Efficient Maintenance and Self-Collision Testing for Kinematic Chains

Itay Lotan
Dept. of Comp. Science
Stanford University
Stanford, CA 94305
itayl@stanford.edu

Fabian Schwarzer
Dept. of Comp. Science
Stanford University
Stanford, CA 94305
schwarzf@stanford.edu

Dan Halperin
School of Comp. Science
Tel Aviv University
Tel Aviv 69978, Israel
danha@post.tau.ac.il

Jean-Claude Latombe
Dept. of Comp. Science
Stanford University
Stanford, CA 94305
latombe@cs.stanford.edu

ABSTRACT

The kinematic chain is a ubiquitous and extensively studied representation in robotics as well as a useful model for studying the motion of biological macro-molecules. Both fields stand to benefit from algorithms for efficient maintenance and collision detection in such chains. This paper introduces a novel hierarchical representation of a kinematic chain allowing for efficient incremental updates and relative position calculation. A hierarchy of oriented bounding boxes is superimposed on of this representation, enabling high performance collision detection, self-collision testing, and distance computation. This representation has immediate applications in the field of molecular biology, for speeding up molecular simulations and studies of folding paths of proteins. It could be instrumental in path planning applications for robots with many degrees of freedom, also known as hyper-redundant robots. A comparison of the performance of the algorithm with the current state of the art in collision detection is presented for a number of benchmarks.

Categories and Subject Descriptors

F.2.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity—*Nonnumerical Algorithms and Problems, Geometrical problems and computations*; I.3.5 [**Computing Methodologies**]: Computer Graphics—*Computational Geometry and Object Modeling*; J.3 [**Computer Applications**]: Life and Medical Sciences—*Biology and genetics*

SoCG'02, June 5-7, 2002, Barcelona, Spain.

General Terms

Algorithms, Experimentation, Performance

Keywords

Collision Detection, Modelling Kinematic Chain, Bounding Volume Hierarchy

1. INTRODUCTION

A kinematic chain is a common and useful representation of a manipulator arm or a hyper-redundant robot in robotics [6], as well as of a protein backbone in molecular biology [7]. It is a serial linkage composed of N links and $N-1$ joints. A kinematic chain may alter its configuration over time by changing its degrees of freedom while adhering to constraints imposed by the chain-like structure. The problem at hand is the efficient maintenance of a representation of the chain in order to efficiently answer queries — after every time step — about the self-collision status of the chain, the existence of collisions with obstacles, or the minimum distance from other objects.

Current approaches to collision detection do not deal specifically with chains but rather propose general methods for either rigid objects or objects composed of separately moving pieces. These approaches can be divided into two main classes of algorithms, one that tracks features in order to compute minimal separation, and the second that uses a subdivision and approximation of the object itself or the space it lives in. Examples of feature based approach are the Lin-Canny collision checker [19] and the KDS of Agarwal et al [1]. The partition methods include occupancy grids such as [8, 13, 15] or the one suggested by Halperin et al [12] for kinematic chains. Other partitions are based on projection onto subspaces [5], or octtrees [9]. Much work has dealt with a hierarchy of approximations based on bounding volumes (BVs) to describe each object. Different types of BV's have been used such as spheres [3, 14], axis aligned bounding boxes (AABB) [23], oriented bounding boxes (OBB) [10], rectangular swept spheres (RSS) [18], and discrete orientation polytopes (K-DOP) [17].

We could not build on the feature tracking approaches because they assume simple convex polyhedra or require temporal coherence (the position of a link at time $t+\Delta t$ is close to its position at time t, where Δt is the time-step taken by the algorithm). Unfortunately a chain with unbounded motion does not have these two properties. All the space partition approaches except [12] only detect collision between two separate objects and fail to exploit properties specific to a chain. The work in [11] tackles a problem similar to ours. However, while we aim to explore the configuration space of the chain, through controlled changes to a limited number of degrees of freedom, they are interested in the physical simulation of chain-like structures, and therefore must assume all links move simultaneously.

There are two specific properties of the chain that we capitalize on in our approach. First, in a kinematic chain local changes have global effects, namely, a change in a joint alters the positions of all links beyond this joint relative to those located before. If we maintain a single reference frame for the entire chain, we need to update the position of $O(n)$ links every time a change is applied to one of the joints. Therefore, we create and maintain a hierarchy of reference frames and achieve a much lower update cost. The second property becomes apparent when we attempt to detect self-collisions. Current algorithms [10, 14, 17, 23] are only geared towards collisions between two separate objects. They could be applied to finding self-collisions by testing an object against a copy of itself. While this approach would yield correct results, it is very wasteful. The algorithm is bound to find the “trivial” collisions between a part of the object and its copy, as well as the “interesting” collisions between different parts of the object. Our new approach overcomes this inherent inefficiency by using the topology of the chain to locate rigid pieces that are known to be collision-free and avoid testing them for self-collisions.

In the biological domain, macro-molecules are described as long chain-like structures with up to several thousand degrees of freedom (DOFs). A useful representation of these molecules is hence a kinematic chain whose links are the rigid pieces and whose joints are the bonds that allow torsional motion. One particularly interesting class of such macro-molecules are the proteins. A protein uses its DOFs to change its *conformation*[1] in order to minimize its free energy. The active (folded) conformation of a protein is the global free energy minimum [7]. Finding this state is one of the fundamental problems in molecular biology. Methods aimed at finding the folded conformation require computation of internal energy as changes are applied to the DOFs of the protein. One class of such methods, known as Monte-Carlo methods, applies small changes to the molecule at every time step and accepts or rejects the new conformation based on the change in energy. This method would benefit tremendously from efficient self-collision detection because steric clashes entail prohibitively high energy. We would need to compute the other, more expensive components of the energy function only for clash-free conformations.

In classical robotic motion planning, random sampling is a well established technique [2, 16]. A roadmap graph of free-space configurations is created and searched for a path from the initial state to the goal. Many planners use the distance between the obstacles and the robot placed at a milestone to decide which pairs of milestones in the roadmap can safely be connected by straight paths. Planners using potential field methods also rely heavily on calculating distances from obstacles. These planners spend most of their time computing distances and detecting collisions and would benefit greatly from speeding up these operations. Our new approach is particularly suited for hyper-redundant snake-like robots [4, 20, 24]. Self-collisions could be detected efficiently as the robot changes its configuration and the distance of the robot from any obstacle expeditiously determined.

Our contributions in this paper are:

- A novel hierarchical representation of a kinematic chain, which allows for $O(\log N)$ time incremental updates for a constant number of changes, and $O(\log N)$ time calculation of the relative position of two arbitrary links.
- A bounding volume hierarchy (BVH) respecting the chain topology is built on top of this representation. It can be used to detect self-collision in a worst case tight bound of $\Theta(N^{\frac{4}{3}})$ time, but is shown to do much better in practice; far better than any existing approach.

