Computer Sciences Technical Report CS-2005-1529

Detecting Collisions in Graph-Driven Motion Synthesis

Mankyu Sung [*] Stephen Chenney Michael Gleicher

Computer Sciences Department
University of Wisconsin, Madison

Abstract

In this paper we consider detecting collisions between characters whose motion is specified by motion capture data. We consider rough collisions, modeling the characters as a disk in the floor plane. To provide efficient collision detection, we introduce a hierarchical bounding volume, the Motion Oriented Bounding Box tree (MOBB tree). A MOBBtree stores space-time bounds of a motion clip. In crowd animation tests, MOBB trees performance improvements ranging between two and an order of magnitude.

Keywords: collision detection, motion graphs, OBB tree, swept volumes

1 INTRODUCTION

Data driven animation based on motion capture is a common way to animate human characters. When multiple characters are animated this way, care must be taken to provide proper interaction between them. A minimal requirement is that they do not interpenetrate. A better goal is that two characters interact appropriately when they are close. In either case, when a system selects captured motion clips for characters, it must check is these motions will lead to the characters colliding so it can respond appropriately. This paper presents algorithms for identifying potential collisions between motion clips, and hence animated characters.

When many characters are animated simultaneously, such as in crowd simulation, efficiency is an important concern. Detecting interactions between fully articulated characters is computationally intensive, and may be excessive if the application requires characters to maintain reasonable seperations. We therefore focus on the use of a simplified character geometry: a bounding cylinder (Figure 1, left). In cases where this simplification is unaccaptable, the efficient algorithms it affords can be used as a culling step to identify potentially interacting agents.

Even with the simplified geometry, checking for interactions between characters on every frame can be prohibitive if there are many characters. The problem is even worse if we are evaluating potential choices for motions for each character. We therefore exploit the fact that data driven animation often selects an entire clip of captured motion at once. We can, therefore check entire motions for collisions before they begin.

The motion clip collision problem considers two motions, along with their transformations that place their starting points in space. One motion may have a time offset, for the event that the two do not start simultaneously. A collision must be detected if at any time during the duration of the motions the bounding cylinders for the characters associated with the motions overlap. If the characters are spatially disjoint, or at the same place at different times, there are no collisions.

To perform motion clip collision detection efficiently, we precompute a hierachical representation of each motion. We refer to our collision data structure as a Motion Oriented Bounding Box (MOBB) tree (Figure 1, right). It is a space-time variant of OBB trees [Gottschalk et al. 1996] targeted at skeletal motion clips, and can be viewed as a continuous collision detection technique based on hierarchies of swept volumes. The design of MOBB trees is motivated by several motion specific properties: the agent's path is densely point sampled in time and can be arbitrarily shaped (Figure 1, center); the time steps are large compared to typical physically based simulation; the aim is to avoid collisions entirely, so we require a yes/no intersection test and have no need for contact points etc.; and agents are moving on the ground plane, so the problem is 2D with time (we are looking for intersections between circles extruded in time – thin disks in space-time). These properties primarily drive the way in which an MOBB tree is constructed, but also influence the intersection testing algorithm.

The next section provides a review of related work. Sections 3 and 4 precisely define MOBB trees, describe their construction and present algorithms for intersection testing. We close with several experiments and a discussion of future work.

2 RELATED WORK

Our algorithm is a novel application of OBB trees [Gottschalk et al. 1996] to swept volume intersection in space-time. OBB trees have been applied to continuous collision detection in the past [**?**; **?**], but existing techniques use bounding boxes fitted to static geometry and account for motion in the intersection test. This restricts their application to simple parameterized motion (such as linear translation [Eberly 2001]) and makes them unsuitable for motion capture data.

The majority of literature in the graphics community is targeted at simulation algorithms where the future motion of the body is either ballistic or bounded but unknown (see Mirtich [1996] for rigid body examples). Kim et. al. [2003a] describe a crowd-specific solution that uses parabolic horns as space-time bounds (spheres of changing radii swept along parabolic paths), while kinetic data structures [Basch et al. 2004] assumes motion along rational curves. Our problem differs in that we know the precise trajectory but in a sampled form, and we have a generate-and-test strategy, rather than a continuous search for the next collision.

