군중 시뮬레이션을 위한 그래프기반 모션합성에서의 충돌감지

(Detecting Collisions in Graph-Driven Motion Synthesis for Crowd Simulation)

성 만 규 †

(Mankyu Sung)

요 약 본 논문에서는 모션캡처데이타를 이용한 두 캐릭터간의 빠른 충돌감지에 대한 연구를 논의한다. 본 연구의 목적이 군중 시뮬레이션이기 때문에, 제안한 알고리즘은 캐릭터를 실린더 형태로 모델링 한 후에 Rough한 충돌감지를 목표로 한다. 이를 위해 계층적인 바운딩 박스 데이타 구조인 MOBB를 제안한다. MOBB는 모션클립에 대한 시공간 바운딩 박스이며, 제안된 알고리즘에 대한 테스트 결과 2배 이상의 속도 향상이 있음을 밝힌다.

키워드 : 군중시뮬레이션, 충돌감지, 모션그래프, 모션캡쳐

Abstract In this paper we consider detecting collisions between characters whose motion is specified by motion capture data. Since we are targeting on massive crowd simulation, we only 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.

Key words : Crowd simulation, Collision Detection, Motion graph, Motion Capture

1. 서 론

Virtual human characters are an essential part of many interactive environments, such as games, visualizations, and simulations. Motion capture data has come to dominate human skeletal animation due to its realism and maturing capture and editing technologies. However, motion capture alone does not ensure the realism of character interactions that arise when clips are assembled into longer sequences and many agents are animated in the same space. Minimal requirement in such cases is that characters do not interpenetrate.

This paper presents algorithms for identifying potential collisions between motion clips, and hence animated characters. Our assumption is that two motion clips to be tested should be known in advance before testing, which means that we should know how long two motions are in time and trajectory of motion path in space. Current motion capture data has such information in their particular format in general [1]. This assumption gives long time step for testing collision since we don't need to check collision at every frame while doing a motion once the motion clip is identified as collision free. This is a particular case for the graph based motion synthesis where the long motions are synthesized by concatenating discrete set of short motions [2-5]. In case of synthesizing motion with motion blending technique [6,7], it's not possible to get to know motion in advance, which

† 정 회 원 : 한국전자통신연구원 디지털액터팀 선임연구원
mksung@etri.re.kr
논문접수 : 2007년 9월 27일
심사완료 : 2007년 11월 22일

정보과학회논문지 : 시스템 및 이론 제35권 제1호(2008.2)

means that we need to test collision at every time step. The long time step is the most important in real time simulation of large number of character.

Our primary target application is crowd animation, where many agents occupy the same environment and move in relatively close proximity. In such a space, we are not concerned with the fine grained interactions of agents (such as a hand shake) but rather with ensuring that the agents maintain reasonable separation as they move. For instance, the locomotion scene of crowds only need to check whether or not current motion clips cause any bumping between characters. Hence, we seek intersections between vertical bounding cylinders placed around the agent at each time-step (Figure 1, left). If close-in interactions are required, our algorithm could be used as an intermediate stage to identify potentially interacting agents. For example, if we want to check fine-grained collisions between characters who are fighting each other with a sword in a battle scene, then we can use our algorithm as a first step for fast checking of collision between the whole characters. Because our algorithm deals with a character holding sword as a whole, it reports a possible collision immediately. After we figure out potential part of motion data that cause collision through our algorithm, we can further investigate which part of body is actually cause collision. In our experiment, we use motion data obtained from a variant of motion graphs (e.g. [3]) in which each executes a sequence of uninterruptible clips separated by choice points where the next motion is decided. At each choice point, given a set of potential next clips, the collision detection system must identify those that will result in collisions with other agents so that they can be removed from consideration.