While the idea of constructing a hierarchy that respects the chain topology seems natural and obvious, theory does not a priori encourage this approach. We can maintain a standard spatial representation of the chain (respecting the spatial arrangement of the links regardless of the kinematics) that can be updated *and* tested for self-collision in time $O(N \log N)$ per step. This seems impossible to beat. However our results below, both the experimental and theoretical, show that the issue is more involved. We show that efficient update is crucial for the effectiveness of the overall solution. Indeed our solution updates the chain in $O(\log N)$ time per change in a single joint. Furthermore, we show that the efficient update comes with a moderate price: self-collision testing now goes up to $O(N^{4/3})$ (which is higher than $O(N \log N)$ but lower than the cost incurred per step by a naïve approach). Finally, what we see in the experiments is that this bound is a crude, conservative estimate of the cost of testing self-collision and in practice, even in highly packed situations, the cost is typically much smaller

The rest of the paper is constructed as follows: We start by presenting our approach in Section 2: first (Section 2.1) we explain how the incremental maintenance of a chain works; then (Section 2.2) we show how to construct an efficiently updateable BVH on top; and finally (Section 2.3) we present an efficient algorithm for detecting a self-collision using the representation. In Section 3 we analyze the performance of our algorithm for updating the representation (Section 3.1) and for detecting self-collision (Section 3.2). The proofs of the worst case complexity of self-collision detection in kinematic chains are given in Section 4 for two types of BVH. Section 5 presents experimental results showing the superiority of our novel approach to the current methods, and Section 6 discusses extensions to the algorithm and future work.

2. ALGORITHM DESCRIPTION

We are given a kinematic chain which may deform along its degrees of freedom at each time step. We would like to update the chain representation and the BVH built on top of it to reflect the latest set of changes and then use the

[1] The chemical term for configuration of a molecule.

hierarchy to test the chain for self-collision. In what follows we expand on these tasks.

2.1 Incremental Maintenance

A chain is a series of N rigid links $L_0, \ldots, L_{N-1}$ connected by $N-1$ joints, such that $J_{i,i+1}$ is the joint connecting L_i and L_{i+1}. Each link is described in terms of its own reference frame. In our representation a joint is simply a rigid transformation between two reference frames given as a rotation matrix R and a translation vector $\mathbf{t}$. For example, $J_{i,i+1}$ is the transformation from frame L_{i+1} to frame L_i. We also maintain a hierarchy of *shortcut* transformations that allow us to move faster along the chain.

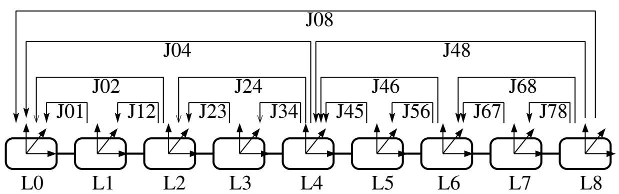


Figure 1: The cached transformations scheme for a kinematic chain.

In Figure 1 we can see the entire set of transformations that are cached for a chain of 9 links. For every $l = 0, \ldots, \log N$, for every $0 \leq i \leq N-1$, if $i \ mod \ 2^l = 0$ then we store the transformation $J_{i,i+2^l}$.This scheme results in $\log N$ levels of shortcuts, with level l having $\frac{N}{2^l}$ cached transformations, each looking ahead 2^l links in the chain. Altogether at most $2N-3$ transformations are cached. This scheme allows for the computation of the transformation between the coordinate frames of two arbitrary links L_i and L_j in $O(\log N)$ time. It will also prove useful when testing two bounding boxes in the hierarchy for possible overlap. It is important to note that we do not explicitly maintain the global position of any of the links. Should we want the coordinates of all links, we can compute them in the frame of L_0 by traversing the chain and applying the cached transformations to the links.

Changes can be applied simultaneously to all joints of the chain. Each change occurs at some joint and is defined as a rigid transformation $C = \{R, \mathbf{t}\}$ at this joint. To keep our representation up-to-date, it is necessary to recompute some shortcut transformations, and this is done one level at a time. At each level we recompute all transformations that shortcut a transformation that was updated on the previous level. Recomputing a shortcut $J_{i,i+2^l}$ at level l amounts to computing the product of the two transformations $J_{i,i+2^{l-1}}$ and $J_{i+2^{l-1},i+2^l}$ at level $l-1$. To understand this better we look again at Figure 1. We would like to bend the chain at joint $J_{3,4}$. This entails applying some rotation to the current cached value of $J_{3,4}$. We now need to see which shortcut transformations need to be recomputed. At the second level, $J_{2,4}$ is no longer correct. We compute it by taking the product of the two transformations it shortcuts, namely $J_{2,3}$ and $J_{3,4}$. In the same way we recompute $J_{0,4}$ as the product of $J_{0,2}$ and $J_{2,4}$, and $J_{0,8}$ as the product of $J_{0,4}$ and $J_{4,8}$.

2.2 Hierarchy of Bounding Volumes

A BVH is needed for performing efficient collision testing. Before building such a hierarchy we need to decide what type of BV to use. There is a trade-off between the tightness with which the volume bounds the geometry and the complexity of testing two such volumes for overlap [10]. For our applications with macro-molecules such as proteins, OBBs seem best suited, since they offer relatively tight bounding for elongated geometry as well as for globular structures, while still allowing efficient overlap tests (as described in [10]). Proteins can fold up to very compact conformations and tight BVs are expected to dramatically reduce the number of overlap tests required to detect self-collision.

The different collision detection algorithms proposed in the literature vary in the amount of deformation the objects may undergo. The sphere hierarchy in [14] and the OBB hierarchy in [10] allow only rigid transformations to the object and no deformation. The AABB hierarchy in [23] and the sphere hierarchy in [3] allow local deformations that remain small throughout the execution, while the oct-tree of [9] can cope with small local deformations having large aggregated effects on the shape of the object. All of these algorithms, however, are poorly suited for dealing with kinematic chains. *A change to a joint angle may bring together pieces of the chain that were previously far apart and separate pieces that were close together.* Adjusting the existing structure to reflect the change in the geometry would inevitably result in a poorly performing hierarchy for [3, 10, 14, 23], or prohibitively long update time for [9]. On the other hand, recomputation of the hierarchy from scratch is a costly procedure.

We therefore propose a different method of building the BVH that respects the topology of the chain. In particular, each box will bound a consecutive set of links along the chain. At the bottom of the hierarchy are the links of the chain, which we will assume to be simple objects, like spheres. (In Section 6.2, in the context of amino acid side-chains, we will see how to handle more complex elementary pieces.) At this level, a box is assigned to every link. Respecting the chain topology entails a simple bottom up system for constructing the hierarchy. When creating a new level, the boxes of the current level are divided into adjacent pairs and a new box is created for each pair, bounding both boxes. We mark a bounding box $B_{i,j}$ where i is the level of the box and j is its index starting from the box containing L_0 at that level. Using this notation, box $B_{i,j}$ is constructed to bound boxes $B_{i-1,2j}$ and $B_{i-1,2j+1}$. This also entails that box $B_{i,j}$ bounds links $L_{j2^{i-1}}$ to $L_{(j+1)2^{i-1}-1}$. Due to our construction method, the hierarchy is guaranteed to be balanced. An example for a short chain can be seen in Figure 2. Note that rather than bounding the underlying geometry tightly, a box bounds only the two boxes immediately below it in the hierarchy.

