CS448B – Paper Critique

OBBTree: A Hierarchical Structure for Rapid Interference Detection

Menelaos Levas
mlevas@leland.stanford.edu

1 Citation of paper

OBBTree: A Hierarchical Structure for Rapid Interference Detection, S Gottschalk, M C Lin, D Manocha, Department of Computer Science, University of North Carolina.

2 Paper overview

A data structure and an algorithm is proposed for detecting interference among complex models undergoing rigid motion. Initially, a hierarchical tree of oriented bounding boxes (OBB) is pre-constructed for each model. At runtime, two such tree structures are traversed to detect overlap between their bounding boxes using a test based on a separating axis theorem. This test is claimed to require fewer than 200 operations per box pair. The algorithm and data structure do not make any assumptions as far as the complexity of the models is concerned, and very accurate and robust collision detection is possible at interactive rates.

3 Problem specification

Input: A pair of complex three dimensional models composed of hundreds of thousands of polygons, with no topology information. Cracks, T-joints, and non-manifold geometry is possible.

Output: Robust and accurate collision detection that is efficient enough to allow interactivity.

4 Basis of the algorithm

A typical way of detecting interference is to construct a hierarchy of bounding volumes (spheres, axis aligned boxes/cubes) that bound the model, and test this hierarchy for overlap against the respective hierarchy of the model with which we want to detect interference. All the previous algorithms focused on selecting bounding volumes that would provide very simple overlap tests. However, these volumes often do not bound the model very tightly. Consequently, when the objects are far from each other a simple test will indicate non interference immediately, but when the objects are almost into each other, huge hierarchies will have to be traversed to accomplish accurate detection.

The authors went down the opposite path when designing their algorithm and data structure. The rationale behind this decision is that it is best to greatly reduce the number of bounding shapes required to tightly embrace the object at the expense of a rather more complex overlap test. Their proposal is to use bounding boxes that are oriented to embrace the model far more tightly.

Initially, an Oriented Bounding Box hierarchy is constructed for the model. This process comprises of two steps. First, a tight fitting OBB is placed around a group of polygons and then multiple nested OBBs are grouped into a hierarchy.

To create a tight fitting OBB, all polygons involved are triangulated. All the vertices are then used to determine first and second order statistics (mean and covariance) from which the axes of the bounding box can be determined. The extreme values of the vertices along these axes are used to determine the size of the box. The low level statistics and mathematics involved will not be examined in detail here. The basic failure of this approach is that because it relies on the statistical distribution of the vertices, it will yield sub optimal orientations (and hence OBB that do not fit the model tightly) if there are many internal points in the model that are somehow aligned to each other. These will bias the statistic calculations.

A proposed improvement is to use the convex hull (ie the smallest convex volume that contains all the vertices) of the vertices rather than the model itself to generate the OBB. Unfortunately, the convex hull may still have points that will bias the calculations. A more elaborate method is to sample the surface of the convex hull so as to ensure a smoother distribution, and use the sampled points for OBB generation. This can be done infinitely densely by integrating over the surface of the convex hull.

To create the full hierarchy, a top down approach is used to generate the OBB tree. First an OBB is generated for the entire model, and it is subsequently split into smaller OBBs. Subdivision occurs by splitting an OBB across its longest axis with a plane orthogonal to that axis. The polygons are then partitioned around that plane and OBBs are constructed for each of the two polygon groups. If the partitioning occurs around the median of the points, balanced OBB trees result. Total runtime for the entire OBB tree construction is $O(n\lg^2 n)$ if convex hulls are used and $O(n\lg n)$ if they are not.