The robotics community has dealt with similar problems in the guise of intersection-free robot motion planning. Several solutions have been proposed based on 4D space-time swept volumes (see Abdel-Malek et. al. [2002] for a survey). Recent advances include work aimed at complex models [Kim et al. 2003b], but the most similar approach to ours is due to Foisy and Hayward [1993]. They use a hierarchy of convex swept volumes, each volume specified by a set of bounding planes. Our method is simpler due to the use of OBB trees and targeted specifically at motion in the ground plane.

MOBB trees detect collisions between the cylindrical bounding volumes of skeletal motion, not the skeleton itself. Redone et. al. [**?**] provide a solution for close-in skeletal motion that also exploits hierarchies of volumes, but assumes small time intervals between tests. Our approach can be viewed as a broad phase test that complements their work.

[*]mksung@cs.wisc.edu, http://www.cs.wisc.edu/graphics

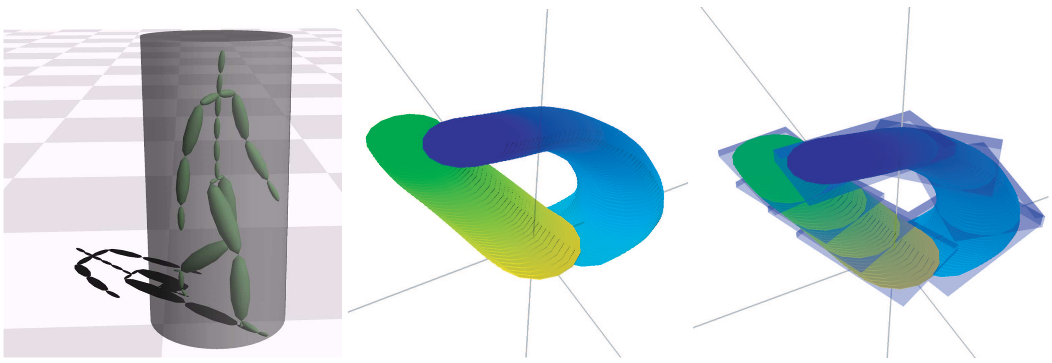

Figure 1: *On the left, a sample frame of motion data from a clip used in our system. The skeleton is bound by a vertical cylinder centered on its root position with a radius large enough to contain the limbs. Center is a 3D space-time visualization of two motions that do not intersect in time (vertical axis) even though their paths cross in space. Each cylindrical bound is now a thin disk - a circle in space extruded in time. On the right, part of the Motion Oriented Bounding Box (MOBB) for the motions. This hierarchical structure bounds the motion in space-time and enables efficient collision queries.*

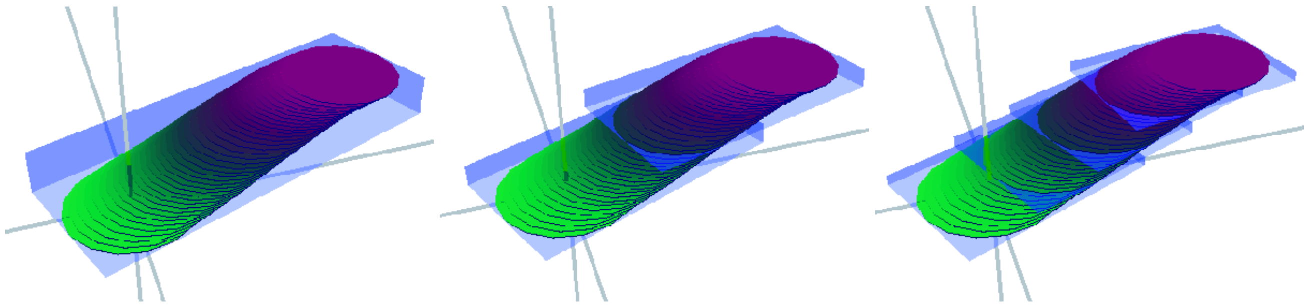

Figure 2: *Three levels in an MOBB tree hierarchy. Boxes are split evenly in the time (vertical) dimension going from one level to the next, and then spatial bounds are generated containing all the samples in that time-slice.*

3 MOBB TREES

The motion clip collision detection problem considers a pair of motions, $\mathbf{m_a}(t)$ and $\mathbf{m_b}(t)$. A motion is a function that maps times to character pose. Because we simplify character geometry as a cylinder, we consider motion functions as providing three values for any given time: the position of the character in the plane (x, y) and the radius of the cylinder around the character r. For simplicity, we center the cylinder around the projection of the character's root joint onto the ground plane. We compute r from the point on the character whose projection is furthest from this position. When the character is assymetric, for example when carrying a large sword, the use of the root as center leads to an oversized bounding cylinder. In practice, this has not been a problem. In fact, we often use a fixed value for r.