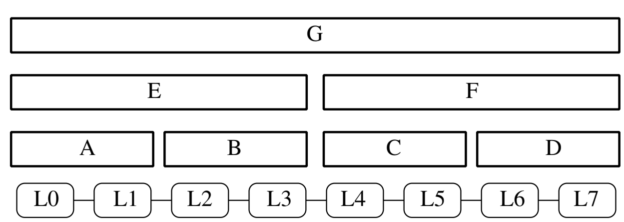


Figure 2: A hierarchy of bounding boxes respecting the chain's topology.

Whenever a change is applied to the chain, all bounding boxes that contain the affected joint need to be updated to make sure they still bound the geometry correctly. This is a simple bottom up process that proceeds along the path from the lowest level box containing the changed joint, to the box enclosing the whole chain. At each level a box is recomputed to bound the two boxes just below it. For example, in Figure 2 we would like to apply a change to the joint between L_4 and L_5. Before we update the boxes we must update the shortcut transformations as described in Section 2.1. The lowest level box that contains this joint is C. We compute a new box C' to replace C, bounding L_4 and L_5 (see Figure 3). At the next level F is no longer valid, and we compute F' to bound C' and D. At the top level, the root box G is also recomputed. When simultaneous changes are applied to the chain, a corresponding number of paths need to be updated. Instead of updating one path at a time, we update all paths together, one level at a time. First, we update all shortcuts at the next level and then all boxes at that level. This allows us to avoid multiple updates of the same box or shortcut when update paths converge.

2.3 Detecting Self-Collisions

The main purpose of our algorithm is the detection of self-collisions. As explained in the Introduction, the immediate way to do this would be to test the hierarchy of the chain against a copy of itself. We would start by testing the top level box against itself and then continue down the hierarchy using the standard algorithm described in [10]. Since every elementary piece of the object (the links of the chain) occupies the same space as itself, the traversal of the hierarchy will visit all the links at a cost of $\Omega(N)$, independent of the chain configuration. At this complexity it would be better to use a space decomposition method like the grid, which has a much smaller overhead.

Assuming that the number of changes applied since the last query is a small constant, and that before applying the last set of changes the chain was collision-free, we can actually do better in practice. We observe that the joints at which the changes were applied, divide up the chain into rigid pieces that have not undergone any deformation since the last query. These "rigid" pieces, which are contained inside boxes that were unaffected by the last change, will still be collision free and there is no need to test within them. Therefore, during the traversal, when coming across a box that was not recomputed in the last update, we do not search its sub-hierarchy for self-collisions.

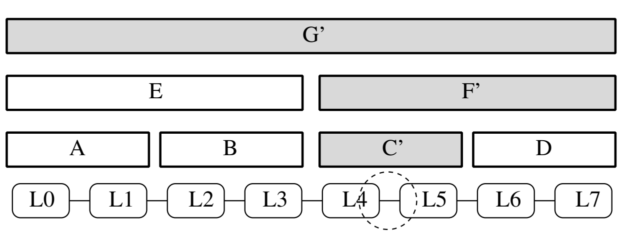


Figure 3: The hierarchy after applying a change.

Figure 3 illustrates a hierarchy after a change was applied to the joint between L_4 and L_5. The grayed boxes are the ones that required recomputing. Checking for self-collisions commences by testing the top box against itself. Since, box G' was affected by the last change we continue to descend. We now need to examine all pairs of its children, namely (E, E), (E, F') and (F', F'). The box E was not affected by the last change so the pair (E, E) can be discarded. Next we test the pair(E, F'). Since E and F' are separate branches of the tree, testing them is like testing two separate objects. If they overlap we continue to test their children, otherwise we quit that path. Lastly, we test the pair (F', F'). Since F' was affected by the last change we need to descend this path as well. This continues until a collision is found or all paths end without finding overlap. Effectively, we are testing for possible collisions between the two rigid pieces $\{L_0, \ldots, L_4\}$ and $\{L_5, \ldots, L_7\}$, but we are doing it implicitly and without computing a separate hierarchy for each rigid part.

3. ANALYSIS

In [10], Gottschalk et al suggest a cost function for evaluating the performance of a BVH for detecting collisions between two objects. Klosowski et al [17] expand the cost function by taking into account any updating costs incurred as the object flies through space. The cost of one collision test is given as:

$$T = N_v \times C_v + N_p \times C_p + N_u \times C_u, \quad (1)$$

where N_v is the number of BV pairs tested for overlap, C_v is the cost of one overlap test, N_p is the number of pairs of elementary pieces tested for overlap, C_p is the cost of an elementary pair overlap test, N_u is the number of nodes in the hierarchy that required updating and C_u is the cost of updating a node. This function applies also to the case of self-collision detection. In trying to minimize the cost function we need to take into account two fundamental and interdependent tradeoffs. The first is between C_v and $N_v + N_p$. The more complex the volumes are the tighter they fit the geometry and the less the number of overlap tests performed. The second is between updating $(N_u \times C_u)$ and testing $(N_v \times C_v + N_p \times C_p)$. Building a hierarchy that allows for efficient updating could curtail the performance of the collision checker. In what follows we explain the choices we made and their influence on the total cost T.

3.1 Updating

As we saw in Sections 2.1 and 2.2 updating the structure after changing one joint involves updating shortcut transformations and recomputing boxes along the path from the changed joint to the root of the hierarchy. Assuming we update the levels in order from the bottom up, at a given level a transformation is updated in $O(1)$ time. We are able to update a bounding box in $O(1)$ time as well by trading tightness for speed. Each bounding box in made to bound the two boxes below it, which results in not-so-tight bounding of the chain. Since the hierarchy has $O(\log N)$ levels, and at each level at most one transformation and one box are updated, the total cost of the update process is $O(\log N)$.

When κ simultaneous changes are applied, we update all changes together one level at a time. This ensures that no transformation nor box are updated more than once, since converging update paths are merged. We therefore have that $N_u \times C_u$ is $O(\kappa \log N)$ for a small κ, but never exceeds $O(N)$ as κ grows.

