Collision Detection for Deforming Necklaces *

Pankaj K. Agarwal† Leonidas Guibas‡ An Nguyen§ Daniel Russel¶
Li Zhang‖

November 6, 2003

Abstract

In this paper, we propose to study deformable necklaces — flexible chains of balls, called beads, in which only adjacent balls may intersect. Such objects can be used to model macromolecules, muscles, ropes, and other linear objects in the physical world. We exploit this linearity to develop geometric structures associated with necklaces that are useful for collision detection in physical simulations. We show how these structures can be implemented efficiently and maintained under necklace deformation. In particular, we study a bounding volume hierarchy based on spheres which can be used for collision and self-collision detection of deforming and moving necklaces. As our theoretical and experimental results show, such a hierarchy is easy to compute and, more importantly, is also easy to maintain when the necklace deforms. Using this hierarchy, we achieve a collision detection upper bound of $O(n \log n)$ in two dimensions and $O(n^{2-2/d})$ in d-dimensions, $d \geq 3$. To our knowledge, this is the first subquadratic bound proved for a collision detection algorithm using predefined hierarchies. In addition, we show that the power diagram, with the help of some additional mechanisms, can be used to detect self-collisions of a necklace in a way that is complementary to the sphere hierarchy.

*A preliminary version of this paper appeared in the Proc. of the 18th ACM Symp. on Computational Geometry, 2002. The work by P.A. was supported by NSF under grants CCR-00-86013, EIA-98-70724, EIA-99-72879, EIA-01-31905, and CCR-02-04118, and by a grant from the U.S.-Israeli Binational Science Foundation. The work by L.G., A.N., and D.R. was supported by NSF grants CCR-9910633, CCR-0204486,ITR-0086013, ARO grant DAAD19-03-1-0331, and a Stanford Graduate Fellowship.

†Department of Computer Science, Box 90129, Duke University, Durham NC 27708-0129. pankaj@cs.duke.edu

‡Department of Computer Science, Stanford University, Stanford, CA 94305. guibas@cs.stanford.edu

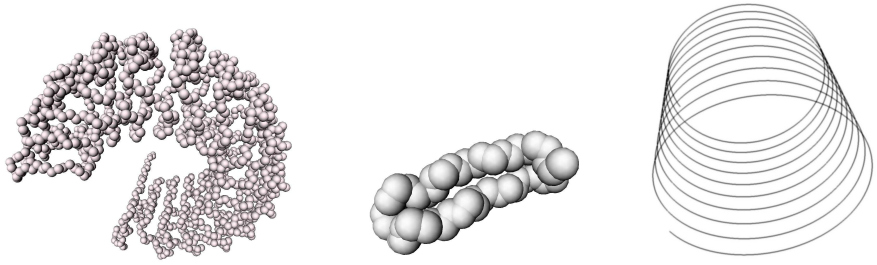
§Department of Computer Science, Stanford University, Stanford, CA 94305. anguyen@cs.stanford.edu

¶Department of Computer Science, Stanford University, Stanford, CA 94305. drussel@cs.stanford.edu

‖HP Labs, 1501 Page Mill Road, Palo Alto, CA 94304. l.zhang@hp.com

1 Introduction

In many applications objects are hierarchically organized into groups and the motions of objects within the same group are highly correlated. For example, though not all points in an elastic bouncing ball, elongating muscle, or folding rope follow exactly the same rigid motion, the trajectories of nearby points are similar and the overall motion is perhaps best described as the composition of a global rigid motion with a small local deformation. Similarly, the motion of an articulated figure, e.g., a man walking or a protein deforming, is most succinctly described as a set of relative motions of limbs or parts against other parts. Motivated by such applications, we study a simple model for deformable *linear* objects such as protein backbones, muscles, and ropes. We represent such an object as a sequence of spheres. We call the linear object a *necklace*, and its spherical elements *beads*. Spheres are widely used as primitive elements in engineering modeling [6], and they are obviously the appropriate choice for proteins. (See Figure 1 for a few examples of necklaces.) Spheres also simplify substantially the basic geometric calculations and allow us to focus on the combinatorial issues that form our main interest. In this paper we study how to track different geometric attributes of a necklace, such as its power diagram or a bounding sphere hierarchy, which are useful in detecting collision between two necklaces or self-collision within a single necklace. Though a necklace lives in $\mathbb{R}^3$, it has an essential one-dimensional character, which allows us to develop simpler algorithms.[1]



(i) (ii) (iii)

Figure 1. A few necklaces: (i) Protein 1a4yA0, (ii) a fragment of a protein, and (iii) a helix.

The exact way in which a necklace moves and deforms depends on the physical model used and is application dependent. Since we do not model the physics, we take a *black box* view of the physical simulation. We assume that at certain times (the time steps of the simulation) an oracle moves the beads forming the necklace according to the underlying physics and reports their new positions back to us. Though in general every single bead moves at each step, we assume that the time steps chosen by the simulator are such that the motion of each bead at every step is small, when compared to the overall scale of the simulation. Thus the basic problem we address is how to repair a geometric structure after small displacements of its defining elements.

[1] It is worth noting that, though modeling some aspects of linear objects is simpler than modeling surfaces or solids, linear objects can come into self-proximity and self-contact in more elaborate ways than their higher-dimensional counterparts. So from a certain point of view, dealing with collisions for deformable linear objects is the hardest case.

Related work. A commonly used approach to expedite the collision detection between complex shapes is based on hierarchies of simple bounding volumes surrounding each of the objects. To build such a hierarchy, a specific geometric shape is selected as the bounding volume of choice. Common choices are axis-aligned bounding boxes (AABBs) [19, 2], arbitrarily oriented bounding boxes (OBBs) [14], k-DOPs [23], and spheres [29, 19]; see [24] for a survey. For a given placement of two non-intersecting objects, their respective hierarchies are refined only to the coarsest level at which the primitive shapes in the two hierarchies can be shown to be pairwise disjoint. The choice of a bounding shape usually presents a trade-off between the ease of testing for intersection two such shapes, and the total number of bounding volume checks required for detecting collision. Recent work by Zhou and Suri [34] provides a theoretical framework that suggests why bounding-volume hierarchies work so well for collision detection in practice. Following the publication of the preliminary version of this paper, Erickson [11] showed that hierarchic collision detection is $O(n \log n)$ for a broad class of practical geometric model and bounding volumes.

Motion in the physical world is in general continuous over time. Since the exact motion is hard to predict, most systems sample the motion at discrete time steps and repeatedly test for collisions. Instead of performing a full collision check *ab initio* at each time step, in many cases an attempt is made to expedite collision checking by exploiting temporal coherence (see e.g. [27]). In the context of bounding-volume hierarchies, the hierarchy is locally refined or coarsened at each time step, as objects move closer or further apart. Though fixed time-sampling is customary for motion integration, collisions tend to be rather irregularly spaced over time, which makes the choice of time step hard—a large time step will miss some of the collisions, and a short time step will generate unnecessary computation. Basch *et al.* [12] and Erickson *et al.* [12] presented kinetic data structures for detecting collision between two rigidly moving polygons using the kinetic data structure (KDS) framework, which was originally proposed by Basch *et al.* [3] (see [15] for a survey of results on kinetic data structures). Their algorithms avoid many of the problems that arise in the fixed-step time-sampling method by focusing at discrete events when the the structure must be updated. Roughly speaking, these methods maintain a hierarchical representation of each polygon and derive from that a set of geometric conditions on when the hierarchy should be refined/coarsened. Unfortunately, the bounding-volume-hierarchy based methods are not directly amenable to detecting collision between multiple moving objects. Agarwal *et al.* [1] and Kirkpatrick *et al.* [21, 22] proposed global approaches for detecting collision between many moving polygons in the plane, by maintaining a tiling of the common exterior of the polygons into flexible cells, so that the polygons are known to be disjoint until one of the cells self-collides. It is not clear, however, how to extend these techniques to 3-space.

Most of the work to date on bounding-volume hierarchies has focused on collision detection between rigid objects. Very little is known about maintaining such hierarchies for deformable objects. Motivated by applications in indexing spatio-temporal databases, there has been some recent work on maintaining a bounding-volume hierarching of a set of independently moving points. For example, R-trees, which are basically axis-aligned bounding box hierarchies, have been proposed for moving points [31, 28]. The bounding box at each node of the hierarchy changes as the points move. Since these hierarchies aggregate bounding volumes based on spatial proximity, they are

expensive to maintain as the objects undergo large deformations, even if one does not maintain the minimum bounding box at each node.