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 [8] 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

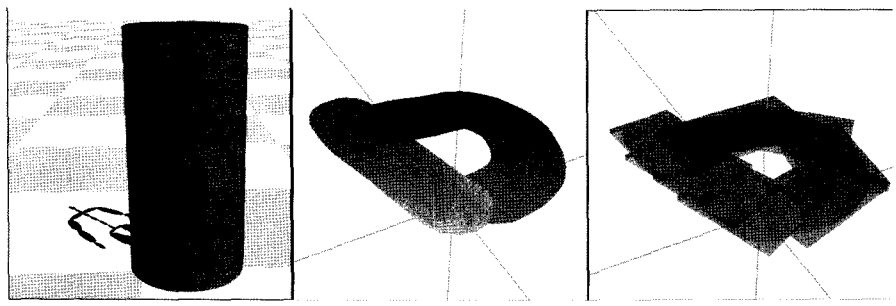

Fig. 1 On the left, a sample frame of motion data from a clip used in our system. The skeleton is bound by a vertical cylinder centered on its root position with a radius large enough to contain the limbs. Center is a 3D space-time visualization of two motions that do not intersect in time (vertical 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.

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 [8] to swept volume intersection in space-time. OBB trees have been applied to continuous collision detection in the past [9], [10], 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 [11]) 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 [12] for rigid body examples). Kim et. al. [13] 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 [14] 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. [15] for a survey). Recent advances include work aimed at complex models [16], but the most similar approach to ours is due to Foisy and Hayward [17]. 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. [18] 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.

3. MOBB TREES

The path of a character consists of a sequence of (x, y, t) samples, ordered on time. We assume, without loss of generality, that the first sample is at time $t = 0$. It is convenient to think of these as samples from a function m(t): $R \rightarrow R^2$. Each sample is obtained by projecting the root node of a motion capture frame onto the ground plane. Associated with each sample is the radius of the bounding cylinder for that frame, $r(t)$. In our experiments, the $r(t)$ is a maximum 2D distance from root to all projected joint positions. Thus, it's variable value depending on motions. Even if character is holding some object such as sword, we can take account those easily by projecting those object onto ground and deal with that as another joint.

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,

$$(t_{start}, t_a) \leq o_a < t_{end,a}$$
$$(t_{start}, t_b) \leq o_b < t_{end,b}$$

such that

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

In practice, the time offsets could be arbitrary real numbers, while the samples are discrete. We therefore interpret $m_a(o_a)$ to be the sample from the time closest to but below o_a. An alternative is to interpolate 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 $(m(t),t)$, 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 interpolation scheme would result in swept disks in space-time.

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 $(t_{max} - t_{min})/2$ and $(t_{max} - t_{min})/2$ to t_{max}. In other words, the hierarchy is built by subdividing the volume at its midpoint perpendicular to the time dimension.

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.

The major axis of the OBB tree, a0, is found by subtracting the location of the first sample, $\mathrm{m}(t_{min})$ from that of the last, $\mathrm{m}(t_{max})$, and normalizing:

$$a_0 = \mathbf{m}(t_{max}) - \mathbf{m}(t_{min}) \;/\; \| \mathbf{m}(t_{max}) - \mathbf{m}(t_{min}) \|$$

The minor axis is perpendicular: $a_1 = (-a_{0,y}, a_{0,x})$.

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 d_{max} and d_{min} are the largest and smallest extents found, then the center of the OBB is at

$$C_{local} = (d_{max} + d_{min}) / 2$$

and the dimensions of the box are

$$d = (d_{max} - d_{min}) / 2$$

The center is transformed back into global coordinates and stored, along with a_0, a_1 and d. Each node also stores t_{max} and t_{min}.

After computing the bound at one node, we divide the sample sequence into two equally sized sub-sequences and recurse. 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 $n_{min}/2$ sample distance tests.

Standard OBB tree construction algorithms [8] 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

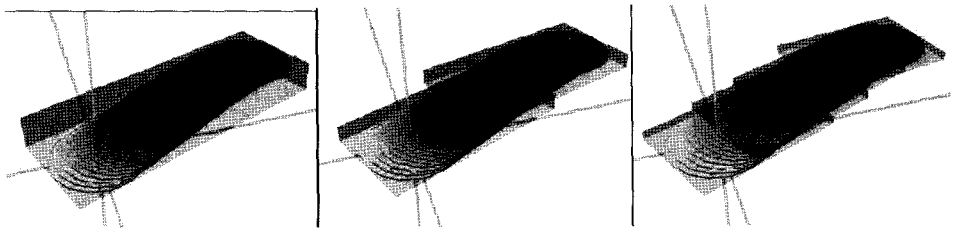

Fig. 2 Three levels in an MOBB tree hierarchy. Boxes are split evenly in the time (vertical) dimension going from one level to the next, and then spatial bounds are generated containing all the samples in that time-slice