Because we are working with data-driven animation, motions are representing by samples. We assume, without loss of generality, that the first sample is at time $t = 0$. We also assume that the motions are sampled finely enough that we need not consider interpolation between samples.

A collision detection test is given two motions, $\mathbf{m}_a$ and $\mathbf{m}_b$, a 2D transformation for each, $\mathbf{T}_a$ and $\mathbf{T}_b$, and a time offset for each, t_a and t_b measured in a global timeframe. Define t_{start} as $\max(t_a, t_b)$ and $t_{end,a}$ and $t_{end,b}$ as the last sample in $\mathbf{m}_a$ and $\mathbf{m}_b$ respectively. The test should return a positive result (an intersection) if there exist sample times,

$$\begin{array}{ccccc} (t_{start} - t_a) & \leq & o_a & < & t_{end,a} \\ (t_{start} - t_b) & \leq & o_b & < & t_{end,b} \end{array}$$

such that

$$\|\mathbf{T}_a\mathbf{m}_a(o_a) - \mathbf{T}_b\mathbf{m}_b(o_b)\| < r_a(o_a) + r_b(o_b) \tag{1}$$

In practice, the time offsets could be arbitrary real numbers, while the samples are discrete. We therefore interpret $\mathbf{m}_a(o_a)$ to be the sample from the time closest to but below o_a. An alternative is to interpolate $\mathbf{m}_a$, but our sample spacing is sufficiently fine with respect to the agent's speed and size that this is unnecessary for the purposes of collision avoidance.

In 3D space-time, each sample is a short cylinder, axis aligned with the time dimension, the center of the base at $(\mathbf{m}(t); t)$, radius $r(t)$ and height equal to the sample spacing, dt. Detecting a collision is equivalent to identifying collisions between these space-time cylinders, appropriately transformed.

An MOBB tree is a hierarchical bounding volume in 3D space-time. Each node in the hierarchy is an oriented bounding box with one axis parallel to the time domain and the other two axes lying in the xy-plane. This is equivalent to a 2D spatial OBB extruded in the time domain. Each node bounds a set of samples from t_{min} to t_{max}. The children of a node in the tree bound the subsets of samples from t_{min} to $\frac{t_{max}-t_{min}}{2}$ and $\frac{t_{max}-t_{min}}{2}$ to t_{max}. In other words, the hierarchy is built by subdividing the volume at its midpoint in time.

The following sections describe how we construct a MOBB tree from a motion, and how we intersect two MOBB trees.

3.1 Fitting MOBB trees

MOBB trees are built in a manner analogous to standard OBB trees: given a sequence of samples to bound, we must compute the orientation and dimensions of the box, and then recurse on the two child sub-sequences. Note that in each node we store data defining a 2D OBB tree and the time range for which it is valid, which can be thought of as the third dimension of the space-time box. Three levels of an example tree are shown in Figure 2.

Given a set of sample points in 2D, the optimal oriented bounding box (OBB) can be computed by computing the second order statistics of the points [Gottschalk et al. 1996]. In practice, we find it sufficient to approximate the optimal box with one obtained by a simpler method.

Given a set of sample points, we choose the major axis of the OBB, $\mathbf{a}_0$, by subtracting the location of the first sample, $m(t_{min})$ from that of the last, $m(t_{max})$, and normalizing:

$$\mathbf{a}_0 = \frac{\mathbf{m}(t_{max}) - \mathbf{m}(t_{min})}{||\mathbf{m}(t_{max}) - \mathbf{m}(t_{min})||}$$

The minor axis is perpendicular: $\mathbf{a}_1 = (-a_{0,y}, a_{0,x})$.

