Exact Collision Detection for Interactive Environments

Jonathan D. Cohen
Computer Science Department
University of North Carolina
Chapel Hill, NC 27599-3175

Ming C. Lin
Computer Science Department
Naval Postgraduate School
Monterey, CA 93943

Dinesh Manocha *
Computer Science Department
University of North Carolina
Chapel Hill, NC 27599-3175

Madhav K. Ponamgi
Computer Science Department
University of North Carolina
Chapel Hill, NC 27599-3175

Abstract:

We demonstrate algorithms for collision detection in large-scaled, interactive environments. Such environments are characterized by the number of objects undergoing rigid motion and the complexity of the models. We do not assume that the motions of the objects are expressible as closed form functions of time, nor do we assume any upper bounds on their velocities.

Over the last few years, a great deal of interest has been generated in *virtual or synthetic* environments, such as architectural walkthroughs, visual simulation systems for designing the shapes of mechanical parts, training systems, etc. Interactive, large-scaled virtual environments pose new challenges to the collision detection problem, because they require performance at interactive rates for thousands, of pairwise intersection tests [1].

Our algorithms use a two-level hierarchical detection test for each model to *selectively* compute the *precise* contact between objects, achieving real-time performance without compromising accuracy. At the top level, we use a *dimension reduction* approach (based on a technique termed as *sweep and prune*) and the *temporal and geometric coherence* that exists between successive frames to overcome the bottleneck of $O(n^2)$ pairwise comparison tests in a large environment of n moving objects [1]. After we eliminate the objects pairs which are not in the close vicinity of each other, we employ an exact collision detection scheme. The exact convex polytope collision detection algorithm is based upon [2]: locality (captured by walking from one Voronoi region of the candidate feature to one of its neighboring feature's Voronoi region according to the constraints failed) and coherence (used to reduce computation efforts in a dynamic environment). We can deal with non-convex objects or articulated bodies by using a sub-part hierarchical tree representation.

*Supported in part by a Junior Faculty Award and ARPA Contract #DAEA 18-90-C-0044

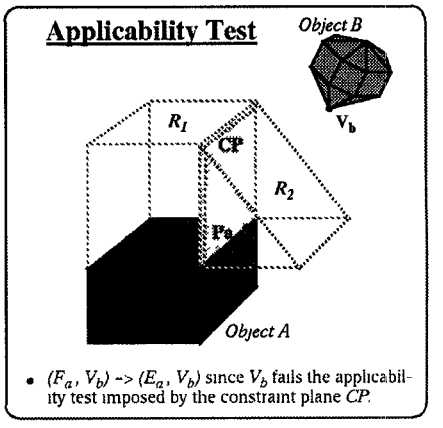

Figure 1: An Example of collision detection for convex polyhedra

• Exact Collision Detection

The Voronoi regions form a partition of space outside the polytope according to the closest feature. The collection of Voronoi regions of each polytope is the generalized Voronoi diagram of the polytope. Note that the generalized Voronoi diagram of a convex polytope has linear size and consists of polyhedral regions. Each Voronoi region is bounded by a set of constraint planes with pointers to the neighboring cells (which each share a constraint plane with it).

If a point lies on a constraint plane, then it is equidistant from the two features which share this constraint plane in their Voronoi regions. If a point P on object A lies inside the Voronoi region of the feature f_B on object B, then f_B is a closest feature to the point P.

Our method is straightforward. We start with a candidate pair of features, one from each polytope, and check whether the closest points lie on these features. Since the polytopes and their faces are convex, this is a local test involving only the neighboring features of the current candidate features. If either feature fails the test, we step to a neighboring feature of one or both candidates, and try

again. With some simple preprocessing, we can guarantee that every feature has a constant number of neighboring features. This is how we can verify or update the closest feature pair in *expected constant* time. Fig. 1 shows how this algorithm works on a pair of simple convex polytopes.

• **Sweep and Prune**

The exact polytope collision detection algorithm is fast enough to perform collision detection at real-time, in a multi-body environment of moderate size. However, the $O(n^2)$ pairwise detection tests become a bottleneck for large n. By exploiting the spatial and temporal coherence, we use a method whose running time is a linear function of the number of objects and the number of pairs of objects in close vicinity of each other.