Our results. In this paper we propose a different approach for maintaining a bounding-volume hierarchy of a necklace, which is easy to maintain and leads to fast collision-detection algorithms. First of all, we define in Section 2 a hierarchy that relies only on *topological proximity* in the object (as opposed to physical proximity in space), since this notion of proximity is better preserved under deformation. For our linear necklaces this gives us an especially simple rule for aggregation: we build a balanced binary tree on the sequence of beads, with the intermediate aggregates corresponding to sets of leaves that are descendants of internal nodes in the tree. Each node is associated with a sphere containing all the beads that are stored at the leaves of subtree rooted at that node. We present two different ways of forming the hierarchy. The first, called the the *wrapped hierarchy*, stores at each node the smallest sphere containing all beads in the subtree rooted at the node. The second, called the *layered hierarchy*, stores at each internal node the smallest sphere containing the two spheres stored at the children of that node. We compare the two hierarchies and discuss pros and cons of each of them. Surprisingly, it turns out that, in any dimension, a bounding sphere in the layered hierarchy is at most a factor of $\sqrt{\log n}$ bigger than the corresponding one in the wrapped hierarchy, and this bound is tight in the worst case. We present efficient algorithms for constructing and maintaining the wrapped hierarchy as the necklace deforms, exploiting the relative stability of a combinatorial description of this hierarchy. In other words, we maintain a description of the hierarchy in an implicit combinatorial form, instead of an explicit geometric form. But unlike KDS based methods, we can update the wrapped hierarchy after small motions of the defining beads, without a need for explicit motion plans. Thus our approach is better suited for incorporation into a physics-based motion integrator in which only sampled states of the system are generated.

Next, in Section 3, we analyze the well-known methods for detecting collision and self-collision using sphere hierarchies. While these methods work well in practice, no nontrivial, subquadratic, bound is known on their running time. The quadratic bound arises in the case in which both hierarchies are traversed completely and all leaves of one have to be checked for intersection against all leaves of the other. We show that a slight variant of the folklore self-collision checking method, using the wrapped sphere hierarchy and local refinement as necessary, achieves subquadratic time bounds: $O(n \log n)$ in two dimensions, and $O(n^{2-2/d})$ in d-dimensions for $d \geq 3$ — to our knowledge, this is the first subquadratic worst-case bound for collision-detection algorithms using bounding volume hierarchies[2]. Collision-detection based on bounding-volume hierarchies can still be expensive, however. Time $\Theta(n^{2-2/d})$ is required, when the necklace is tightly packed. We therefore propose another method, based on power diagrams (see [7] and Section 3.2 for the definition), which is especially fast in this case, since the size of the power diagram is linear in all dimensions for packed configurations [10]. Our method basically keeps track of the shortest edge of the power diagram. While it was known that the closest pair of a set of disjoint balls defines an edge in the

[2]A similar bound was reported concurrently with our work for oriented bounding boxes in [25]. They studied kinematic chains analogous to our necklaces, although with a different motivation. They were interested in Monte-Carlo simulations of proteins in which a single torsional bond rotates at each time step, possibly by a large angle

power diagram [17], that result does not apply directly to our problem since we allow adjacent beads of a necklace to overlap.

Finally, we present and discuss experimental results, which validate our claims and prove the effectiveness of our methods. Our simulations show that the wrapped hierarchy is much more stable under motion than the power diagram, one of the alternatives for collision detection. For typical data the tightness of fit of the layered and wrapped hierarchies are fairly close. Furthermore, when the layered and wrapped hierarchies are used in their entirety, the greater simplicity of the bounding volume calculations for the layered hierarchy makes it faster. However, we can exploit the combinatorial structure of the wrapped hierarchy to perform much faster collision detection tests between disjoint objects.

2 Necklaces and Bounding-Sphere Hierarchy

2.1 Necklaces

A *necklace* consists of a sequence of n closed balls $\mathcal{B} = \{B_1, B_2, \ldots, B_n\}$, called *beads*, in the Euclidean space $\mathbb{R}^d$. We assume that only adjacent balls along the necklace may intersect and no ball is fully contained in another set of balls. We also assume the beads satisfy the following two properties:

uniformity: there is a constant $\rho \geq 1$ such that the ratio of the radii of any two balls in a necklace is in the interval $[1/\rho, \rho]$; and

connectivity: the beads form a connected set — in other words, any two consecutive beads along the necklace have a point in common.

We refer to the polygonal path connecting the centers of the beads (in order) as the *backbone* of the necklace. Whatever conditions we adopt, we assume that they are maintained by the underlying physics causing a necklace to move or deform. We remark that similar "necklace conditions" were studied for establishing the optimality of tours in the plane [8]. As mentioned in the introduction, these conditions capture the properties of a large family of shapes such as proteins and ropes. The following lemma is a simple yet useful property implied by the necklace conditions.

Lemma 2.1 *Suppose that the minimum radius of the beads is* 1. *Then any ball with radius* R *contains* $O(\rho^d R^d)$ *beads, and any* m *consecutive beads are contained in a ball with radius* $O(\rho m)$.

2.2 Bounding sphere hierarchy

Given a sequence of beads, we can construct a bounding volume hierarchy by iteratively grouping the beads into larger and larger sets. The grouping can be represented as a (binary) hierarchy tree

where each leaf corresponds to a bead, and each internal node corresponds to a subset of the beads underneath the node. For each internal node, we compute a sphere, which we call a *cage*, that encloses all the beads represented by the leaves underneath the node.

Typically, the grouping should be done in a way such that the cages have small size. Although this approach is good for rigid objects [30], a sophisticated grouping may incur high cost when the necklace deforms because it has to be re-computed whenever the beads move. Because of the uniformity and connectivity of the necklaces, we group the beads by simply constructing a balanced binary tree on top of the beads according to their order in the necklace. Although this may not be the best way to construct the hierarchy as two beads far away in the sequence may be spatially close, the benefit is obvious: we never need to change the topology of the tree when the necklace is deforming, and such grouping results in reasonably compact bounding cages by the necklace properties.

For each node v in the hierarchy tree, let $\mathcal{B}_v \subseteq \mathcal{B}$ be the set of beads stored at the leaves of the subtree rooted at v. $\mathcal{B}_v$ is a contiguous subchain of the original necklace. A cage $\mathcal{C}_v$, stored at v, is a sphere that contains all the beads in $\mathcal{B}_v$. This is one instance where we heavily use the *a priori* known structure of the type of object we are modeling. We define the *level* of a node in the tree to be the maximum distance to a leaf node under it. By definition, the leaves are at level 0. Define the height of a tree to be the level of its root.

Given the hierarchy tree, we still need to decide how the cages are computed. We consider two different methods for computing the cages. In one method, the cage is defined as the *minimum enclosing sphere* (MES) of the beads underneath the corresponding node. The resulted hierarchy, denoted as $\mathcal{W} = \mathcal{W}(\mathcal{B})$ is called the *wrapped hierarchy*. The cages of the children of a node in $\mathcal{W}(\mathcal{B})$ can stick out of the cage of its parent; see Figure 2 (i). In the other method, where the resulted hierarchy is called *layered hierarchy* [29] and denoted as $\mathcal{L} = \mathcal{L}(\mathcal{B})$, each cage is computed as the MES of the *cages of its two children*; see Figure 2 (ii). Though the wrapped hierarchy is slightly more difficult to compute than the layered hierarchy, it is tighter fitting and most importantly it can be maintained more easily under deformation — a fact that at first seems counter-intuitive. We will therefore mostly focus on the wrapped hierarchy.

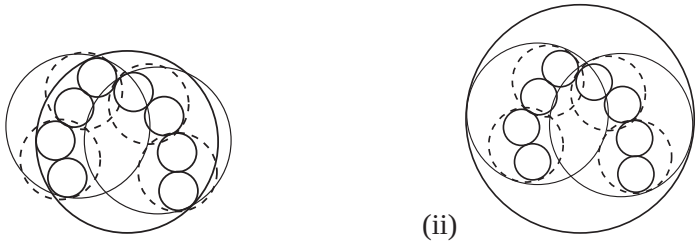


Figure 2. Wrapped (left) and layered (right) sphere hierarchies. The base beads are black. Notice that each cage in the wrapped hierarchy is supported by 2 or 3 beads.

For a set S of spheres in $\mathbb{R}^d$, let $\mathcal{M}(S)$ be the smallest sphere that contains all the spheres in S. The *basis* of S, denoted as $\mathbb{B}(S)$, is the smallest subset $A \subseteq S$ so that $\mathcal{M}(A) = \mathcal{M}(S)$. It is well

known that $|A| \leq d+1$ in $\mathbb{R}^d$. The following well-known property of minimum enclosing spheres will be crucial for our algorithm.

Lemma 2.2 *Let S be a set of spheres, and let $A \subseteq S$. If all spheres in S are contained in $\mathcal{M}(A)$, then $\mathbb{B}(A) = \mathbb{B}(S)$.*

For the wrapped hierarchy, $\mathcal{C}_v = \mathcal{M}(\mathcal{B}_v)$. Let $\mathbb{B}_v = \mathbb{B}(\mathcal{C}_v)$. The key property of the wrapped hierarchy that is of interest to us is that $\mathcal{C}_v$ is fully determined by $\mathbb{B}_v$, a set of at most four spheres from $\mathcal{B}_v$ for $d = 3$.