To compute the dimensions of the box, we iterate over the samples and keep track of the maximal extents seen. To compute these, each sample is transformed into the box's local coordinate system and $r(t)$ is added (subtracted) to get the maximal (minimal) extent in each dimension. If $\mathbf{d}_{max}$ and $\mathbf{d}_{min}$ are the largest and smallest extents found, then the center of the OBB is at

$$\mathbf{c}_{local} = \frac{\mathbf{d}_{max} + \mathbf{d}_{min}}{2}$$

and the dimensions of the box are

$$\mathbf{d} = \frac{\mathbf{d}_{max} - \mathbf{d}_{min}}{2}$$

The center is transformed back into global coordinates and stored, along with $\mathbf{a}_0$, $\mathbf{a}_1$ and $\mathbf{d}$. Each node also stores t_{max} and t_{min}.

After computing the bound at one node, we divide the duration of the sample sequence in half. Recursion stops when a fixed number of samples, n_{min}, remain and the samples are stored in the node (OBBs are still computed and stored for leaf nodes). We experimented with various values for n_{min}, ranging from 1 to 50. Optimal performance occurred at $n_{min} = 10$, which reflects the cost of a 2D OBB intersection relative to computing the distance between two samples. Performance is near best when one OBB test is equivalent to $\frac{n_{min}}{2}$ sample distance tests.

Standard OBB tree construction algorithms [Gottschalk et al. 1996] use second order statistics to determine the box axes. We tested this method but found it gave essentially identical results (comparing box area) as our method, which is simpler to implement.

3.2 Intersection Testing

Intersection testing of two MOBB trees is very similar to testing standard OBB trees. Note that we are seeking only yes/no intersection queries, and hence can exit as soon as an intersection is found. Input to the intersection test is two MOBB nodes, A and B, their 2D spatial transformations from world coordinates, $\mathbf{T}_a$ and $\mathbf{T}_b$ (rotation and translation) and the time offsets, t_a and t_b. An example collision test is presented in Figure 3.

We first test that the nodes overlap in time: if $t_a + A.t_{max} < t_b + t_{min}$ then the boxes do not overlap in time (Figure 3(c)) and we can exit with no collision, and the same if $t_a + A.t_{min} > t_b + t_{max}$. If temporal overlap is found, we test A's and B's 2D OBBs using separating axes tests. There are only four axis tests required. If the OBBs do not overlap there is no collision (Figure 3(b) and (d)), otherwise we perform one of two actions (Figure 3(a) and (e)): if one of the boxes is a leaf, we call a procedure to test the leaf against the other tree; otherwise we make recursive calls to compare all the child nodes.

A leaf versus tree node test checks the time interval covered by the leaf against the time intervals covered by the node's children (Figure 4). If a child overlaps the leaf, we recurse on the child.

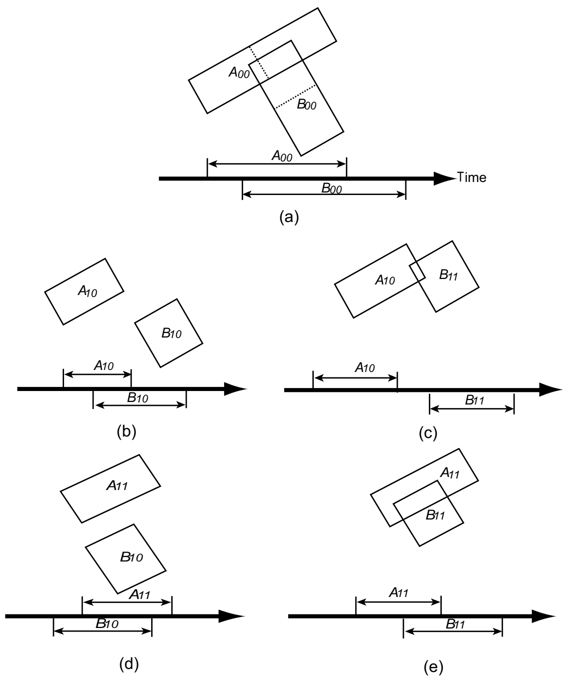

Figure 3: *Testing two internal MOBB tree nodes, A_{00} and B_{00}. The relative spatial and temporal arrangement of the nodes is shown in (a). The algorithm first tests for temporal overlap, and then spatial overlap. In this case there is an intersection, so the algorithm recurses with the four combinations of child nodes. At the next level, tests (b) and (d) fail because there is no spatial overlap, test (c) fails because there is no temporal overlap (no spatial test is done), and test (e) leads to further recursion.*