3.2 Detecting Self-Collision

For the sake of simplifying the analysis below and the proofs of worst-case bounds, which follow in Section 4, we will concern ourselves only with *well-behaved* chains. A well-behaved chain has the following two properties:

1. The ratio of the volume of the largest link to the volume of the smallest link is bounded. More precisely, we require that the ratio of the volume of the largest bounding sphere of any link to the smallest volume of any link is smaller than some constant β

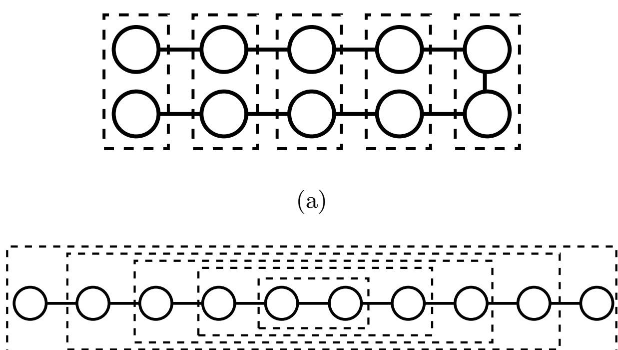
2. The centers of the minimal bounding spheres of no two links may come within a distance smaller than some constant δ

It was shown in [13] that in well-behaved chains each link may overlap no more than k other links, for a constant k. Consequently, there may be no more than $O(N)$ overlaps between the links of the chain at any given time. All kinematic chains described in the Introduction are well-behaved.

The immediate, brute-force algorithm for detecting self-collision would test all pairs of links for overlap, making $N_v = \Theta(N^2)$. Alternatively, we could use a BVH to speed up the computation. For a kinematic chain there are two types of BVHs that can be used:

Spatially-Adapted BVH: A hierarchy as described in [10, 17], which attempts a good spatial partition of a given configuration of the chain.

Chain-Aligned BVH: A hierarchy as presented in Section 2.2 that respects the topology of the chain.



(a)

(b)

Figure 4: A single joint change corrupting a spatially-adapted hierarchy. We examine only the second level of the hierarchy, where there are $\frac{N}{2}$ boxes. In (a) the boxes are spatially-adapted with no overlap. A change in the middle joint creates the configuration in (b). Recomputing the boxes for the new conformation while maintaining the topology causes all boxes to overlap.

For any given configuration of the chain one could construct a spatially adapted hierarchy. For such a hierarchy the following theorem holds (see Section 4.1 for a proof):

THEOREM 1. *The maximum number of BV overlap tests needed to detect self-collision in a well-behaved chain of N links using a spatially-adapted BVH is $\Theta(N)$.*

Hence N_v for a spatially adapted BVH is at the worst case $\Theta(N)$, but usually better than that. Although very efficient for detecting collisions in the configuration for which it was adapted, such a hierarchy is expensive to update. If we choose to maintain the current topology of the hierarchy, a single joint change could entail $N_u = \Theta(N)$ and corrupt the hierarchy, so as to raise N_v to $O(N^2)$ at the worst case. Such a scenario is illustrated in Figure 4. We could instead build a new hierarchy, adapted to the new chain configuration at a total cost of $O(N \log N)$ [10].

Although our BVH is not adapted for a good spatial partition of most chain configurations, the following theorem shows it maintains a good upper bound for N_v, which is tight in the worst case (see Section 4.2 for a proof):

THEOREM 2. *The maximum number of BV overlap tests needed to detect self-collision in a well-behaved chain of N links using a chain-aligned BVH of OBBs is $\Theta(N^{\frac{4}{3}})$.*

Moreover, its advantages over the spatially-adapted BVH are threefold. First and foremost, our BVH allows for an efficient $O(\log N)$ update scheme as described above, far superior to the $O(N \log N)$ effort needed to rebuild an adapted BVH. Second, the quality of our BVH (its worse case behavior) cannot be corrupted by any series of updates. It remains $O(N^{\frac{4}{3}})$ throughout the execution of the algorithm and does significantly better on the average (as the experiments reported in Section 5 show). Third, assuming a correlation of spatial distance and topological distance (distance along the chain), also known as *low contact order*, which exists in many proteins of interest to biologists [21], the worst case configurations become more rare. All in all, our approach yields a BVH that is not optimal but good enough most of the time, and, most importantly, allows for an efficient updating scheme.

4. WORST-CASE BOUNDS

All the proofs that follow will be based on the following constructions:

- When the links of the chain are not all spheres of equal radius, we replace each link by an enclosing sphere of radius r, where r is the radius of the bounding sphere of the largest link. Thus we may treat all links as spheres of equal radius.

- Consecutive links of the chain are touching and may even overlap. Each pair of consecutive links that does not touch is expanded until the links touch.

These constructions do not affect the asymptotic bounds proven below because they only change the size of the links by a constant factor.

In the proofs below we will distinguish between two different methods of constructing the BVH:

- The BVs of the hierarchy are constructed to bound tightly around the links of the chain they enclose, for all levels of the BVH. We refer to such a BVH as *tight* (this is the case when the BV is a minimal bounding sphere or a minimal OBB).

- When this is not the case and a BV is constructed to bound the two BVs just below it in the hierarchy, we refer to the BVH as *loose* (this is the case for the OBBs that we use).

4.1 Spatially-Adapted BVH

We would like to prove Theorem 1. First we show that $O(N)$ is an upper bound for such a hierarchy and then present a configuration, in which this bound is realized.

LEMMA 1. *The number of BV overlap tests needed to detect self-collision in a well-behaved chain of N links using a spatially-adapted, tight BVH is $O(N)$.*

PROOF. We prove this bound by constructing the hierarchy and showing that at each level there are $O(2^i)$ box overlaps ($i = 0, \dots, \log N$), which sum up to $O(N)$ for the entire hierarchy. We start by constructing an axis-aligned box A that bounds the entire chain. We will construct a recursive partition of A into *cells*, assign to each cell the links whose centers fall inside it, and then construct an AABB that will bound all these links. The partition into cells proceeds as follows: at each iteration we partition each cell in two using a plane orthogonal to one of the principal axes. The partition is chosen such that the centers of half the spheres fall on one side of it, and the centers of the other half fall on the other side. We also require that the cell be longer than $8r$ along the dimension which will be partitioned. Note that as long as there are $4k$ or more links assigned to a cell, there must be at least one dimension which is long enough to be partitioned. Recall that k is the constant limiting the amount of packing allowed in a well-behaved chain (See Section 3.2).