2.3 Construction and maintenance of the hierarchy

The wrapped hierarchy $\mathcal{W}(\mathcal{B})$ can be constructed by computing $\mathcal{M}(\mathcal{B}_v)$ at each node of the hierarchy. There is a complex linear-time deterministic algorithm for computing the minimum enclosing sphere of a set of congruent spheres [26], and there is a simple randomized algorithm with linear expected time [32]. If the beads have different radii, the randomized algorithm is slightly more complicated but with the same overall running time [20]. Therefore, it takes $O(n \log n)$ time to construct $\mathcal{W}(\mathcal{B})$ of a necklace $\mathcal{B}$ with n beads.

Next we describe how we maintain the wrapped hierarchy as the necklace deforms. As the necklace deforms, $\mathcal{C}_v$ at all nodes v of the hierarchy changes continuously, but $\mathbb{B}_v$ remains the same for a period. At certain discrete *events* $\mathbb{B}_v$ changes typically by a pivoting step in which

(i) an old basis bead leaves the basis, and

(ii) a new bead from the enclosed subnecklace enters the basis.

At times only one of these events happens (i.e., the size of the basis reduces by one or increases by one), but the total number of basis beads will always remain at most four for $d = 3$. This combinatorialization of a continuous phenomenon is an insight analogous to what is exploited in kinetic data structures.

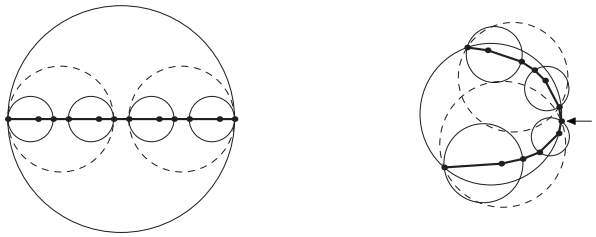


Figure 3. A combinatorially defined sphere hierarchy is stable under deformation. Only the top level cage differs between the two conformations.

We expect that under smooth deformation the combinatorial description of the cages (i.e. their basis beads) will stay unchanged for a fairly long time, and when finally the basis of a cage needs

to change, that change will be easy to detect and the *basis update* simple to perform. For instance, in Figure 3 we show a 2-D example of the upper layers of such a hierarchy in two quite different configurations of a deforming necklace. It so happens that all the hierarchy cages except for the root cage continue to have the same combinatorial description at all intermediate configurations.

Recall that our goal is to update the wrapped hierarchy at each time step. Let $\mathcal{B}(t)$ denote the configuration of the necklace at time t. Similarly define $\mathcal{B}_v(t)$, $\mathcal{C}_v(t)$, $\mathbb{B}_v(t)$, and $\mathcal{W}(t) = \mathcal{W}(\mathcal{B}(t))$. At time step t, we need to verify the correctness of the hierarchy, i.e., determine whether $\mathbb{B}_v(t-1)$ is still the basis of $\mathcal{C}_v(t)$, for all nodes v in the hierarchy, and update those which are no longer correct.

The verification is done in a hierarchical manner, bottom up with a top down pass from each tree node; we call this method the *cascade verification*. Suppose that we have checked the validity of the descendants of a node u. We first compute the minimum enclosing sphere $\sigma = \mathcal{M}(\mathbb{B}_u(t-1))$. By Lemma 2.2, it is sufficient to check that all the beads in $\mathcal{B}_u$ are contained in σ. This can be either done directly in linear time, which we call *naive verification*, or indirectly as follows. Maintain a frontier, F, initially containing the children of u. At each step we take a node v out of F and determine whether $\mathcal{C}_v(t) \subseteq \sigma$. If the answer is yes, we move to the next node in F. If $\mathcal{C}_v(t) \not\subseteq \sigma$ and v is an internal node, then we add its children to F. Finally, if $\mathcal{C}_v(t) \not\subseteq \sigma$ and v is a leaf, then we conclude that the bead B_v has *escaped* from σ and $\mathbb{B}_u(t-1)$ is no longer valid and needs to be updated. If we can continue the above process until F becomes empty without encountering an escaped leaf node, we know that $\mathbb{B}_u(t) = \mathbb{B}_u(t-1)$.

If beads escape from an enclosing cage $\mathcal{C}_u$, a basis update must be performed. At least one of the escaped beads must be a basis bead for the new cage $\mathcal{C}_u$. The LP-type algorithm [32] allows easy exploitation of this knowledge in a natural manner, as well as easy use of other heuristic information about which beads are likely to be basis beads. The cost of the update is expected to be linear in the number of beads enclosed by the cage.

Remark. It is tempting to try to accelerate the above process by noting that the geometry belonging to a cage must be contained in the intersection of the cages on the path from that node to the root, and checking to see whether this volume is contained in the cage being verified. However, in practice the extra complexity of this check more than outweighs its benefits. While in the worst case, the above procedure may take $\Theta(n \log n)$ time if all paths need to be traversed, our experiments suggest than in most instances the actual time is closer to linear, as only paths to the basis beads need to be checked.

2.4 Tightness of layered hierarchy

While the wrapped hierarchy is always tighter than the layered hierarchy, it is interesting to know exactly how much difference there can be between the two. The radius of a cage $\mathcal{C}_v$ in the wrapped hierarchy is only determined by the set $\mathcal{B}_v$ On the other hand, the radius of $\mathcal{C}_v$ in the layered hierarchy depends on how the beads in $\mathcal{B}_v$ are ordered. In the following, we show that no matter how we order the beads, the radius of the cage at the root of the layered hierarchy is at most $\sqrt{h+1}$

times the that of the wrapped hierarchy, where h is the height of the hierarchy tree and is $\lceil \log n \rceil$ for the balanced hierarchy tree we use in this paper.[3] This bound is almost tight in the sense that there exist a sequence $\mathcal{B}$ of spheres (points actually) such that the radius of cage at the root of $\mathcal{L}(\mathcal{B})$ is $\sqrt{h}$ larger than the radius of $\mathcal{M}(\mathcal{B})$.

Theorem 2.3 *Let $\mathcal{B}$ be a necklace with n beads in $\mathbb{R}^d$, and let T be a hierarchy tree of $\mathcal{B}$ whose height is h. If we denote by τ_W, τ_L the radii of the root spheres for the wrapped and layered hierarchies of the point set, respectively, then*

$$\tau_L \leq \tau_W \sqrt{h+1}.$$

This bound is almost tight in the worst case.

The following lemma proves the upper bound.

Lemma 2.4 *Let $\mathcal{B}$ be a necklace, let O and R be the center and the radius of $\mathcal{M}(\mathcal{B})$, and let O_ℓ and R_ℓ be the center and the radius of a cage on level ℓ in the layered hierarchy of $\mathcal{B}$. Let $d_\ell = |OO_\ell|$. Then*

$$R_\ell^2 \leq (\ell+1)(R^2 - d_\ell^2).$$

Proof. Without loss of generality, let us assume that $R = 1$. We prove the lemma by induction on ℓ. We should emphasize that O and R are fixed throughout the induction.

For $\ell = 0$, a 0-th level cage B is a bead in $\mathcal{B}$. Since $B \subseteq M(\mathcal{B})$, we have that $R_0 = R(B) \leq 1 - d_0$ where $d_0 \leq 1$. Therefore

$$R_0^2 \leq (1-d_0)^2 \leq 1 - d_0^2.$$

Now, we assume that the lemma holds for all spheres at level $\ell - 1$ in the layered hierarchy and show that the lemma still holds for spheres at level ℓ.

Let C and r be the center and the radius of a cage $\mathcal{C}$ stored at a node v whose level is ℓ in the layered hierarchy, and let C_1, C_2 and r_1, r_2 the centers and radii of the cages $\mathcal{C}_1, \mathcal{C}_2$ stored at the two children of v. Let $d = |OC|$, $d_1 = |OC_1|$, and $d_2 = |OC_2|$, see Figure 4. By induction hypothesis, $r_1^2 \leq \ell(1 - d_1^2)$ and $r_2^2 \leq \ell(1 - d_2^2)$. We will show that $r^2 \leq (\ell+1)(1-d^2)$. This is clearly true when $\mathcal{C}$ is identical to one of $\mathcal{C}_1$ and $\mathcal{C}_2$, so assume that C is bigger than both $\mathcal{C}_1$ and $\mathcal{C}_2$.

Let $a_1 = |CC_1|$, $a_2 = |CC_2|$, $s = (r_1 + r_2)/2$, and $a = (a_1 + a_2)/2$, as in Figure 4. Then

$$r = r_1 + a_1 = r_2 + a_2 = s + a.$$

Let $\Delta = (a_2 - a_1)/2 = (r_1 - r_2)/2$. Since $\cos(\angle OCC_1) = -\cos(\angle OCC_2)$, using the law of cosines, we obtain:

$$\frac{a_1^2 + d^2 - d_1^2}{2a_1 d} = -\frac{a_2^2 + d^2 - d_2^2}{2a_2 d}.$$

[3] All logarithms are taken base 2 in this paper.