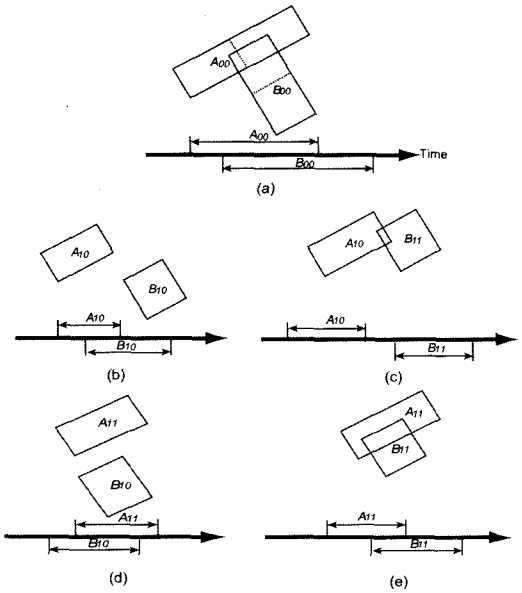

Fig. 3 Testing two internal MOBB tree nodes, A00 and B00. The relative spatial and temporal arrangement of the nodes is shown in (a). The algorithm first tests for temporal overlap, and then spatial overlap. In this case there is an intersection, so the algorithm recurses with the four combinations of 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

world coordinates, T_a and 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.

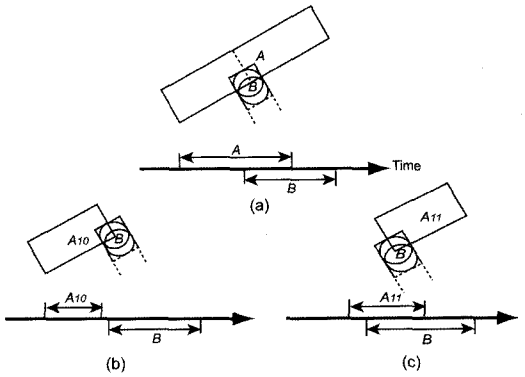

Fig. 4 Testing an internal node against a leaf node. In (a), we have reached a leaf node B while at a non-root node A in the other tree. The time range of B is tested against A's children, A10 and A11. In one case, (b), the time interval does not overlap, so we stop with no intersection found. In the other case, (c), a temporal overlap is found so the algorithm recurses with A11 and B. Note that we do not perform a spatial test in case (c); experiments found it gave no advantage

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. 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.

4. UNRESTRICTED MOBB TREE

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-

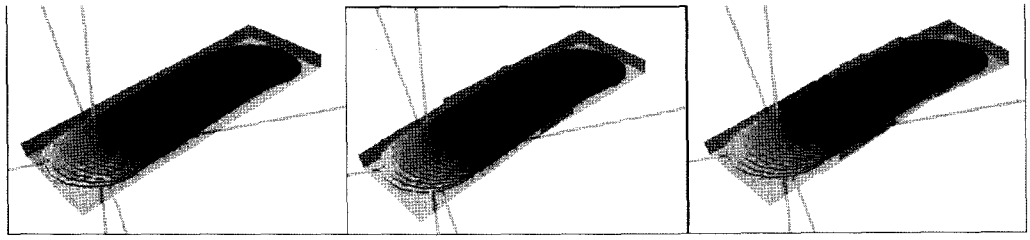

Fig. 5 Three levels in an unrestricted MOBB (UMOBB) tree hierarchy. In this tree, bounding volumes are 3D OBBs in space-time, removing the restriction that one axis align with the time dimension. Boxes are still split evenly in the time dimension, but a 3D OBB is fitted to the samples. UMOBB trees have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