We recursively partition all cells in A into smaller and smaller cells until no more than $4k$ links are assigned to each cell. We say that the cells created at the ith iteration of the partitioning belong to the ith level of the hierarchy. We define n_{ij} to be the number of cells at level i whose minimal distance from cell j ($j = 0, \dots, 2^i - 1$) is less than $2r$. From the construction it is simple to show that $\sum_{j=0}^{2^i-1} n_{ij} = O(2^i)$. Each cell is represented in the hierarchy by an AABB constructed around the cell by increasing each of its dimensions by r in each direction. This AABB will obviously bound all links assigned to the cell. It will also overlap all cells within a distance of $2r$ from the cell, but nothing farther away since the side of each cell is at least $2r$. The number of neighbors of neighbors is also linear in the number of cells at each level so the total number of possible overlaps is bounded by $O(N)$. □

We now proceed to show a lower bound:

LEMMA 2. *The number of BV overlap tests needed to detect self-collision in a well-behaved chain of N links using a spatially-adapted, tight BVH is $\Omega(N)$.*

PROOF. This bound is straightforward and can be immediately deduced from the configuration in Figure 5. In this configuration we have two rows of spheres whose centers all lie on the same plane. Each sphere is tangent to all of its neighboring spheres. We note that no two spheres can be bounded inside a convex BV that does not overlap some other sphere in the configuration. This is obvious from the configuration. Therefore, at the level of the hierarchy, where we group pairs of spheres together into BVs, each BV is guaranteed to overlap at least one other BV. Since at this level there are $\frac{N}{2}$ BVs, the number of overlaps is clearly $\Omega(N)$, and the lemma is proven. □

Putting together the results of Lemmas 1 and 2 yields the proof of Theorem 1.

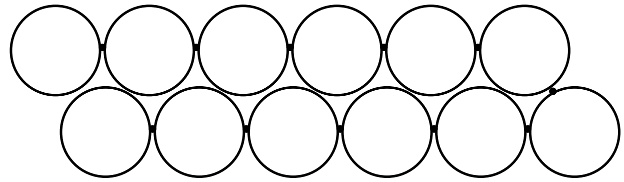


Figure 5: A 2D projection of a chain configuration achieving the $\Omega(N)$ lower bound.

4.2 Chain-Aligned BVH

We now turn to the proof of Theorem 2. We will first prove the upper bound and then present a chain configuration, for which the bound is attained for any convex BVH.

LEMMA 3. *Given a chain-aligned, loose BVH over a well-behaved chain as described in Section 2.2, the number of BV overlap tests needed to detect self-collision is $O(N^{\frac{4}{3}})$.*

In order to prove this lemma we first show that this bound holds for *tight* bounding spheres, and then proceed to show that the OBBs in our loose hierarchy can be bounded inside "well-behaved" spheres, giving our hierarchy the same upper bound.

LEMMA 4. *Given a chain-aligned and tight BVH of spheres over a well-behaved chain, the number of BV overlap tests needed to detect self-collision is $O(N^{\frac{4}{3}})$.*

PROOF. We construct a hierarchy bottom-up as described in Section 2.2 using spheres as BVs, making sure it is tight. Let's look at level i of the hierarchy, where each consecutive $g_i = 2^i$ links are bounded by a single sphere. The largest bounding sphere at level i has radius $g_i r$. It is obtained when the links are arranged on a straight line. We take one group of g_i spheres at level i, and let B_i be its bounding sphere. Consider a sphere C_i concentric with B_i and having radius $3g_i r$. Any bounding sphere at level i intersecting B_i is fully contained in C_i, since at the least it has to be tangent to B_i, and its maximum radius is $g_i r$. Now let's bound the number of groups whose bounding sphere may lie in C_i. Since we aim for an upper bound we assume that each such group is tightly packed. Nevertheless, there is a lower bound on the volume it occupies which is $q g_i \frac{4}{3}\pi r^3$, where q is a positive constant smaller than 1 due to the fact that each link can overlap at most k other links. Hence, from volume considerations the number of bounding spheres contained in C_i is at most:

$$M_i = \frac{\frac{4}{3}\pi 27 g_i^3 r^3}{\frac{4}{3} q g_i \pi r^3} = \frac{27 g_i^2}{q} \tag{2}$$

M_i is therefore the maximum number of bounding spheres that overlap B_i. We note that there are exactly $\frac{N}{g_i}$ bounding spheres at this level, so the number of spheres that intersect one sphere can only grow until it reaches $\frac{N}{g_i}$:

$$\frac{N}{g_i} \geq \frac{27 g_i^2}{q}$$
$$g_i \leq \frac{1}{3} q^{\frac{1}{3}} N^{\frac{1}{3}} \tag{3}$$

We recall that $g_i = 2^i$, and plug that into Equation (3) to find the level at which M_i reaches its maximal value:

$$i_{max} = \frac{1}{3}\log N + \log \frac{1}{3} q^{\frac{1}{3}} \qquad (4)$$

We define T_i to be the maximum number of possible BV (sphere) overlaps at level i of the hierarchy, where $i = 0, \ldots, \log N$ (0 is the bottom level). For all levels smaller than i_{max}, $T_i = \frac{N}{g_i} M_i$. For all levels greater than i_{max} we can use the trivial upper bound of $(\frac{N}{g_i})^2$ for the number of collisions at each level. In what follows we ignore the constant part of Equation (4) as it has no effect on the bounds we prove. The total number of collisions at all levels is therefore:

$$\begin{aligned} T &= \sum_{i=0}^{\log N} T_i \\ &= \sum_{i=0}^{\frac{1}{3}\log N} \left(\frac{27 g_i^2}{q}\right)\left(\frac{N}{g_i}\right) + \sum_{i=\frac{1}{3}\log N}^{\log N} \left(\frac{N}{g_i}\right)^2 \\ &= \frac{27N}{q} \sum_{i=0}^{\frac{1}{3}\log N} 2^i + N^2 \sum_{\frac{1}{3}\log N}^{\log N} \left(2^{-i}\right)^2 \\ &= \frac{27N}{q}\left(2N^{\frac{1}{3}} - 1\right) + \frac{4}{3}\left(N^{\frac{4}{3}} - 1\right) \\ &= O\left(N^{\frac{4}{3}}\right) + O\left(N^{\frac{4}{3}}\right) \\ &= O\left(N^{\frac{4}{3}}\right) \qquad (5) \end{aligned}$$

We have shown an upper bound of $O(N^{\frac{4}{3}})$ on the number of overlapping spheres. In a side note, we would like to draw the reader's attention to the fact that we could easily extend this proof to $d > 3$ dimensional links bounded by a d dimensional BVH of spheres, using the same proof with very minor changes. In that case we would get an upper bound of $O\left(N^{\frac{2(d-1)}{d}}\right)$. $\square$

The upper bound we have just proven in Lemma 4, would hold for all chain-aligned and loose BVH as long as the BV at level i can be bounded by a sphere of radius $c\,2^i r$, for an absolute constant c. In what follows we use that to show that the same upper bound holds for our OBB hierarchy as well. We start by proving the following helper lemma:

LEMMA 5. *Given two OBBs contained in a sphere D of radius R, the OBB bounding both of them is contained in a sphere of radius $\sqrt{3}R$ concentric with D.*

PROOF. Let the two boxes b_1 and b_2 be bounded inside the sphere D of radius R. Let B_{12} denote their OBB. Construct a bounding cube Q for sphere D, whose faces are parallel to those of B_{12}. Q contains B_{12} since along any of the main axes that define B_{12}, the faces of Q are farther out (or touching). Now take a bounding sphere of Q. Since a side of Q is of length $2R$, its diagonal has length $2\sqrt{3}R$, and hence the radius of the bounding sphere E of Q is $\sqrt{3}R$. E is concentric with D because along each of the axes defining B_{12} the farthest point of b_1 or b_2 from the center of D is at distance at most R, hence the farthest point of B_{12} from the center of D is at a distance at most $\sqrt{3}R$. $\square$

