's core idea is to take half the maximum and minimum absolute value as the semi-axis length. And calculate the new centre position based on the semi-axis size. The bounding box calculated in this way has relatively better tightness, as shown in figure 2(b).

![Figure 2. OBB bounding box before and after improvement](image)

The specific method steps are as follows:

Step1: Assuming that the primitive of the object is a triangular facet, the vertex coordinates of the i-th triangular element are respectively \(P_i, Q_i\) and \(R_i\). As shown in the following equations, the centre of the bounding box is \(c\).

\[
c = \frac{1}{3} \sum_{i=1}^{n} (P_i + Q_i + R_i)
\]  

Step2: The three direction axes \(o_x\), \(o_y\), and \(o_z\) of the OBB bounding box can be obtained by the principal component analysis method.

Step3: Calculate the maximum distance from the centre \(c\) to the vertices of all the triangle faces toward the direction axes \(o_x\), \(o_y\) and \(o_z\) respectively, then obtain \(minX, minY, minZ, maxX, maxY, maxZ\). By taking half of the sum of these maximum and minimum values, the OBB bounding box's semi-axis length in each direction axis can obtain.

Step4: Since the recalculated semi-axis length of the bounding box is obtained, it is necessary to calculate the recalculated centre position \(c_1\) of the bounding box. Because the semi-axis length of the OBB bounding box is known, the eight vertices of the bounding box can obtain, namely \(corner_{i}\) (i=1, 2, 3, 4, 5, 6, 7, 8). The coordinate values of the eight vertices divided equally to get the recalculation centre \(c_1\). As shown in the following equations.

\[
\begin{align*}
    corner_1 &= c + maxX \cdot o_x + minY \cdot o_y + minZ \cdot o_z \\
    corner_2 &= c + minX \cdot o_x + minY \cdot o_y + minZ \cdot o_z \\
    corner_3 &= c + minX \cdot o_x + maxY \cdot o_y + minZ \cdot o_z \\
    corner_4 &= c + maxX \cdot o_x + maxY \cdot o_y + minZ \cdot o_z \\
    corner_5 &= c + maxX \cdot o_x + minY \cdot o_y + maxZ \cdot o_z \\
    corner_6 &= c + minX \cdot o_x + minY \cdot o_y + maxZ \cdot o_z \\
    corner_7 &= c + minX \cdot o_x + maxY \cdot o_y + maxZ \cdot o_z \\
    corner_8 &= c + maxX \cdot o_x + maxY \cdot o_y + maxZ \cdot o_z \\

    c_1 &= \left( \sum_{i=1}^{n} corner_i \right) / 8 
\end{align*}
\]  

3
2.3. Improvement of hierarchy structure and traversal

Unlike traditional algorithms that use binary trees as the hierarchical tree structure, this article uses octrees as the tree structure, effectively reducing the tree's height. When the model is large, the child nodes that will not interfere can quickly eliminate. The tree structure is shown in figure 3 below.

![Hierarchical bounding box tree structure](image)

Figure 3. Hierarchical bounding box tree structure

The traditional hierarchical bounding box algorithm's update strategy is to update all the tree nodes before the intersection test. The AOO algorithm in this paper adopts an on-demand update strategy. The bounding box is updated when the nodes intersect, which avoids updating many bounding boxes that are unlikely to interfere and reduces the interference check time, as shown in figure 4 below.

![Schematic diagram of on-demand update strategy](image)

Figure 4. Schematic diagram of on-demand update strategy

This paper proposes a distance-first traversal method. When traversing the intersection of hierarchical trees, the distance between two tree nodes is first calculated. The intersection test is preferably performed on the two bounding boxes that are closer, which effectively speeds up the test speed.

Assuming that the root nodes of the object to be tested are \(a_i\) and \(b_i\), the specific steps of the distance-first traversal method are as follows:

Step 1. Judge whether the OBB of node \(a_i\) and \(b_i\) intersect, if they do intersect, go to Step2. Otherwise, node \(a_i\) and \(b_i\) do not intersect.

Step 2. If \(a_i\) is a leaf node, go to Step3. Otherwise, maintain a priority queue (small top heap), calculate the distance between the OBB centre of a child node of \(a_i\) and the OBB centre of \(b_i\), store it in the priority queue, and then take the top of the heap as a new \(a_i\) and go to Step1, finally pop the top of the heap, repeat the above operation until the priority queue is empty.