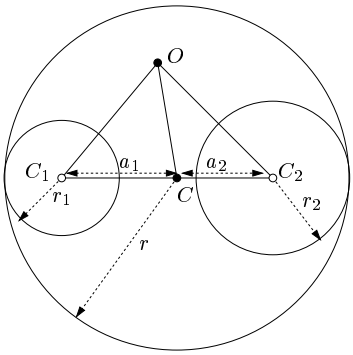


Figure 4. A sphere in the layered hierarchy and its two children. Lemma 2.4 proves that the farther the centers of the children are from the center of their parent, the smaller their radii must be in comparison.

That is,

$$d^2 \;=\; \frac{a_2 d_1^2 + a_1 d_2^2}{a_1 + a_2} - a_1 a_2 \,. \tag{1}$$

Using Equation (1) we have

$$\begin{aligned} d^2 &\leq \frac{(a+\Delta)(1-r_1^2/\ell) + (a-\Delta)(1-r_2^2/\ell)}{2a} - (a-\Delta)(a+\Delta) \\ &= \frac{2a - a(r_1^2+r_2^2)/\ell - \Delta(r_1^2-r_2^2)/\ell}{2a} - (a^2-\Delta^2) \\ &= 1 - \frac{1}{2\ell}(r_1^2+r_2^2) - \frac{\Delta}{2a\ell}(r_1^2-r_2^2) - (a^2-\Delta^2) \\ &= 1 - \frac{s^2+\Delta^2}{\ell} - \frac{2s\Delta^2}{a\ell} - (a^2-\Delta^2)\,. \end{aligned}$$

Thus,

$$(\ell+1)(1-d^2) - r^2 \;\geq\; (\ell+1)\left[\frac{s^2+\Delta^2}{\ell} + \frac{2s\Delta^2}{a\ell} + (a^2-\Delta^2)\right] - (s+a)^2\,.$$

Simplifying the right hand side of the above inequality, we get:

$$(\ell+1)(1-d^2) - r^2 \geq \frac{(\ell a - s)^2 a_1 a_2 + (s+a)^2\Delta^2}{\ell a^2} \geq 0\,.$$

Thus, $r^2 \leq (\ell+1)(1-d^2)$. This completes the inductive proof. □

The above lemma immediately implies the upper bound in Theorem 2.3 since

$$\tau_L \leq (h+1)(R^2-d^2) \leq (h+1)R^2.$$

The proof of Lemma 2.4 extends to higher dimensions as Equation (1) holds in any dimensions. In what follows, we show that the inequality in Lemma 2.4 can be made almost tight, and we can construct a set of points to attain the upper bound.

Lemma 2.5 *For any $h > 0$, there is a set of 2^h points in the plane such that $\tau_L \geq \tau_W \sqrt{h}$, where τ_W, τ_L denote the radius of the root sphere in the wrapped and layered hierarchy, respectively.*

Proof. We construct a collection of points in the plane such that their wrapped hierarchy has radius 1 and their layered hierarchy has radius $\sqrt{h}$. The construction is done incrementally. We first fix any point O and place a point at O_0 such that $|OO_0| = 1$. Let $S_0 = \{O_0\}$.

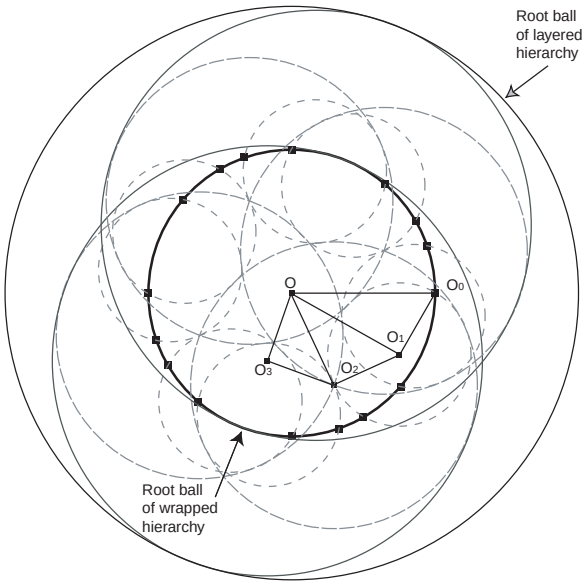


Figure 5. The construction of a set of 16 points on a circle of radius 1 such that the root circle of their layered hierarchy has radius $\sqrt{\log 16} = 2$. The point O_0 is chosen arbitrarily. Right triangles OO_0O_1, OO_1O_2, OO_2O_3 are then constructed such that $|O_0O_1| = |O_1O_2| = |O_2O_3| = 1/2$. The point set is constructed as the closure of the singleton set $\{O_0\}$ with respect to the reflections over the lines OO_1, OO_2, OO_3, and the reflection over the point O. The other circles in the layered hierarchy are also shown.

Suppose that we have constructed the set S_ℓ the first 2^ℓ points for $\ell \leq h-2$. Let O_ℓ be the center of the sphere covering S_ℓ in the layered hierarchy. To construct the set $S_{\ell+1}$, we first find the point $O_{\ell+1}$ such that (1) $\angle OO_{\ell+1}O_\ell = 90°$, and (2) $|O_\ell O_{\ell+1}| = 1/\sqrt{h}$. We can then construct S'_ℓ by flipping all the points in S_ℓ about the line $OO_{\ell+1}$. We then set $S_{\ell+1} = S_\ell \cup S'_\ell$. Finally, we flip S_{h-1} about the center O to obtain S'_{h-1} and set $S_h = S_{h-1} \cup S'_{h-1}$. See Figure 5.

First, we show that the above construction is valid as O_ℓ's are well defined. Since $|OO_{\ell+1}| = \sqrt{|OO_\ell|^2 - 1/h}$, it is easy to derive, by induction, that $|OO_\ell| = \sqrt{1 - \ell/h}$. Therefore, as long as $\ell \leq h-2$, we have that $|OO_\ell| > 1/\sqrt{h}$ and thus $O_{\ell+1}$ always exists.

In the above construction, since each flipping is either about a line passing through the center O or about O itself, the distance from the points to O remain the same, i.e. all the points are on the unit circle centered at O. Thus, the radius of $\mathcal{M}(S_h)$ is 1. Now, we show that in the above construction, the sphere B_ℓ covering S_ℓ in the layer hierarchy has radius $\ell/\sqrt{h}$. This is done by induction on ℓ. Clearly, it is true when $\ell = 0$. Suppose that it is true for ℓ. According to the construction, $O_{\ell+1}$ is the center of $B_{\ell+1}$ because the line segment $O_\ell O'_\ell$ is perpendicular to the line $OO_{\ell+1}$ at the point $O_{\ell+1}$. Thus,

$$R_{\ell+1} = R_\ell + \frac{|O_\ell O'_\ell|}{2} = \frac{\ell}{\sqrt{h}} + |O_\ell O_{\ell+1}| = \frac{\ell+1}{\sqrt{h}}.$$

The radius of the root sphere in the layered hierarchy of S_h is therefore $h/\sqrt{h} = \sqrt{h}$, while the radius of $\mathcal{M}(S_h)$ is 1.

Theorem 2.3 is the combination of Lemmas 2.4 and 2.5. □

Remark. We should remark that in Theorem 2.3, the upper bound applies to any grouping by which we construct the layered hierarchy while the lower bound construction is only valid for that *particular* hierarchy tree. If we group the points constructed in Lemma 2.5 differently, e.g. by grouping antipodal points at the first level, then we may obtain a layered hierarchy in which all the spheres have radius 1.

3 Collision Detection

Let $\mathcal{B} = \langle B_1, \ldots, B_n \rangle$ be a necklace with n beads. We describe algorithms for determining whether any two nonadjacent beads in $\mathcal{B}$ intersect. As we will see below, we can easily modify these algorithms to detect collision between two necklaces. We first describe an algorithm based on the wrapped hierarchy of $\mathcal{B}$, which works well when the necklace is not tightly packed. Next, we describe an algorithm based on the power diagram of $\mathcal{B}$, which is efficient for tightly packed necklaces.

3.1 Collision detection with the wrapped hierarchy

The following algorithm, shown in Figure 6, is the standard framework for collision detection using bounding volume hierarchies, adapted to necklaces. We use T to denote the wrapped hierarchy of $\mathcal{B}$, with ρ being its root. Let $l(u)$ (resp. $r(u)$) denote the left (resp. right) child of a node u. The algorithm traverses T in a top-down manner and maintains a queue Q of node pairs (u, v) so that $\mathcal{C}_u$ and $\mathcal{C}_v$ intersect. The algorithm either finds a pair of intersecting cages or deduces that there are no intersecting pair of nonadjacent beads in $\mathcal{B}$.

The correctness of the above algorithm is obvious. The running time largely depends on how the SPLIT procedure is implemented. There are numerous heuristics for deciding which node to split.