Next we prove another helper lemma:

LEMMA 6. *At level i, where 2^i links are grouped together inside one bounding box, each OBB can be bounded inside a sphere of radius $c\,2^i r$, where c is an absolute constant.*

PROOF. We prove this lemma by induction on the level of the hierarchy. For $i = 0, 1, \ldots, 4$ we verify this fact. Since there is a constant number of spheres involved, we compute the worst case scenario and extract a constant — c_1. We will take our constant c to be at least c_1. We assume the lemma is correct up to level $i-1$. Now consider a consecutive set of 32 OBB's b_j, $j = 0, \ldots, 31$, at level $i-5$ that are (eventually) bounded together at level i. The bounding sphere of each b_j has radius at most $c\,2^{i-5} r$ by the induction hypothesis. We take a ball S of radius $2^i r$ that contains the entire part of the chain bounded by the 32 b_j boxes. Assume for simplicity that it is centered at the origin. Now each sphere bounding one of the b_j boxes must intersect S or at least touch it. This means that the farthest point on a box b_j from the origin is at distance at most $2^i r(1+\frac{c}{16})$ from the origin. We therefore construct another sphere T_0 concentric with S whose radius is $2^i r(1 + \frac{c}{16})$.

Every pair of boxes in the set b_j is clearly bounded by T_0. In particular those pairs which will be bounded together at the next level. We now apply Lemma 5 to those pairs and realize that a sphere T_1 of radius $\sqrt{3}$ times larger than the radius of T_0 and concentric with T_0 will bound the OBBs created at level $i-4$. Continuing this line of argument up to level i we get that a sphere T_5 of radius $2^i r(1 + \frac{c}{16})\sqrt{3}^5$ bounds the OBB that bounds all of the b_j boxes. We need this radius to be smaller than $c\,2^i r$ to complete the proof.

$$\begin{aligned} 2^i r\left(1 + \frac{c}{16}\right)\sqrt{3}^5 &\leq c\,2^i r \\ c\left(\frac{1}{\sqrt{3}^5} - \frac{1}{16}\right) &\geq 1 \qquad (6) \end{aligned}$$

We choose c according to Inequality (6) (which is a valid choice since $\sqrt{3}^5 < 16$). If $c_1 > c$ we make c_1 be the desired constant. $\square$

Lemma 6 assures us that our hierarchy is well behaved, and the OBB's do not become too big. Therefore Lemma 4 applies and the upper bound on the number of possible overlaps of OBBs is $O(N^{\frac{4}{3}})$.

We now turn to showing that this upper bound is tight (for any convex BVH):

LEMMA 7. *Given a chain-aligned BVH of convex BVs over a well-behaved chain, the number of BV overlap tests needed to detect self collision is $\Omega(N^{\frac{4}{3}})$.*

PROOF. We prove the lemma by presenting a chain configuration whose BVH requires this much work. We start by taking d links and placing them along the X axis starting at the origin and proceeding in the positive direction. The center of the dth link is therefore positioned at $x_0 = 2(d-1)r$. We now place d more links parallel to the Y axis, starting at $(x_0, 2r)$ and moving in the positive direction. Finally we place another set of d links parallel to the Z axis starting just above where we left off. This sub-chain of $3d$ links we call a *unit*. We take this unit and make $\frac{d}{8} - 1$ copies of it placing each one translated by $(2r, -2r, 0)$ relative to the previous. We connect the units either at their beginning or end to create a chain. We call these $\frac{d}{8}$ units a *layer*. Figure 6 illustrates one such layer.

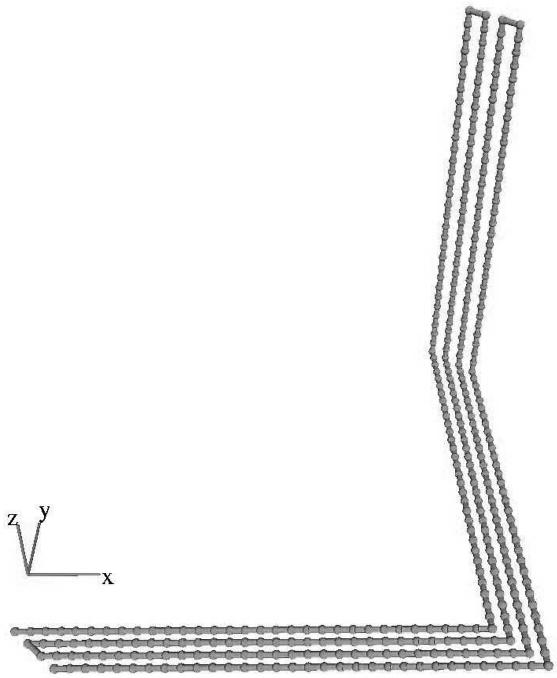


Figure 6: One layer of the chain construction.

We now produce $\frac{d}{8} - 1$ copies of the layer and place them above the first layer, each translated by $(0, -2r, 2r)$ relative to the one below it. We connect the layers at the point where the lower ends. The direction of the chain along an upper layer is exactly the reverse order of the lower. It can be shown that the convex hull of each unit of the construction above contains the point $(2(d-1)r, (d-1)r, \frac{1}{4}(d-1)r)$. This means that all these convex hulls are pairwise intersecting and hence any hierarchy of convex BVs will therefore induce that many intersecting pairs at the level where all the links of each unit are grouped together in one BV. We have $\frac{d^2}{64}$ units each consisting of $3d$ links, so we used $3d^3/64$ links. This makes d be $N^{1/3}$ times a small constant. Since we have $d^2/64$ units, we get $\Omega(N^{2/3})$ convex hulls and therefore $\Omega(N^{4/3})$ intersecting pairs. $\square$

We have shown that at one level of the hierarchy there could be $\Omega(N^{4/3})$ overlapping BVs, and therefore the lower bound of Lemma 7 is achieved. Putting Lemma 3 and 7 together, we have proven Theorem 2.

5. EXPERIMENTAL RESULTS

We tested our algorithm, which will henceforth be called **ChainTree**, in two benchmarks against the following three algorithms, based on standard approaches:

Grid - Explicit maintenance of the chain and collision detection by indexing into a 3D grid using a hash table. Both updating and testing for self-collisions in $O(N)$ time.

1-OBBTree - An OBB hierarchy is created from scratch after each set of changes, and then tested against itself for self-collisions. Updating takes $O(N \log N)$ time and testing is worst case $O(N)$.

K-OBBTree - After each set of changes an OBB hierarchy is created from scratch for each piece of the chain that remained rigid. Each pair of OBB hierarchies is tested for overlap. Updating takes $O(N \log N)$ time and testing is worst case $O(N)$.