Step 3. If \(b_i\) is a leaf node, then the intersection occurs, enter the precise detection stage, and perform the triangle intersection detection. Otherwise, maintain a priority queue (small top heap), calculate the distance between the OBB centre of a child node of \(b_i\) and the OBB centre of \(a_i\), store it in the priority queue, and then take the top of the heap as a new \(b_i\) and go to Step1, finally pop the top of the heap, repeat the above operation until the priority queue is empty.
3. Results & Discussion
The experimental environment is CPU I5-4210H 2.9GHz, memory 8GB, graphics card NVIDIA GTX 860M, development language is C++, based on the OpenGL graphics library, and the object is composed of a set of triangular faces in obj format. Under the same test conditions, the algorithm's performance in this paper is compared with the literature [3] and literature [5], and the average interference check time is calculated. There are scenes 1-7 in total, and the number of triangles contained in the workpiece in each stage is different. The test object discussed in this article consists of two parts: the workpiece object and the tool object at the end of the robot arm, as shown in figure 5 below.

![Object to be tested](image)

In this paper, the AOO algorithm performs the update operation when the interference checks conduct, so the update time is included in the interference check time. The comparison results of different algorithms are shown in table 1 and figure 6 below. From table 1 and figure 6(a), the construction speed and interference check efficiency of the algorithm in this paper are higher than those in literature [3] and literature [5]. Figure 6(b) and figure 6(c) show that the average frame time of the algorithm in this paper is lower than that of other algorithms, which means that the entire simulation period's smoothness has improved. It can be seen from figure 6(d) that the operating memory efficiency has also improved.

![Comparison of interference check time](image)

| Scene | Algorithm[3] | Algorithm[5] | AOO algorithm |
|-------|--------------|--------------|---------------|
|       | Construction /ms | Interference /ms | Update /ms | Construction /ms | Interference /ms | Update /ms | Construction /ms | Interference /ms |
| 1     | 166.469      | 62.304       | 1.957       | 82.869        | 77.325        | 0.193       | 35.112        | 13.472         |
| 2     | 239.286      | 100.786      | 0.920       | 110.761       | 46.641        | 0.183       | 49.558        | 4.5264         |
| 3     | 419.858      | 281.742      | 0.928       | 198.342       | 115.428       | 0.184       | 71.530        | 24.5586        |
| 4     | 894.687      | 1019.207     | 0.885       | 484.266       | 246.208       | 0.278       | 175.203       | 229.72         |
| 5     | 1206.356     | 1934.871     | 0.954       | 600.088       | 240.044       | 0.184       | 200.589       | 494.634        |
| 6     | 1553.916     | 1984.768     | 0.889       | 778.505       | 264.966       | 0.196       | 262.737       | 534.128        |
| 7     | 2101.045     | 2370.313     | 0.906       | 933.024       | 352.855       | 0.186       | 311.356       | 433.812        |
Figure 6. Performance comparison of different algorithms, (a) shows the total time for interference check, (b) shows average time per frame during the interference, (c) shows maximum frame time during the interference, (d) shows running memory usage.

According to the experimental data, the interference check time of different algorithms can be seen. The improved algorithm has dramatically shortened the bounding box tree's construction time compared with other algorithms through exploratory analysis, and the interference check speed has been improved. It can be seen from the comparison chart of running memory occupation that the cost of the algorithm in this paper is much smaller than other algorithms in the interference check test.

4. Conclusion
Aiming at the problem that the traditional industrial robot may interfere with the workpiece when running in the offline simulation environment, this paper uses a hybrid bounding box algorithm with an octree structure to replace the traditional binary tree hierarchical bounding box processing method. The bounding box construction and the bounding box tree are optimized in the algorithm's rough detection stage. The efficiency of the algorithm is significantly improved, and the system overhead has been reduced. In future work, the algorithm's space complexity will be optimized, and an attempt will be made to apply the algorithm to a multi-object simulation system.

References
[1] Melero FJ, Aguilera A and Feito FR 2019 Fast collision detection between high resolution polygonal models J Computers & Graphics pp 83 97-106
[2] Stuvel SA, Magnenat-Thalmann N, Thalmann D, Egges A and van der Stappen AF 2014 Hierarchical structures for collision checking between virtual characters J Computer Animation and Virtual Worlds pp 25(3-4) 333-342
[3] Liang T, Wei-guo S, Tian-cheng H, Lei-lei L, Wei-xing C and Yan Z 2018 Collision detection of virtual plant based on bounding volume hierarchy: A case study on virtual wheat J Journal of Integrative Agriculture pp 17(2) b306-314
[4] Advisor DM, Lin MC, Brooks FP and Gottschalk S 2000 Collision queries using oriented bounding boxes The University of North Carolina at Chapel Hill
[5] Zhi H, Pengxuan W, Congbao W, Xiangang W and Jingyi W 2020 Collision detection method of blisk grinding and polishing J Computer Integrated Manufacturing Systems pp 26(12) 3350-3358
[6] Gottschalk S, Lin MC and Manocha D 1996 OBBTree: A Hierarchical Structure for Rapid Interference Detection J ACM SIGGRAPH Computer Graphics pp 30(Annual Conference Series)