Algorithm: BVH-COLLISION-DETECT (T)

```
Q = {(ρ, ρ)}
while Q ≠ ∅ do
  (u, v) = DELETE-FRONT (Q)
  if C_u ∩ C_v ≠ ∅ then
    if leaf(u) ∧ leaf(v) then
      if u ∉ {v, adj(v)} then
        return (COLLISION)
        fi
      else if u = SPLIT (u, v) then
        INSERT ((l(u), v), Q), INSERT ((r(u), v), Q)
        elseINSERT((u, l(v)), Q), INSERT ((u, r(v)), Q)
        fi
      fi
    fi
  end-while
```

Figure 6. The collision-detection algorithm for two necklaces using their sphere hierarchy. SPLIT (u, v) procedure determines whether the algorithm should recursively explore the children of u or of v.

But no subquadratic worst-case bounds were obtained for general objects. We show that we can use a simple heuristic so that the above algorithm runs in subquadratic time for necklacs, in particular, in $O(n \log n)$ time in $\mathbb{R}^2$ and in $O(n^{4/3})$ time in $\mathbb{R}^3$.

We now consider the above collision-detection algorithm from a different perspective, by viewing it as trying to derive a non-intersection proof by finding a sufficient set of *separating bounding volume pairs* while walking down the hierarchy. The algorithm reports a collision if it fails to produce such a set of pairs. Although the framework can be stated in a more general setting, we focus on self-collision for necklaces. More precisely, suppose that $\mathcal{B} = \langle B_1, B_2, \ldots, B_n \rangle$ is a necklace with n beads, and T is a bounding sphere hierarchy built on $\mathcal{B}$. Each internal node v in T stores a cage $\mathcal{C}_v$ on the subset $\mathcal{B}_v$ of spheres stored at the leaves of the subtree rooted at v. A family $\Sigma = \{(\mathcal{C}_{u_i}, \mathcal{C}_{v_i})\}$, where u_i, v_i are nodes of T, is called a *separating family* for B if:

(S1) for any i, $\mathcal{C}_{u_i}$ and $\mathcal{C}_{v_i}$ are disjoint, and

(S2) for any i, j, where $|j - i| > 1$, there exists a k so that $B_i \in \mathcal{C}_{u_k}$ and $B_j \in \mathcal{C}_{v_k}$.

The size of Σ is the number of pairs in the family.

A separating family Σ serves as a proof of non-collision between the nonadjacent beads of $\mathcal{B}$. The minimum size of a separating family is crucial as it provides a lower bound on the cost of any collision-detection algorithm that uses the hierarchy T and follows the general approach described in Figure 6. We analyze the size of the separating for the wrapped hierarchy of $\mathcal{B}$.

There has been some prior research on constructing separating families for a set of balls—construct a family $\mathcal{P}$ of "canonical subsets" of $\mathcal{B}$ and compute a separating family of $\mathcal{B}$ using the canonical subsets in $\mathcal{P}$. If we are allowed to define $\mathcal{P}$ arbitrarily, i.e., it is not the set of cages stored at the nodes of a bounding-volume hierarchy on $\mathcal{B}$, there always exists a set of $O(n)$ separating pairs for n disjoint balls [18, 4]. However, to our knowledge, the separating families based on predefined hierarchical structures have not been studied combinatorially. Here, we will show that for the wrapped hierarchy in $\mathbb{R}^d$, there always exists a separating family of size $O(\max\{n \log n, n^{2-2/d}\})$ if there is no collision between any pair of two non-adjacent beads.

Theorem 3.1 *Let $\mathcal{B} = \{B_1, B_2, \cdots, B_n\}$ be a sequence of n beads in $\mathbb{R}^d$, satisfying the uniformity assumption. Then there exists a separating family for $\mathcal{B}$ of size $O(\max\{n \log n, n^{2-2/d}\})$ in its wrapped hierarchy. This bound is asymptotically tight in the worst case.*

Proof. We assume that the minimum radius of the beads is 1. By Lemma 2.1, there are at most $O(R^d)$ beads contained in any ball with radius R. We now give an algorithm for constructing a separating family Σ.

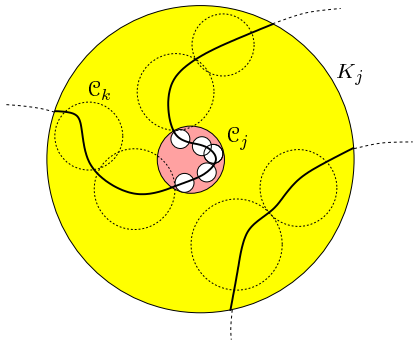


Figure 7. Separating pairs in Σ_j^i.

Fix an integer $0 \leq i \leq \log n - 1$. Set $r_i = 2^i$. Let $\mathcal{C}_j$ be the cage in the wrapped hierarchy that encloses the beads $B_{(j-1)r_i+1}, \ldots, B_{jr_i}$. Clearly, the radius of $\mathcal{C}_j$ is at most $R_i = O(r_i)$, by Lemma 2.1. Let $\Xi^i = \{\mathcal{C}_1, \mathcal{C}_2, \ldots\}$ be the resulting set of spheres; $|\Xi^i| \leq \lceil n/2^i \rceil$. For each $\mathcal{C}_j \in \Xi^i$, let K_j be the set of points that are at most $8R_i$ distance away from a point in $\mathcal{C}_j$, i.e.,

$$K_j = \{x \mid \exists y \in \mathcal{C}_j \, |xy| \leq 8R_i\}.$$

K_j is a ball of radius at most $9R_i$ concentric with $\mathcal{C}_j$. For any ball $\mathcal{C}_k \in \Xi^i$ so that $\mathcal{C}_k \subseteq K_j \setminus \mathcal{C}_j$, we add the pair $(\mathcal{C}_j, \mathcal{C}_k)$ to the separating family. Let Σ_j^i be the set of pairs added to the family for $\mathcal{C}_j$. We repeat this process for all balls in Ξ^i, and set $\Sigma^i = \bigcup_j \Sigma_j^i$. Set

$$\Sigma = \bigcup_{i=0}^{\log n - 1} \Sigma^i.$$

We claim that Σ is a separating family for $\mathcal{B}$. It is obvious from the construction that all the pairs in Σ are disjoint. We need to argue that it covers all the pairs of beads. Consider a pair of disjoint beads (B_ℓ, B_m). Denote by $\mathcal{C}_\ell^i$ and $\mathcal{C}_m^i$ the i-th level (the level increases bottom up starting from 0) cages that contain B_ℓ and B_m, respectively. Let $\nu = \max\left\{i \mid \mathcal{C}_\ell^i \cap \mathcal{C}_m^i = \emptyset\right\}$. It is easy to see that every point in $\mathcal{C}_\ell^\nu$ is within distance $4R_{\nu+1} = 8R_\nu$ from the center of $\mathcal{C}_m^\nu$ because $\mathcal{C}_\ell^{\nu+1}$ and $\mathcal{C}_m^{\nu+1}$ intersect. Therefore $(\mathcal{C}_\ell^\nu, \mathcal{C}_m^\nu) \in \Sigma$. Hence, Σ is a separating family.

Next, we bound the size of Σ. Note that

$$\sum_{i=\frac{1}{d}\log n+1}^{\log n-1} |\Xi^i| = \sum_{i=\frac{1}{d}\log n+1}^{\log n-1} \left\lceil \frac{n}{2^i} \right\rceil = O(n^{1-1/d}).$$

Since $|\Sigma^i| \leq \binom{|\Xi^i|}{2}$,

$$\sum_{i=\frac{1}{d}\log n+1}^{\log n-1} |\Sigma^i| = O(n^{2-2/d})\,.$$

It thus suffices to bound $|\Sigma^i|$ for $0 \leq i \leq (1/d)\log n$.

Every pair $(\mathcal{C}_j, \mathcal{C}_k)$ in Σ_j^i "covers" $r_i^2 = 2^{2i}$ pairs of beads. For any pair of beads (B_u, B_v) covered by this pair, their centers are within distance $9R_i$ because $\mathcal{C}_k \subseteq K_j$. Therefore, Σ^i covers $2^{2i}|\Sigma^i|$ pairs of beads. By a packing argument, there are at most $O((9R_i)^d) = O(2^{di})$ beads whose centers are within distance $9R_i$ from the center of the bead B_u. Hence, $2^{2i}|\Sigma^i| = O(n2^{di})$, which implies that $|\Sigma^i| = O(n2^{(d-2)i})$. Therefore,

$$\begin{aligned} |\Sigma| &= O(n^{2-2/d}) + \sum_{i=0}^{\frac{1}{d}\log n} O(n2^{(d-2)i}) \\ &= O(\max\{n\log n, n^{2-2/d}\})\,. \end{aligned}$$

This completes the proof of the upper bound.