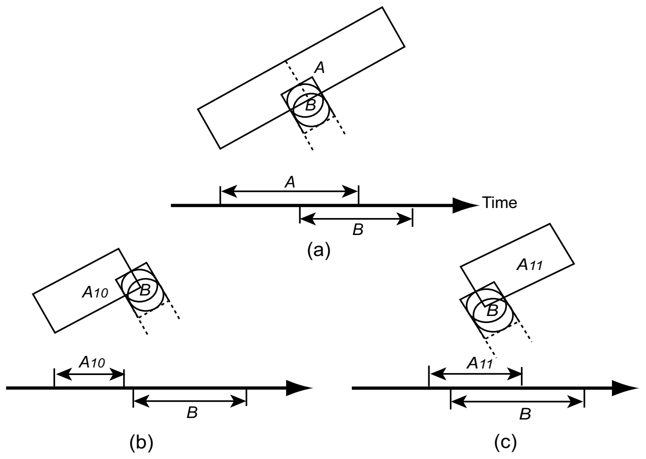

Figure 4: *Testing an internal node against a leaf node. In (a), we have reached a leaf node B while at a non-root node A in the other tree. The time range of B is tested against A's children, A_{10} and A_{11}. In one case, (b), the time interval does not overlap, so we stop with no intersection found. In the other case, (c), a temporal overlap is found so the algorithm recurses with A_{11} and B. Note that we do not perform a spatial test in case (c); experiments found it gave no advantage.*

Recursion continues until we have two leaves, at which point corresponding samples are found from A and B and the distance between them compared to the bounding radii. In other words, we explicitly search for samples with o_a and o_b satisfying Equation 1.

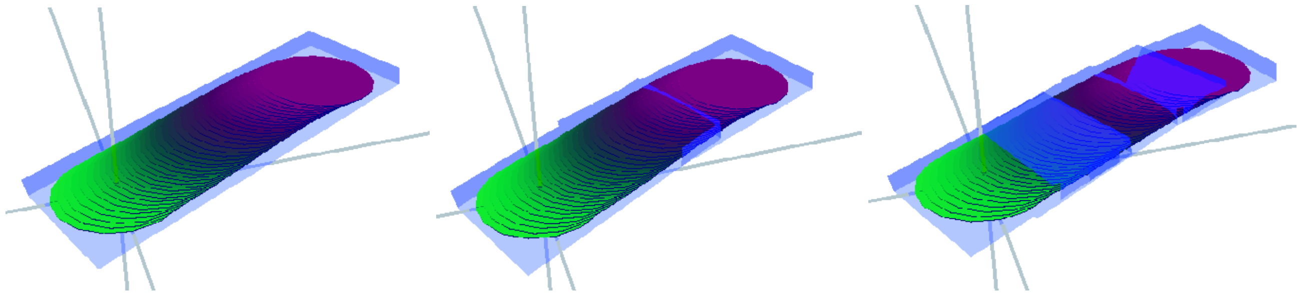

Figure 5: *Three levels in an unrestricted MOBB (UMOBB) tree hierarchy. In this tree, bounding volumes are 3D OBBs in space-time, removing the restriction that one axis align with the time dimension. Boxes are still split evenly in the time dimension, but a 3D OBB is fitted to the samples. UMOBB trees have tighter bounds than MOBB trees, but are more expensive to test for intersection.*

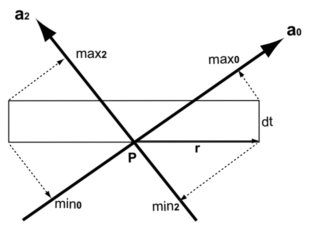

Figure 6: *Computing the extent of a sample disk for a 3D space-time OBB. The disk has the center of its base located at the sample point,* **p***, with radius vector* **r** *and height dt. We must find the minimal and maximal projections onto the axes $\mathbf{a}_0$ and $\mathbf{a}_2$. Note the asymmetry in the cylinder's position.*