The core of the proposed collision detection mechanism is the overlap test between two OBBs. A simple minded algorithm will test every edge of each box with every facet of the other box for intersection, resulting to a total of 144 edge to face tests, before (non)interference can be determined. The authors propose a far more efficient test that relies on eliminating the possibility of interference as early as possible so that subsequent tests can be skipped. The basis of the test is that any two disjoint convex objects in 3-space can be separated by a plane that is either parallel to a facet of either object or parallel to an edge from each object. Since each OBB has 3 facet orientations and 3 edge directions there are 15 such planes for an OBB pair (3 for the facet of one OBB, 3 for the facet of the other OBB, and 9 for all the edge pairs of the two OBBs). The idea behind the test is to project the two OBBs on an axis that is orthogonal to each of these planes and test if their projections overlap. If they do not in any one of these 15 tests then the OBBs are known to be disjoint and the test terminates. If they overlap in all 15 tests then the OBBs overlap. In the worst case, all 15 possible axes will be tested with a cost of 200 floating point operations. On average though, the test will return earlier if the OBBs are disjoint. Specific optimizations can be performed in special cases when OBBs have zero or infinite thickness.

5 Results and discussion

An important feature of this paper is that it is not merely a theoretical proposal, but in fact the algorithm has been implemented and tested. This way, actual results are available, and they indicate that the algorithm performed well given hugely complex input, maintaining the much desired interactive rates.

Obviously, the greatest advantage of the algorithm is that is requires significantly fewer bounding volumes to approximate a model, which is the main source of its efficiency. All other bounding volume schemes converge linearly to the approximated model. OBBs on the other hand, exhibit quadratic convergence. The number of bounding volumes in the hierarchy is vital as it is often the case when we test two models for interference, we test many bounding volume pairs. Reducing the number of these comparisons will greatly outbalance the increased complexity of testing two OBBs for overlap. In this respect, the algorithm is successful.

The robustness and accuracy of this process is also worth mentioning. It does not need to handle any sort of special cases such as parallel facets or edges. No adjacency information is required in the model, and interference detection can be performed even for space curves (approximated as degenerate triangles).

The OBB pair overlap test is a significant contribution of this paper. The efficiency of this test is significant compared to older algorithms. A most important advantage is that it does not use divisions, square roots or other functions that have special cases and is therefore numerically stable. It is simple enough to be implemented in assembly or even hardware for dramatic boosts in performance.

Naturally though, this algorithm suffers from the same problem experienced by all other successful collision detection algorithms: it is good for *most* situations, and the keyword here is most. In fact there is no single collision detection algorithm that performs well under *all* circumstances. For this particular algorithm, performance is very good when two extremely complex models are practically into each other, but not so good when two models are far apart. A simple bounding sphere test will suffice for that situation. In fact, most applications usually require lots of tests for one of the two scenarios and few tests for the other. Which of the two is the most common is entirely application specific. Consequently, hybrid algorithms may be the way to go.

The authors acknowledge one more shortcoming in addition to the one above: Since orientation for each bounding box is maintained as a transformation of its parent, it is possible that numerical errors will propagate down the tree hierarchy. Although in practice this did not cause any errors it is still a theoretical possibility.

I believe that the most major shortcoming of this algorithm is its storage requirements. Given that the OBB tree has about twice as many boxes as there are triangles, and that each box requires 168 bytes of storage data, the size of memory required to store the model expands by a factor of 6. Memory requirements are perhaps not as much of an issue as it used to be but memory bandwidth is. An implementation of this algorithm will have to transfer huge amounts of memory through buses that are not becoming faster in the same rate as logic. In fact, limited memory bandwidth is the prime bottleneck in graphics performance. The authors propose a couple of ways to reduce the memory footprint of an OBB, both of them involving trade-offs. Using floating point arithmetic for instance, will probably give rise to the propagation errors mentioned earlier. Using quaternions will complicate the OBB overlap test. A reasonable optimization would be instead of storing a rotation matrix, a translation vector and the sizes of each of the OBB's axes, to store a single concatenated matrix that would incorporate rotation, translation and scaling. It is not clear in the paper if the rotation matrices used in the existing implementation are 3x3 (instead of 4x4 required for a concatenated matrix). In any case, the single matrix solution could also result in performance gains.

It is unclear if the lower bounding volume count in the structure balances out the increased storage space required for each volume compared to, for example a sphere hierarchy where each sphere requires only 32 bytes of storage, but far more spheres are required to approximate the model. It only remains to be seen if this proposal gains an acceptance that is wide enough to spawn research for resolving the storage problem, through inexpensive (de)compression schemes. This is an area that could form the basis of another paper.