To show the bound tight in the worst case, consider an $n^{1/d} \times \cdots \times n^{1/d}$ d-dimensional grid. We can form a necklace $\mathcal{B}$ by tracing and connecting the segments parallel to the x-axis in the grid. By the construction, $\mathcal{B}$ contains $n^{1-1/d}$ x-axis aligned segments, each of length $n^{1/d}$. We claim that any separating family of $\mathcal{B}$ in its wrapped hierarchy has size $\Omega(\max\{n\log n, n^{2-2/d}\})$. First we show that for two parallel segments, each with m beads, at distance δ where $\delta \in [i, i+1)$, we need $\Omega(m/i)$ pairs to cover the beads on them. Suppose that $A_1, A_2, \ldots, A_m$ and $B_1, B_2, \ldots, B_m$ are the beads on the two parallel segments. Consider the $\Omega(m/i)$ pairs $P = \left\{(A_{i\cdot j}, B_{i\cdot j}) | 1 \leq j \leq \left\lfloor \frac{m}{i} \right\rfloor\right\}$. Since the two segments are at most $i+1$ away from each other, it is impossible to separate two pairs in P using the same pair of cages. Thus, it requires $\Omega(m/i)$ pairs to separate the two segments.

On the other hand, if we project each line segment to the subspace orthogonal to the x-axis, then each line segment becomes a lattice point in $d-1$ dimensional space. The bound on the number of lattice points in spherical shells implies that there are $\Omega(i^{d-2})$ lattice points in the distance range

$[i, i+1]$ from a lattice point in $\mathbb{R}^{d-1}$. Therefore, there are $\Omega(n^{1-1/d} i^{d-2})$ pairs of segments at distance δ apart for $\delta \in [i, i+1)$. Further, we can pick the segments so that any cage containing two beads on different segments has radius comparable to the minimum enclosing sphere of the whole necklace. For example, in two dimensions, we pick all the line segments corresponding to the odd rows. Since there are $\Omega(n^{1/d})$ beads on each segment, the number of pairs in any separating family is

$$\sum_{i=1}^{n^{1/d}} \Omega\left(n^{1-1/d} \cdot i^{d-2} \frac{n^{1/d}}{i} \right) = \sum_{i=1}^{n^{1/d}} \Omega(n i^{d-3}) = \Omega(\max\{n \log n, n^{2-2/d}\}) .$$

□

¿From Theorem 3.1, it follows that there always exists a collection of subquadratically many ($O(n \log n)$ for $d = 2$ and $O(n^{2-2/d})$ for $d \geq 3$) separating pairs if any two non-adjacent beads are disjoint. The above constructive proof also suggests the following simple balancing heuristic in deciding which node to split: we always split the node that contains more beads and break the ties arbitrarily. Then in the process of the algorithm, the cages we compare always contain similar number of beads. Therefore, the proof of Theorem 3.1 applies — the number of pairs examined by the algorithm is bounded by the quantity given in Theorem 3.1. Therefore, we conclude the following that

Theorem 3.2 *Let $\mathcal{B}$ be a necklace in $\mathbb{R}^d$ with n beads. Using its wrapped hierarchy $\mathcal{W}(\mathcal{B})$, we can determine in time $O(\max\{n \log n, n^{2-2/d}\})$ time whether two nonadjacent beads in $\mathcal{B}$ intersect.*

If we wish to determine whether two necklaces $\mathcal{B}_1$ and $\mathcal{B}_2$ intersect, we invoke the algorithm BVH-COLLISION-DETECT by inititializing the queue to (ρ_1, ρ_2), where ρ_i is the root of the wrapped hierarchy of $\mathcal{B}_i$. The same analysis implies the following.

Corollary 3.3 *Let $\mathcal{B}_1, \mathcal{B}_2$ be two necklaces in $\mathbb{R}^d$ with n beads each. Using their wrapped hierarchies, we can determine in time $O(\max\{n \log n, n^{2-2/d}\})$ time whether $\mathcal{B}_1, \mathcal{B}_2$ intersect.*

Remark. We can also use layered hierarchy for collision detection between two necklaces, but no subquadratic bound on the size of a separating family based on the layered hierarchy is currently known. However, we believe combining the separating set bound for the wrapped hierarchy, Theorem 3.1, with cage radius ratio bound, Theorem 2.3, should yield a subquadratic bound for the layered hierarchy.

3.2 Using power diagrams

Theorem 3.1 gives us a subquadratic algorithm for collision detection between two necklaces, using the wrapped hierarchy. While the bound is $O(n \log n)$ in $\mathbb{R}^2$, it is $\Theta(n^{4/3})$ in $\mathbb{R}^3$. The algorithm takes $\Omega(n^{4/3})$ time when the necklaces are tightly packed (e.g. globular proteins) since many cages in the hierarchy overlap with each other and the algorithm has to traverse deep down the hierarchy before being able to locate disjoint cages. A recent result by Erickson [10] shows that the Delaunay

triangulation has linear complexity for a dense set of points. Although that result does not directly apply to the power diagrams (see below for the definition), we may still expect that a similar result holds for the power diagram of a dense set of balls with comparable sizes. Guibas *et al.* [17] had shown that the closest pair of balls are neighbors in the power diagram if the balls are pairwise disjoint. Therefore we may use the power diagram for detecting collision between a set of pairwise-disjoint balls. We, however, allow consecutive beads to overlap, so that result no longer applies. We first show that we can still use the power diagram to perform collision detection between two necklaces, and then discuss how to maintain the power diagram under our motion model. We prove our result about the power diagram in a more general setting.

Let $\mathcal{B} = \{B_1, \ldots, B_n\}$ be a collection of balls in $\mathbb{R}^d$, and let c_i, r_i be the center and radius of B_i, respectively. The *power* of a point $x \in \mathbb{R}^d$ to B_i, denoted as $\pi(x, B_i)$, is

$$\pi(x, B_i) = \|x - c_i\|^2 - r_i^2.$$

The *power cell* of B_i is

$$V(B_i) = \{x \in \mathbb{R}^d \mid \pi(x, B_i) \leq \pi(x, B_j)\ \forall 1 \leq i \leq n\}.$$

The power cells of spheres in $\mathcal{B}$ cover $\mathbb{R}^d$. The *power diagram* of $\mathcal{B}$ is the decomposition of the

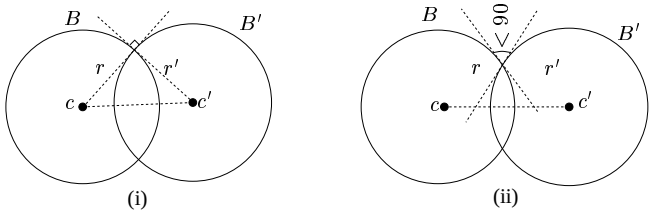


Figure 8. (i) Two orthogonal balls, (ii) two balls farther than orthogonal from each other.

space induced by these power cells and their boundaries. The dual of the power diagram of B is called the *weighted Delaunay triangulation* of $\mathcal{B}$. Assuming that the spheres are in general position, a simplex $\mathrm{conv}(A)$, for $A \subseteq \mathcal{B}$ and $|A| \leq d+1$, is in $DT(\mathcal{B})$ if $\bigcap_{B_i \in A} V(B_i) \neq \emptyset$. See [7] for details on power diagrams and weighted Delaunay triangulations. Two balls B and B' of radius r and r' and centered at c and c' are called *orthogonal* if

$$\pi(B, B') = \|c - c'\|^2 - r^2 - r'^2 = 0,$$

i.e., they intersect and the angle between their tangent planes at any of their intersection points is 90°. We say that B is *farther than orthogonal* from B' is $\pi(B, B') > 0$ (i.e., the angle between their tangent planes is less than 90°); see Figure 8. It can be checked that if a ball centered at a point $x \in \mathbb{R}^d$ is orthogonal to both B_i and B_j, then $\pi(x, B_i) = \pi(x, B_j)$. The following well-known lemma will be useful for our purpose:

Lemma 3.4 *A subset $A \subset \mathcal{B}$ form a simplex in $DT(\mathcal{B})$ if there is a ball B that is orthogonal to every ball in A and every ball in $\mathcal{B} \setminus A$ farther than orthogonal from B.*

Let $\mathcal{P} \subset \mathcal{B} \times \mathcal{B}$ be a family of pairs of balls. $\mathcal{B}$ is *disjoint with respect to* $\mathcal{P}$ if every pair of distinct balls in $\mathcal{B} \times \mathcal{B} \setminus \mathcal{P}$ is disjoint. For example, a necklace is disjoint with respect to the set $\mathcal{P} = \{(B_i, B_{i+1}) \mid 1 \leq i \leq n-1\}$. We define the *closest pair* of balls in $\mathcal{B}$ to be a pair of distinct balls $(B_i, B_j) \in \mathcal{B} \times \mathcal{B} \setminus \mathcal{P}$ so that $d(B_i, B_j) \leq d(B_k, B_l)$ for all $(B_k, B_l) \notin \mathcal{P}$; recall that $d(B_i, B_j) = |c_i c_j| - r_i - r_j$.

For any given $\mathcal{P}$, a ball A is a *neighbor* of B if the pair (A, B) is in $\mathcal{P}$. A ball A is a *neighbor* of a pair (B, C) if A is a neighbor of either B or C. A ball B is called a *proper connector* if for every neighboring ball A of B, there is another neighbor C of B such that A and C are disjoint. Balls that are not proper connectors are called *improper*. If the balls are disjoint, then there are no improper balls. Only the first and the last beads are improper in a necklace. We have the following result.