The **1-OBBTree** and **K-OBBTree** structures, as well as the OBB overlap testing used by the **ChainTree** are based on the *PQP* library [10, 18] from UNC. The benchmarks were run on a Sun Ultra Enterprise 5500 machine with eight 400 MHz UltraSPARC-II CPUs and 4.0 GB of RAM (no parallelization was used).

For the first benchmark, we created a pseudo-molecular chain of spheres of radius 1, spaced 4 apart. We looked at chains of 1,000, 2,500, 5,000 and 10,000 spheres in a compact cube-like configuration. We ran each structure for 20,000 iterations. Each iteration the chain underwent one torsional change and then tested for self-collision. If a collision is detected the last move is undone before applying the next change. The results showing average time per operation are presented in Figure 7. Note that the testing time for **ChainTree** and **K-OBBTree** may be too small to be discernable in the graph.

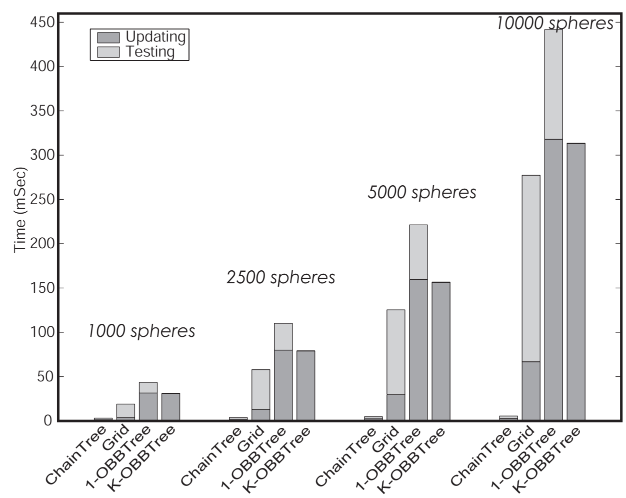


Figure 7: Detecting a collision in a compact chain.

For the second benchmark we used real protein backbones taken from protein structures in the Protein Data Bank (PDB). The proteins were chosen because of their size so we would have a small, a medium and a long backbone. The starting configuration of the proteins was the highly compact folded state. We extracted the positions of the backbone atoms for each of the proteins and ran the same simulation as in the previous benchmark for each of them. The results are in Figure 8.

In Table 1 we see the number of box overlap tests that our algorithm uses in each of the benchmarks. Clearly the average number of box tests performed in practice is much smaller than the $O(N^{\frac{4}{3}})$ worst case bound, although it is not

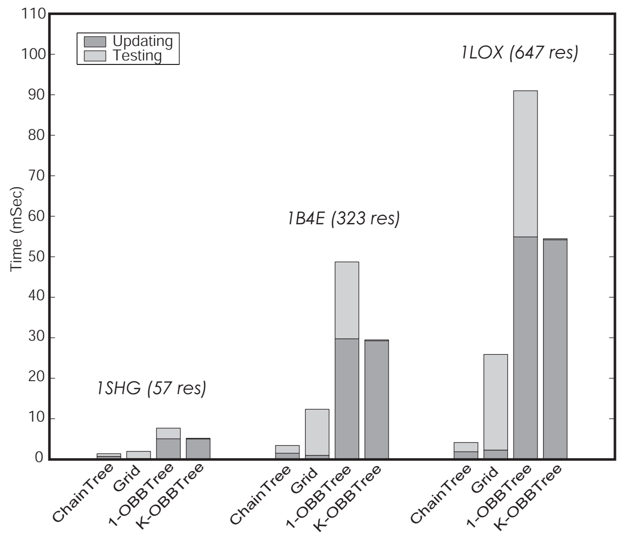


Figure 8: Detecting a collision in protein backbones (each residue contributes 3 atoms to the chain).

as small as the number attained by the adapted hierarchy of the **K-OBBTree** structure.

	1000	2500	5000	10000
ChainTree	703	715	905	964
K-OBBTree	114	131	163	178

(a)

	1SHG	1B4E	1LOX
ChainTree	242	664	747
K-OBBTree	73	113	146

(b)

Table 1: The average number of box overlap tests per self-collision test for (a) the compact pseudo-molecular chain and (b) the protein backbones.

6. EXTENSIONS AND FUTURE WORK

6.1 Tighter Bounding

To achieve the $O(\log N)$ update time our bounding boxes are not tight around the geometry of the chain. Each box is built to bound the two boxes below it in the hierarchy and thus a box will not bound the geometry perfectly. That is, there will be some amount of extra volume in every box. This wasted space accumulates the higher a box is located in the hierarchy and may cause unnecessary work when testing for collisions. Obviously, the extra volume in each box is located along its faces. During the update stage, immediately after a box is recomputed, we would like to *shrink* it, i.e. push in each face until it actually touches a piece of the chain. To accomplish this we came up with a shrinking scheme that can be incorporated into the update procedure, which makes sure all boxes are tight. The basic idea is that we can recursively find extremal features of the underlying geometry by exploiting the hierarchy below each box we are trying to "shrink".

In the worst case this scheme requires examining all bottom level links contained in a box, which requires $\Theta(N)$ work for high level boxes. However, our experience shows that in practice it performs much better, since we use the hierarchy to prune the number of links that actually need to be examined.

6.2 Proteins and Side-Chains

A protein molecule is built by stringing together elementary pieces called *amino acids*. An amino acid has two parts: a *backbone* (a chain of three atoms) and a *side-chain* of up to 20 atoms. The backbone pieces of all the amino acids of a protein are concatenated to form the main chain, each contributing two torsional degrees of freedom. The side-chains then stick out from the backbone, each having between 0 and 4 degrees of freedom. A first approximation to integrate side-chains is to consider each one together with the backbone atom it is connected to as one rigid piece and approximate it using a sphere or a box. This way we are left with a simple kinematic chain and no modifications to the algorithm are required.

At this stage it is important to notice that our algorithm does not require the links of the chain to be simple objects. A link may be a complex entity described by its own hierarchy. Moreover, a link does not have to be rigid, as long as its degrees of freedom do not affect the chain, but only its own shape. We can build a small hierarchy that represents the side-chain and the backbone atom it is connected to, and use the top level box (or sphere) as an elementary piece in the main kinematic chain. The hierarchy for a side-chain could be very crude because it is made up of 20 atoms or less. A change to a degree of freedom of a side-chain will require updating all the boxes that contain it (the same as for a chain joint change) but no shortcut transformations.

6.3 Other Extensions

Our BVH could also be used to find collisions between a chain and other objects. If the other object is simple, namely similar in size and geometric complexity to a link of the chain, we can detect a collision in worst case $\Theta(N^{\frac{2}{3}})$ time. If the object is another chain of equal size we can detect a collision in worst case $\Theta(N^{\frac{4}{3}})$ time. In general, it is possible to test for collisions against any object represented as a hierarchy of the same type of BVs. Another useful extension to our algorithm allows for finding all pairs of links, which are less than some cutoff distance away from each other. This capability is very useful when computing internal energy for macro-molecules. This is accomplished by growing the links by half the cutoff distance and then finding all self-collisions.