4 UNRESTRICTED MOBB TREES

The MOBB trees described thus far always use the time axis as one of the OBB axes in 3D space-time. This restriction can be relaxed, essentially treating the samples as regular 3D geometry and building 3D OBBs to bound them. We refer to trees built in this manner as Unrestricted MOBB trees, or UMOBB trees. UMOBB trees are expected to give tighter space-time bounds (Figure 5), and hence require fewer tests to identify non-intersecting cases.

To determine the axes of the 3D OBB in space-time, we determine the first axis, $\mathbf{a}_0$, by subtracting the space-time location of the first sample, $(\mathbf{m}(t_{min});t_{min})$ from that of the last, $(\mathbf{m}(t_{max});t_{max})$ and normalizing (similar to the 2D case, but including the time dimension). We obtain a second axis, $\mathbf{a}_1$, mutually orthogonal to $\mathbf{a}_0$ and the time dimension, $(0,0,1)$. Finally, $\mathbf{a}_2 = \mathbf{a}_0 \times \mathbf{a}_1$.

The dimensions of the box are found by transforming the sample points into the box's coordinate system, projecting the extents of the samples' cylinders onto the axes and hence finding the maximal projection across all samples. Figure 6 illustrates the projection. First we compute **r**, which is the vector with length equal to the disk's radius, r, aligned with $\mathbf{a}_0$ in the xy-plane:

$$\mathbf{r} = r\frac{(\mathbf{a}_{0,x}, \mathbf{a}_{0,y}, 0)}{\|(\mathbf{a}_{0,x}, \mathbf{a}_{0,y}, 0)\|}$$

From the figure, we see that

$$\begin{aligned} min_0 &= -\mathbf{r}\cdot\mathbf{a}_0 \\ min_2 &= \mathbf{r}\cdot\mathbf{a}_2 \\ max_0 &= \mathbf{r}\cdot\mathbf{a}_0 + dt\,\mathbf{a}_{0,t} \\ max_2 &= -\mathbf{r}\cdot\mathbf{a}_2 + dt\,\mathbf{a}_{2,t} \end{aligned}$$

The extents in the remaining direction, $\mathbf{a}_1$, are at distance r by construction. Taking the maximum and minimum over all samples gives us the necessary information to compute the box origin and dimensions. The node of an UMOBB tree stores the box properties (origin, axes, dimensions) in addition to t_{min} and t_{max}. Keeping the times allows for a fast early reject test when looking for box intersections.

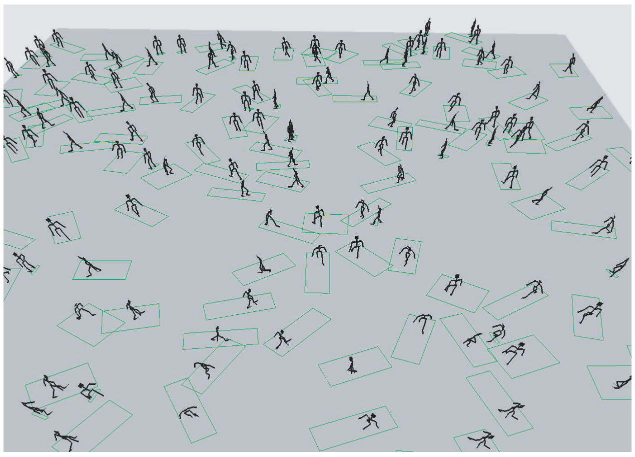

Figure 7: *Snapshot of the crowd simulation used to explore the performance of MOBB trees. The environment is a 30×40 meter rectangle containing 100 agents.*

Intersection testing of UMOBB trees is is essentially identical to that of MOBB trees, the only difference being the use of 3D OBB tests. A 2D transformation plus a time offset becomes a 3D space-time transformation by applying the rotation about the t-axis and using the temporal offset as a translation in the t dimension. Otherwise the algorithm is identical.