For each object in the environment, we compute a smallest fitting, axis-aligned bounding box, which is dynamically resized using a hill-climbing algorithm based on the properties of convex polytopes and the principle of coherence.

These bounding boxes are used to prune down the number of pairwise collision detection tests. We take the orthogonal projections of the bounding boxes onto the x, y, and z axes and sort the intervals in three separate lists. If the projections overlap in *all three* dimensions, we activate the exact collision detection algorithm. At each frame, we sweep over the sorted lists of intervals, updating them and culling out the pairs of non-overlapping bounding boxes.

Each polytope in the environment now has a bounding box. We only call the exact collision detection algorithm between two objects when their bounding boxes overlap. Due of coherence, the sorted lists of intervals, constructed from the projections of the bounding boxes, change little from frame to frame, so we use a bubble or insertion sort to update these lists in expected linear time. Without the bounding box hierarchy, we would have to perform a quadratic number of pairwise exact detection tests.

• **Video Description**

We have successfully demonstrated the algorithm in an architectural walkthrough environment and a multi-body simulation system. For simulations consisting of thousands of models, the overall algorithm has exhibited linear time performance in practice.

We first demonstrate how the polytope collision detection algorithm [2] tracks the closest features between two convex polytopes in real time. We maintain and update the closest features using the Voronoi regions of the polytopes. Since the polytopes move only slightly between the frames, the temporal and geometric coherence is well preserved. Thus, the expected running time of this algorithm is constant, independent of the complexity of the polytopes.

This is followed by demonstration the algorithm on an articulated body, namely a hand, represented by the union of multiple convex polytopes. To demonstrate the efficiency of the algorithm and effectiveness of the coherence, we keep track of all pairs of closest features between each finger and the nearby objects. However, using a subpart hierarchical tree representation as in [2] can eliminate the unnecessary computation.

The over all collision detection algorithm is tested on a

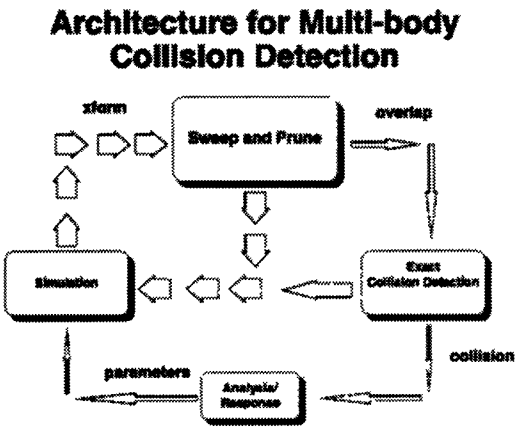

Figure 2: The Architecture for multi-body simulation

walkthrough and a multi-body dynamic simulation environment. The video footage of walkthrough was taken from a user's **actual walkthrough experience**. The collision detection algorithm performs very well in large environments composed of hundreds of polytopes. The kitchen, like most realistic environments, is fairly static; therefore, we can sort the bounding boxes of the polytopes in sub-linear time at each instance (this requires a modified sorting algorithm which sorts only the moving objects). We see very little degradation of frame rate when we add the capability of collision detection to our walkthrough environment.

We further demonstrate a multi-body simulation with several complex objects in a moderately dense environment. The objects are placed in a bounded 3-dimensional simulation volume whose walls act as barriers. (The architecture of multi-body simulation is shown is Fig. 2.) Unlike the walkthrough environment, *all* the objects move independently. The dynamics of multi-body simulation assumes elastic collision. In order to show the asymptotic time bound on the algorithm, we increase the number of possible interactions quadratically, but the frame time using our collision detection algorithm increases only linearly.

• **Acknowledgements** We are grateful to Dr. Fred Brooks and the walkthrough group at UNC Chapel Hill for the model of the kitchen.

References

[1] J. Cohen, M. Lin, D. Manocha, and K. Ponamgi. Interactive and exact collision detection for large-scaled environments. Technical Report TR94-005, Department of Computer Science, University of North Carolina, 1994.

[2] M. C. Lin. *Efficient Collision Detection for Animation and Robotics.* PhD thesis, University of California at Berkeley, December 1993. Department of Electrical Engineering and Computer Science.