Theorem 3.5 *The closest pair $(B_i, B_j) \in \mathcal{B} \times \mathcal{B}$ satisfies at least one of the following properties:*

(DT1) (B_i, B_j) is an edge in $DT(\mathcal{B})$,

(DT2) B_i and B_j have a common neighbor, or

(DT3) B_i or B_j has an improper ball as a neighbor.

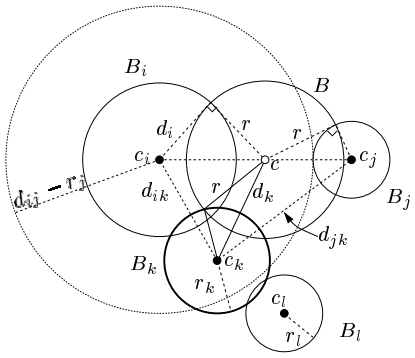


Figure 9. The power diagram can be used for collision detection in necklaces. The ball B_k cannot intersect ball O more than orthogonally, certifying the power diagram edge B_iB_j. x is the distance between B_i and B_j.

Proof. Suppose that (B_i, B_j) is the closest pair of balls in $\mathcal{B}$ with respect to $\mathcal{P}$. Assume that (B_i, B_j) does not satisfy (DT2) and (DT3). We would like to show that B_iB_j is an edge in $DT(\mathcal{B})$.

Let B be the smallest ball orthogonal to both balls B_i and B_j, and let r and c be the radius and center of B, respectively, see Figure 3.2. Consider another ball B_k, where $k \neq i, j$, in $\mathcal{B}$. We would like to show that B_k does not intersect B more than orthogonally. According to [17], it is sufficient to consider those balls intersecting either B_i or B_j. Since B_i and B_j do not share common neighbor, we assume, without loss of generality, that B_k intersect B_i but disjoint from B_j. For a ball $B_u \in \mathcal{B}$, let $d_u = \|cc_u\|$, and for any pair of balls $B_u, B_v \in \mathcal{B}$, let $d_{uv} = \|c_u c_v\|$. By construction,

$$d_{ij} = d_i + d_j.$$

We list the following conditions that various distances have to satisfy:

(C1) By orthogonality,
$$d_i^2 = r^2 + r_i^2, \quad \text{and} \quad d_j^2 = r^2 + r_j^2.$$

(C2) Since B_j and B_k are disjoint and (B_i, B_j) is the closest pair, the distance between B_i and B_j is not more than that between B_j B_k, i.e.,
$$d_{jk} \geq d_{ij} - r_i + r_k.$$

(C3) The triangle inequality $d_{ik} \geq d_{jk} - d_{ij}$ and (C2) imply
$$d_{ik} \geq r_k - r_i.$$

(C4) since B_k is proper, there is another neighbor ball B_l of B_k so that B_i, B_l are disjoint. Since (B_i, B_j) is the closest pair, $d_{ij} - r_i - r_j \leq d_{il} - r_i - r_l$. On the other hand, B_k and B_l intersect, therefore $d_{ik} + r_k > d_{il} - r_l$. Hence,
$$d_{ik} \geq d_{ij} - r_j - r_k.$$

Given the above relations, we derive that the ball B_k is farther than orthogonal from B, i.e.
$$d_k^2 \geq r_k^2 + r^2.$$

¿From Equation (1) in the proof of Lemma 2.4 (substituting $a_1 = d_i$, $a_2 = d_j$, $d_1 = d_{ik}$, and $d_2 = d_{jk}$) and condition (C2) above, we obtain
$$d_k^2 = \frac{d_j d_{ik}^2 + d_i d_{jk}^2}{d_{ij}} - d_i d_j \geq \frac{d_j d_{ik}^2 + d_i (d_{ij} - r_i + r_k)^2}{d_{ij}} - d_i d_j \, .$$

There are two cases to consider, depending on which of $r_k - r_i$ and $d_{ij} - r_j - r_k$ is larger. We substitute the larger of the two with d_{ik}:

1. If $r_k - r_i \geq d_{ij} - r_j - r_k$, i.e., $r_k \geq (d_{ij} + r_i - r_j)/2$, we obtain
$$\begin{aligned}
d_k^2 &\geq \frac{d_j (r_k - r_i)^2 + d_i (r_k + d_{ij} - r_i)^2}{d_{ij}} - d_i d_j \\
&= r_k^2 + \frac{2 r_k}{d_{ij}} \left(-d_j r_i + d_i (d_{ij} - r_i) \right) + \frac{1}{d_{ij}} \left(d_j r_i^2 + d_i (d_{ij} - r_i)^2 \right) - d_i d_j \\
&= r_k^2 + 2 r_k (d_i - r_i) + r_i^2 + d_i d_{ij} - 2 d_i r_i - d_i d_j \\
&= r_k^2 + 2 r_k (d_i - r_i) + (d_i - r_i)^2 \\
&\geq r_k^2 + (d_i - r_i)(d_{ij} + r_i - r_j) + (d_i - r_i)^2 \\
&= r_k^2 + (d_i - r_i)(d_i + r_i) + (d_i - r_i)(d_j - r_j) + (d_i - r_i)^2 \\
&= r_k^2 + r^2 + (d_i - r_i)(d_j - r_j) + (d_i - r_i)^2 \\
&\geq r_k^2 + r^2.
\end{aligned}$$

2. When $r_k < (d_{ij} + r_i - r_j)/2$, similarly, we get

$$\begin{aligned} d_k^2 &\geq \frac{d_j(d_{ij} - r_j - r_k)^2 + d_i(d_{ij} - r_i + r_k)^2}{d_{ij}} - d_i d_j \\ &= r_k^2 + \frac{2r_k}{d_{ij}}(d_i(d_{ij} - r_i) - d_j(d_{ij} - r_j)) + \\ &\quad \frac{d_j(d_{ij} - r_j)^2 + d_i(d_{ij} - r_i)^2}{d_{ij}} - d_i d_j \\ &= r_k^2 + \frac{2r_k}{d_{ij}}(d_i(d_i - r_i) - d_j(d_j - r_j)) + \\ &\quad (d_i + d_j)^2 - 2(d_j r_j + d_i r_i) + \frac{d_j r_j^2 + d_i r_i^2}{d_i + d_j} - d_i d_j \,. \end{aligned}$$

Note that

$$\begin{aligned} \frac{d_j r_j^2 + d_i r_i^2}{d_i + d_j} &= \frac{d_j(d_j^2 - r^2) + d_i(d_i^2 - r^2)}{d_i + d_j} \\ &= d_i^2 - d_i d_j + d_j^2 - r^2 \\ &= r_i^2 - d_i d_j + r_j^2 + r^2 \,, \end{aligned}$$

and thus,

$$d_k^2 \geq r_k^2 + r^2 + \frac{2r_k}{d_{ij}}(d_i(d_i - r_i) - d_j(d_j - r_j)) + (d_i - r_i)^2 + (d_j - r_j)^2 \,.$$

If $d_i(d_i - r_i) \geq d_j(d_j - r_j)$, it is clear that $d_k^2 \geq r_k^2 + r^2$. If not, using $r_k > (d_{ij} + r_i - r_j)/2$, we have that

$$\begin{aligned} d_k^2 &> r_k^2 + r^2 + \frac{1}{d_i + d_j}((d_i + r_i) + (d_j - r_j))(d_i(d_i - r_i) - d_j(d_j - r_j)) \\ &\quad + (d_i - r_i)^2 + (d_j - r_j)^2 \,. \end{aligned}$$

Using the fact that $d_i^2 - r_i^2 = d_j^2 - r_j^2$, we can simplify

$$\begin{aligned} &((d_i + r_i) + (d_j - r_j))(d_i(d_i - r_i) - d_j(d_j - r_j)) \\ &= d_i(d_j^2 - r_j^2) - (d_i + r_i)d_j(d_j - r_j) + (d_j - r_j)(d_i(d_i - r_i) - d_j(d_j - r_j)) \\ &= (d_j - r_j)(d_i d_j + d_i r_j - d_i d_j - d_j r_i + d_i^2 - d_i r_i - d_j^2 + d_j r_j) \\ &= (d_j - r_j)(d_i + d_j)(d_i - r_i - d_j + r_i) \,. \end{aligned}$$

It follows that

$$d_k^2 > r_k^2 + r^2 + (d_i - r_i)^2 + (d_j - r_j)(d_i - r_i).$$

In all cases, $d_k^2 \geq r_k^2 + r^2$. □

Theorem 3.5 gives us a way to check self-collision for a necklace: we can first compute the power diagram of all the beads and then check every pair for each power diagram edge. In addition, we check those pairs which share common neighbors, i.e. pairs (B_i, B_{i+2}) for $1 \leq i < n-1$, and those pairs involving B_2 or B_{n-1} (the only beads having an improper neighbor in the necklace). Clearly, the number of additional pairs is $O(n)$, and the cost of the checking is dominated by the complexity of power diagram, which we expect to be linear for a densely packed necklace.