5 EXPERIMENTS

We performed a series of experiments to explore the benefits of MOBB trees under an application workload. Our test environment is a crowd simulator in which agents wander through the world avoiding collisions (Figure 7). The agents are animated with a Snap-Together Motion [Gleicher et al. 2002] style motion graph built from 51 motions. The motions have an average length of only 2.1 seconds, or 63 frames. With $n_{min} = 10$ (the number of samples in a leaf node), the tree is very shallow - only 3 levels on average. This limits to some extent the benefits we see from a hierarchical method. The radius of the bounding sphere around each agent was fixed at $r = 1.85$ meters.

As a base case for comparison, we used MOBB trees that performed bounding box tests only at the root, referred to as "1 level" trees in Table 1. These trees simulate a collision detection method that tests an OBB bound for the entire motion, and then uses binary search on time to identify potentially overlapping samples (a hierarchy in time but not space). Our results hence show the performance advantage gained from a hierarchy of bounds compared to an algorithm that uses only a root bound but is otherwise intelligent about

Method	Time (10^{-6}s)	# 2D	# 3D	# sample
MOBB 1 level	4.6	0.45	0	10.3
UMOBB 1 level	4.5	0	0.45	10.0
MOBB	2.3	1.7	0	0.91
UMOBB	2.2	0.16	1.53	0.77
Hybrid 1	2.1	0.40	1.34	0.77
Hybrid 2	2.0	1.5	0.25	0.81

Table 1: *Results for our application-based experiment. See the text for a description of the methods. The table shows average time per intersection query, the average number of 2D and 3D OBB tests per intersection query, and the average number of sample-sample overlap tests. Note that, even though the UMOBB contains no explicit 2D nodes, some 3D boxes end up aligned and hence are treated as 2D. This explains the non-zero count for 2D tests in the UMOBB trees.*

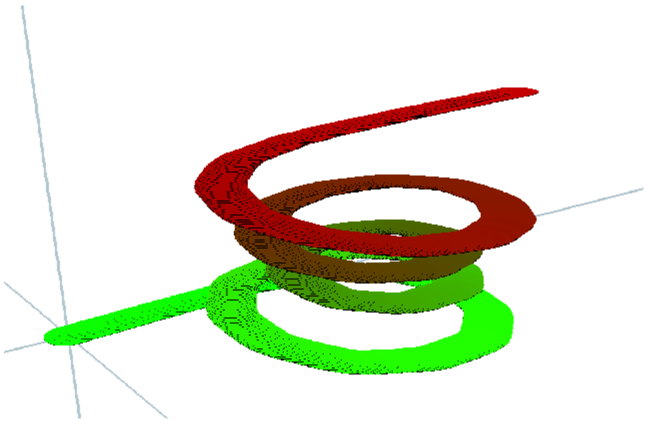

Figure 8: *One of the long motions used in our experiments. The motion clip is of someone walking in circles.*

avoiding sample tests.

In addition to results for the "1 level", MOBB and UMOBB trees, we also experimented with two hybrid trees: one used an MOBB node for the root and UMOBB nodes for the rest of the tree; while the other used a UMOBB node for the root and MOBB nodes elsewhere. These are listed as "Hybrid 1" and "Hybrid 2" in Table 1.

The experiments were performed on a PC running Windows with a 3.0GHz Athlon processor. Each experiment ran for 2 minutes of simulated time. Approximately 80% of all queries returned negative at the root node test, which is a sufficiently large percentage to make the fast but inaccurate MOBB trees' 2D OBB test perform very similarly to the more expensive UMOBB trees' 3D OBB test at the root node level (the results for "Level 1" testing). However, at nodes deeper in the tree the MOBB boxes improve in fit, and they perform better than UMOBB nodes due to their cheaper cost per test. The hybrid trees confirm this result. Overall then, in this environment it matters little which hierarchical method we use.

The short motions of our target environment limit the performance gains available through a hierarchical method. We see only around a factor of 2 improvement over a non-spatial (but still temporal) hierarchy. Longer motions provide greater performance improvements, so we conducted another experiment using a simulation style workload (in terms of percentage of positive tests) but in an isolated test environment.