The hierarchy could also be used to efficiently compute minimal distance from some obstacle. The obstacle could either be simple or have its own hierarchy. Finding minimal separation is analogous to testing for a collision. As the hierarchies are traversed, when checking two BVs, we compute the minimal distance between them, and proceed down that path only if the distance is smaller than the minimum found so far. This use of BVHs was suggested by Quinlan [22], and later by Larsen et al [18], who also suggest RSS as a better BV for this application. We believe our algorithm would work well with RSS for both self-collision detection and answering minimal distance queries.

7. CONCLUSIONS

We presented a novel representation for kinematic chains, allowing for efficient maintenance of the relative positions of the links, as well as fast detection of self-collision, as the chain undergoes changes at its joints. We update the chain structure of N links in $O(\log N)$ time per change, with an upper bound of $O(N)$ for as many as N simultaneous changes. We build an OBB hierarchy over the chain, and use it to detect self-collisions with a proven worst-case complexity of $O(N^{\frac{4}{3}})$. However we found that in practice we can expect a much better performance in most cases. The benchmarks we performed clearly show the superiority of our method in maintaining chains of 1000 to 10000 links, with better than 30 time speed up over all other algorithms for chains of 10000 links. Our approach has immediate applications in the fields of robotics and molecular biology. It can be extended to dealing with chains having links with complex geometry, as well as short side-chains.

8. ACKNOWLEDGEMENT

The authors would like to thank Pankaj Agarwal for his helpful suggestions.

This work was partially funded by NSF_ITR grant `DUKE UNI/01-SC-NSF-1009-01` and a grant from Stanford University's Bio-X program.

Work by Dan Halperin has been supported in part by the IST Programme of the EU as a Shared-cost RTD (FET Open) Project under Contract No IST-2000-26473 (ECG - Effective Computational Geometry for Curves and Surfaces), by The Israel Science Foundation founded by the Israel Academy of Sciences and Humanities (Center for Geometric Computing and its Applications), and by the Hermann Minkowski – Minerva Center for Geometry at Tel Aviv University.

References

[1] P. Agarwal, J. Basch, L. Guibas, J. Hershberger, and L. Zhang. Deformable free space tilings for kinetic collision detection. In *4th Int. Workshop on Alg. Found. Rob. (WAFR)*, March 2000.

[2] J. Barraquand, L. Kavraki, J.-C. Latombe, T. Li, R. Motwani, and P. Raghavan. A random sampling scheme for path planning. *Int. J. of Rob. Research*, 16(6):759–774, 1997.

[3] J. Brown, S. Sorkin, C. Bruyns, J. Latombe, K. Montgomery, and M. Stephanides. Real-time simulation of deformable objects: Tools and application. In *Comp. Animation*, 2001.

[4] G. Chirikjian and J. Burdick. The kinematics of hyper-redundant robotic locomotion. *IEEE Tr. on Rob. and Auto.*, 11(6):781–793, December 1995.

[5] J. D. Cohen, M. C. Lin, D. Manocha, and M. Ponamgi. I-COLLIDE: An interactive and exact collision detection system for large-scale environments. In *Sym. on Interactive 3D Graphics*, pages 189–196, 218, 1995.

[6] J. J. Craig. *Introduction to Robotics: Mechanics and Control*. Addison-Wesley Publishing Company, Inc, 2nd edition, 1989.

[7] T. E. Creighton. *Proteins : Structures and Molecular Properties*. W. H. Freeman and Company, New York, 2nd edition, 1993.

[8] B. Faverjon. Obstacle avoidance using an octree in the configuration space of a manipulator. In *IEEE Conf. on Rob.*, pages 504–510, 1984.

[9] F. Ganovelli, J. Dingliana, and C. O'Sullivan. Buckettree: Improving collision detection between deformable objects. In *SCCG2000 Spring Conf. on Comp. Graphics*, 2000.

[10] S. Gottschalk, M. C. Lin, and D. Manocha. OBBTree: A hierarchical structure for rapid interference detection. *Comp. Graphics*, 30(Annual Conf. Series):171–180, 1996.

[11] L. J. Guibas, A. Nguyen, D. Russel, and L. Zhang. Deforming necklaces. In *Symp. on Comp. Geometry*, 2002.

[12] D. Halperin, J.-C. Latombe, and R. Motwani. Dynamic maintenance of kinematic structures. In J.-P. Laumond and M. Overmars, editors, *Alg. for Rob. Motion and Manipulation (WAFR '96)*, pages 155–170. A.K. Peters, Wellesley, 1997.

[13] D. Halperin and M. H. Overmars. Spheres, molecules and hidden surface removal. *Comp. Geom.: Theory and App.*, 11(2):83–102, 1998.

[14] P. M. Hubbard. Approximating polyhedra with spheres for time-critical collision detection. *ACM Tr. on Graphics*, 15(3):179–210, 1996.

[15] S. Kambhampati and L. S. Davis. Multiresolution path planning for mobile robots. *IEEE J. of Rob. and Auto.*, RA–2(3):135–145, 1986.

[16] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Tr. on Rob. and Auto.*, 12(4):566–580, 1996.

[17] J. T. Klosowski, J. S. B. Mitchell, H. Sowizral, and K. Zikan. Efficient Collision Detection Using Bounding Volume Hierarchies of k-DOPs. *IEEE Tr. on Visualization and Comp. Graphics*, 4(1):21–36, 1998.

[18] E. Larsen, S. Gottschalk, M. C. Lin, and D. Manocha. Fast distance queries with rectangular swept sphere volumes. In *IEEE Conf. on Rob. and Auto.*, 2000.

[19] M. C. Lin and J. F. Canny. A fast algorithm for incremental distance calculation. In *IEEE Int. Conf. on Rob. and Auto.*, pages 1008–1014, 1991.

[20] S. Ma, S. Hirose, and H. Yoshinada. Development of a hyper-redundant multijoint manipulator for maintenance of nuclear reactors. *Int. J. of Advanced Rob.*, 9(3):281–300, 1995.

[21] K. W. Plaxco, K. T. Simon, and D. Baker. Contact order, transition state placement and the refolding rates of single domain proteins. *J. of Molecular Biology*, 277(4):985–994, April 1998.

[22] S. Quinlan. Efficient distance computation between non-convex objects. In *IEEE Intern. Conf. on Rob. and Auto.*, pages 3324–3329, 1994.

[23] G. van den Bergen. Efficient collision detection of complex deformable models using AABB trees. *J. of Graphics Tools*, 2(4):1–13, 1997.

[24] M. Yim. *Locomotion with a Unit-Modular Reconfigurable Robot*. PhD thesis, Department of Computer Science, Stanford University, 1994.