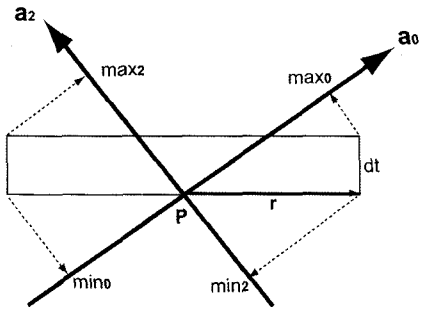

Fig. 6 Computing the extent of a sample disk for a 3D space-time OBB. The disk has the center of its base located at the sample point, p, with radius vector r and height dt. We must find the minimal and maximal projections onto the axes a0 and a2. Note the asymmetry in the cylinder's position.

time, we determine the first axis, a_0, by subtracting the space-time location of the first sample, $(m(t_{min});t_{min})$ from that of the last, $(m(t_{max});t_{max})$ and normalizing (similar to the 2D case, but including the time dimension). We obtain a second axis, a_1, mutually orthogonal to a_0 and the time dimension, (0,0,1). Finally, $a_2 = a_0 \times a_1$.

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 $\boldsymbol{r}$, which is the vector with length equal to the disk's radius, r, aligned with a_0 in the xy-plane:

$$\boldsymbol{r} = r\ (a_{0,x},\ a_{0,y},0)\ /\ \|\ (a_{0,x},a_{0,y},0)\ \|$$

From the figure, we see that

$$min_0 = -\ \boldsymbol{r} \cdot \mathbf{a}_0$$
$$min_2 = \boldsymbol{r} \cdot \mathbf{a}_2$$
$$max_0 = \boldsymbol{r} \cdot \mathbf{a}_0 + dt\ \mathbf{a}_{0t}$$
$$max_2 = -\boldsymbol{r} \cdot \mathbf{a}_2 + dt\ \mathbf{a}_2$$

The extents in the remaining direction, a1, 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.

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 51 Snap-Together Motion [3] style motions, which guarantee visual continuity when we connect them. 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.

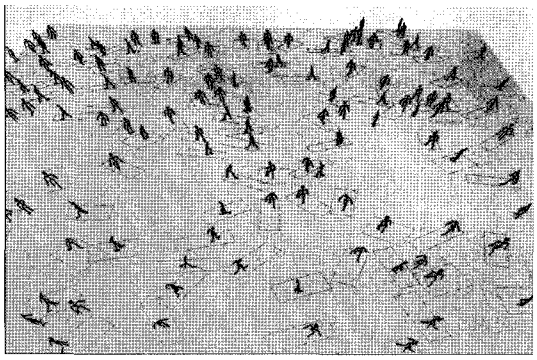

Fig. 7 Snapshot of the crowd simulation used to explore the performance of MOBB trees. The environment is a 30£40 meter rectangle containing 100 agents

This limits to some extent the benefits we see from a hierarchical method. The average radius of the bounding sphere around each agent was r = 185 meters.

Since we are not targeting to get reasonable behaviors from crowds in collision performance test, we just make them select one motion among 51 motions randomly in this experiment.

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 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. More specifically, in "Hybrid 1", the algorithm use MOBB root nodes for collision test first. If it overlapped, then rest testing is done by UMOBB tree. On the other hand, the "Hybrid 2" approach uses UMOBB tree for initial testing and then use MOBB for further testing.

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

Method	Time(10-6s)	#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

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 performance 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.

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

Method	Time(10-6s)	#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

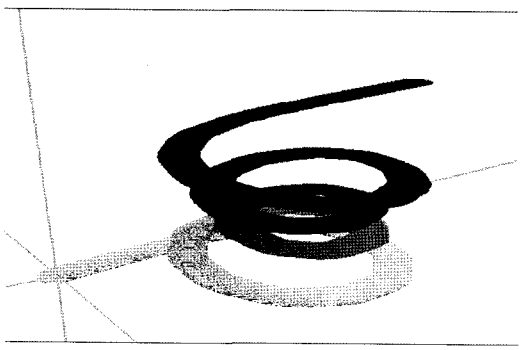

Fig. 8 One of the long motions used in our experiments. The motion clip is of someone walking in circles

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 exploit 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. However, even though we only get factor of two performance improvement when we use relatively short long motions, the total amount of time to spend for testing collision between two characters is very critical in realtime application if we want to simulate large number of crowd.