Our second experiment used 14 motions with an average length of 42 seconds. We performed an identical set of 100000 tests with each style of tree, each test using a random translation, rotation and temporal offset on one of the motions. These tests were done on a 3GHz Pentium 4 PC running Linux. The results are in Table 2, and an example motion appears in Figure 8. On longer motions, MOBB trees perform best by a small margin, and the almost identical per-

Method	Time (10^{-6}s)	# 2D	# 3D	# sample
MOBB 1 level	68.3	0.83	0	306
UMOBB 1 level	68.1	0	0.83	307
MOBB	5.9	27.5	0	4.15
UMOBB	6.8	0.62	21.6	3.10
Hybrid 1	6.8	1.45	20.8	3.10
Hybrid 2	6.0	26.7	0.83	4.15

Table 2: *Results for our experiment using longer motion clips. The methods are described in the text. The table shows average time per intersection query, the number of 2D and 3D OBB tests performed, and the number of sample-sample overlap tests. We see MOBB trees slightly out-performing the unrestricted tree, reflecting the relatively high cost of 3D OBB tests compared to 2D tests.*

formance of the "Hybrid 2" trees (containing almost all MOBB nodes) supports the conclusion that faster tests with looser MOBB bounds are preferable in this application to the slower UMOBB tests. Regardless of the exact type of bounding tree, we consistently see roughly an order of magnitude improvement. Intuitively, the motions are long enough to allow pairs to frequently start nearby (meaning their root nodes overlap) and move away from each other (meaning that spatial testing is effective when temporal is not).

6 Discussion

We have presented a novel bounding volume methodology, Motion Oriented Bounding Box trees, for motion capture clips that exploits a spatio-temporal hierarchy. In practice, we found a restricted form of OBB, with one axis aligned with time, formed the most effective bound for motion data. Experimental tests confirmed that hierarchical bounds are more effective for longer motions – short motions produce trees that are too shallow. Hence, hierarchical bounds are most applicable in planning type applications where long sequences must be tested for intersection, rather than highly reactive environments in which clips are typically short. In the former situation, we saw an order of magnitude improvement in collision detection time, while in the latter case the fastest approach is likely to be a binary search on time for overlapping samples, followed by direct comparison of sample positions.

An extension we are exploring is the application of hierarchical bounds to sequences that are temporally combined at run-time, as occurs in motion graphs. Combinations of short clips obviously produce longer ones, and hence suit our technique. It is insufficient to simply test each short segment; better performance should result from combining small trees from the bottom up into larger trees. Advances in this area will result in more realistic simulation of interactive characters, and hence more engaging virtual worlds.

References

ABDEL-MALEK, K., BLACKMORE, D., AND JOY, K. 2002. Swept volumes: Foundations, perspectives, and applications. *International Journal of Shape Modeling*. Submitted.

BASCH, J., ERICKSON, J., GUIBAS, L. J., HERSHBERGER, J., AND ZHANG, L. 2004. Kinetic collision detection for two simple polygons. *Computational Geometry 27*, 3, 211–235.

EBERLY, D. 2001. *3D Gamem Engine Design: a practical approach to real-time computer graphics*. Academic Press.

FOISY, A., AND HAYWARD, V. 1993. A safe swept volume method for collision detection. In *The sixth international Symposium of Robotics Research*, 61–68.

GLEICHER, M., SHIN, H., KOVAR, L., AND JEPSEN, A. 2002. Snap-together-motion. In *Proceedings of ACM SIGGRAPH 2002 Symposium on Interactive 3D Graphics*, ACM SIGGRAPH.

GOTTSCHALK, S., LIN, M. C., AND MANOCHA, D. 1996. OBB-Tree: A hierarchical structure for rapid interference detection. *Computer Graphics 30*, Annual Conference Series, 171–180.

KIM, D., KIM, H. K., AND SHIN, S. Y. 2003. An event-driven approach to crowd simulation with example motions. Tech. Rep. CS/TR-2003-186, KAIST.

KIM, Y. J., VARADHAN, G., LIN, M. C., AND MANOCHA, D. 2003. Fast swept volume approximation of complex polyhedral models. In *Proceedings of the eighth ACM symposium on Solid modeling and applications*, 11–22.

MIRTICH, B. 1996. *Impulse-based Dynamics for Rigid-Body Simulation*. PhD thesis, University of California, Berkeley.