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

[1] Jaff Lander. Working with motion capture file format. *Game Developer*, January:30 - 37, 1998.

[2] Okan Arikan and D.A. Forsythe. Interactive motion generation from examples. In *Proceedings of ACM SIGGRAPH 2002*, Annual Conference Series, July 2002.

[3] M. Gleicher, H. Shin, L. Kovar, and A. Jepsen. Snaptogether-motion. In *Proceedings of ACM SIGGRAPH 2002 Symposium on Interactive 3D Graphics*. ACM SIGGRAPH, 2002.

[4] Lucas Kovar, Michael Gleicher, and Fred Pighin. Motion graphs. In *Proceedings of ACM SIGGRAPH 2002*, Annual Conference Series, July 2002.

[5] Jehee Lee, Jinxiang Chai, Paul S. A. Reitsma, Jessica K. Hodgins, and Nancy S. Pollard. Interactive control of avatars animated with human motion data. In *Proceedings of ACM SIGGRAPH 2002*, Annual Conference Series, July 2002.

[6] S. I. Park, H. J. Shin, and S. Y. Shin. On-line locomotion generation based on motion blending. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium onComputer Animation 2002*, July 2002.

[7] C. Rose, M. Cohen, and B. Bodenheimer. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Application*, 18(5):32-40, 1998.

[8] S. Gottschalk, M. C. Lin, and D. Manocha. OBBTree: A hierarchical structure for rapid interference detection. *Computer Graphics*, 30(Annual Conference Series):171-180, 1996.

[9] S. Redon, A. Kheddar, and S. Coquillart. Fast continuouscollision detection between rigid bodies. In *Proceedings of Eurographics Conference*, 2002.

[10] S. Redon, Y.J Kim, M.C. Lin, and D. Manocha. Fast continuous collision detection for articulated models. In *Proceedings of ACM Symposium on Solid Modeling and Applications*, 2004.

[11] David Eberly. *3D Gamem Engine Design: a practical approach to real-time computer graphics*. Academic Press, 2001.

[12] Brian Mirtich. *Impulse-based Dynamics for Rigid-Body Simulation*. PhD thesis, University of California, Berkeley, 1996.

[13] Dohan Kim, Ho Kyung Kim, and Sung Yong Shin. An event-driven approach to crowd simulation with example motions. Technical Report CS/TR-2003-186, KAIST, 2003.

[14] Julien Basch, Jeff Erickson, Leonidas J. Guibas, John Hershberger, and Li Zhang. Kinetic collision detection for two simple polygons. *Computational Geometry*, 27(3):211-235, 2004.

[15] Karim Abdel-Malek, Denis Blackmore, and Kenneth Joy. Swept volumes: Foundations, perspectives, and applications. *International Journal of Shape Modeling*, 2002. Submitted.

[16] Young J. Kim, Gokul Varadhan, Ming C. Lin, and Dinesh Manocha. Fast swept volume approximation of complex polyhedral models. In *Proceedings of the ETRI Journal, submitted in 2006 7 eighth ACM symposium on Solid modeling and applications*, pages 11-22, 2003.

[17] A. Foisy and V. Hayward. A safe swept volume method for collision detection. In *The sixth international Symposium of Robotics Research*, pages 61-68, 1993.

[18] S. Redon, Y.J Kim, M.C. Lin, D. Manocha, and J. Templeman. Interactive and continuous collision detection for avatar in virtual environments. In *Proceedings of Virtual Reality Conference 2004 (VR)*, pages 117-283, March 2004.

성 만 규

1989년~1993년 충남대학교 전산학과(학사). 2000년~2005년 Univ. of Wisconsin-Madison, Computer Sciences(석사, 박사). 1995년~2000년 한국전자통신연구원, 연구원. 2006년~현재 한국전자통신연구원, 선임연구원