Under the KDS motion model, it is easy to maintain the power diagram [16]. For the discrete time step model, however, it can be too expensive to recompute the power diagram at each time step. We have a variety of techniques for updating the diagram. One simple and, in practice, fast, solution is to remove some subset of the balls such that the power diagrams of the remaining balls before and after the time step are combinatorially identical, move the balls to the final locations, and then reinsert the removed balls. Such subsets can be found in a variety of manners, the simplest one being to repeatedly remove a point incident on each failed certificate, or repeatedly remove the point incident on the most failed certificates.

Another approach is to convert the discrete-time-step situation to the kinetic data structure framework by creating a motion which interpolates between the initial and final locations of the balls and then apply the kinetic data structure technology. Since the motion is artificial, we can design the motion so that the certificate failure times are easy to compute, or the number of events is small. For example, Edelsbrunner *et al.* [5] describe a motion so that each sphere moves along a straight line in the standard lifting space. We do not elaborate on these options here as they are not the focus of this paper.

4 Experimental Results

We conducted a variety of experiments to test the static and dynamic properties of the wrapped hierarchy and to compare it with the layered hierarchy and the power diagram. The properties of interest were:

1. the *cost of construction*—in terms of the time taken (Table 1),
2. the *cost of verification* the hierarchy, using both the naive method (direct verification of bead inclusion in the cages) and the cascade verification—in terms of both the time taken and the number of intersection tests performed (Table 5),
3. the *cost of collision detection*—in terms of time, the size of the set of separating pairs, and the number of intersection tests performed (Tables 3, 4, 7), and
4. the *frequency of basis changes* as the shape deforms and the *cost of updating* the hierarchies (Tables 5 and 6).

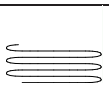
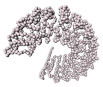
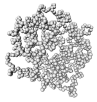
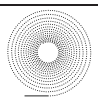
Name	*Picture*	*Necklace size*	*Wrapped time (ms)*	*Layered time (ms)*	ϱ_{avg}	ϱ_{min}	*Power diagram time (ms)*
grill		89	0.24	0.08	.98	.80	3.2
1a4yA0		1380	5.00	1.40	0.92	0.56	270.0
1cem00		1089	2.80	1.10	0.92	0.54	210.0
spline		10	0.02	0.01	0.97	0.86	0.1
		50	0.12	0.05	0.99	0.81	2.4
		100	0.25	0.09	1.00	0.80	9.3
static helix		10	0.03	0.01	0.89	0.70	0.0
		640	2.00	0.60	0.98	0.46	310.0
		10240	35.00	9.10	1.00	0.45	8.8×10^4
scaling helix		10	0.03	0.01	0.93	0.61	0.1
		640	2.10	0.60	0.98	0.46	220.0
		10240	38.00	9.00	0.99	0.45	1.8×10^4
straight spiral curled		1000	2.80	1.00	1.00	1.00	2.6
		1000	3.00	1.00	0.99	0.50	350.0
		1000	2.90	0.90	1.00	0.50	243.0
villin		108	0.27	0.10	0.97	0.83	9.8
		108	0.29	0.10	0.93	0.63	9.0
		108	0.31	0.10	0.93	0.63	10.1
random protein		300	0.81	0.27	0.97	0.83	41.0
		300	0.83	0.27	0.95	0.70	29.0
		300	0.85	0.27	0.94	0.68	41.0

Table 1. Construction costs for the wrapped and the layered hierarchy and the power diagram; ϱ_{avg} (resp. ϱ_{min}) is the average (resp. minimum) value of the ratio of the radii of the cages in the two hierarchies, taken over all the nodes in the tree.

We used three different types of inputs for our experiments:

PROTEINS. taken from PDB files [33] (*1a4yA0* and *1cem00*) or from smaller models (*random protein* and *Villin* headpiece); the latter were used for molecular simulations.

SPLINES. A simple quadratic spline with five control points.

HELICES. All helices used for our experiments are contained in a cubical bounding box. We used two types of helices: the first one, called the *constant helix*, had a constant number of turns, and the second one, called the *scaling helix*, had a number of turns that scaled with the number of points sampled from the helix. This shape is a bad case for the power diagram [9].

SPIRALS AND GRILLS. The spiral series was an animation of a straight segment rolling up in to an 18 turn logarithmic spiral. The grill had six parallel straight segments connected by parabolic arcs.

Table 1 provides pictures of the models used for our experiments.

All the experiments were conducted on a Dell PC with a 2.4 GHz Pentium 4 CPU and 1 gigabyte of RAM, and we used the CGAL library [13] for computing the minimum enclosing sphere of a set of spheres.

4.1 Static properties

In this subsection we discuss our experimental results on the construction and verification of the hierarchy and on collision detection using these hierarchies.

Construction cost. Empirically, the cost of construction depends solely on the number of beads used. It took approximately 35ms to build the wrapped hierarchy for a necklace with 10,240 beads. In general, the time dependence on the number of beads agrees well with the expected $O(n \log n)$ cost, and the running time depends very little on the shape of the necklace. Although the construction cost of the wrapped hierarchy is significantly better than that of the power diagram (88s for a helix with 10,240 beads), the greater simplicity of the layered hierarchy makes its construction even faster (9ms for the same helix).

On average the wrapped and layered hierarchies produced very similar sized cages (Table 1, columns 3 and 4), generally withing 5%. For each node v in the hierarchy tree, let $r_v(\mathcal{W})$ (resp. $r_v(\mathcal{L})$) be the radius of the cage stored at v in the wrapped (resp. layered) hierarchy, and let $\varrho_v = r_v(\mathcal{W})/r_v(\mathcal{L})$. We compute ϱ_{avg} (resp. ϱ_{min}), the average (resp. minimum) value of ρ_v over all nodes v of the tree. As expected, the ϱ_{avg} was most different for the protein backbones—the most curved necklaces tested—where $r_v(\mathcal{W})$ was 8% smaller. The value of ϱ_{avg} is heavily biased toward the numerous nodes of the tree near the leaves, where the wrapped and layered hierarchies hardly differ. This explains why the value of ϱ_{avg} is close to 1. A few of the wrapped hierarchy cages were

much smaller (often a factor of two) than their respective layered cage. These tended to be near the root and so are more important for collision detection of disjoint objects. The value $\varrho_{\min}$ illustrates this phenomenon.

Name	*Lazy update (ms)*	*Cascade verify (ms)*	*Naive verification (ms)*	*Average frontier size*
spline				
10	0.005	0.031	0.028	1.5
100	0.04	0.31	0.34	3.0
1000	4.00	3.10	4.60	2.0
helix				
10	0.01	0.03	0.03	2.0
10240	4.00	54.00	51.00	4.0
1a4ya0	0.61	6.00	6.40	2.8
1cem00	0.53	4.60	4.90	2.7
grill	0.04	0.31	0.33	2.5

Table 2. Verifying and updating the wrapped hierarchy using various approaches. The verification frontier of a node is the self collision separating set for the leaves of the subtree rooted at the node. The average frontier size is the size of such sets averaged over all nodes in the tree.

Verification cost. Surprisingly, the two methods of verification—naive and cascade—performed very similarly over the models tested. The largest difference found was on extremely highly sampled spline curves. In this case the cascade verification was 25% faster than the naive method. This improvement stems from the fact that all the bounding spheres in such a curve are diametrical, i.e., they touch the subnecklace at its endpoint, and only the descendants that contain the two basis beads need to be checked. Another advantage of the cascade method is that the frontier can be cached and reused in later time steps. The average frontier size stayed a small constant for even the most convoluted curves; it was below 10 per node in the tree for all curves tested. These figures can be seen in Tables 5, 2.

Collision detection. We used the static models to investigate the performance of the self-collision-detection algorithm. The cascade based algorithms performed much better than the brute force quadratic algorithms, as expected. The performance of the two hierarchies was quite similar under cascade collision detection for all of the models except for the large protein backbones. There the smaller sphere sizes of the wrapped cages resulted in a modest speed increase (Table 3).

The separating sets produced by the cascade algorithm were much smaller than the number of edges in the power diagram (its effective separating set size), with the exception of the highly sample scaling helix. The results on constant and scaling helices (Table 3) illustrate the dependence of the separating set on packing. When the number of points in a static helix increased by a factor

Name	*Necklace size*	*Quadratic collision time (μs)*	*Wrapped cascade col. time (μs)*	*Wrapped separating set size*	*Layered cascade col. time (μs)*	*Layered separating set size*	*Power diagram edges*
grill	89	410	70	293	71	291	264
1a4yA0	1380	6300	770	2759	950	2759	8500
1cem00	1089	63000	530	2177	700	2177	8600
	10	10	2	9	2	9	27
spline	50	130	13	52	13	52	147
	100	540	26	103	26	101	297
	10	10	7	27	7	27	40
static helix	640	21000	360	1500	360	1500	57000
	10240	5.6×10^6	3000	11000	3000	11000	2.0×10^6
	10	100	3	11	3	11	55
scaling helix	640	3.1×10^5	450	1500	450	1500	45000
	10240	5.6×10^6	11000	4.7×10^4	11000	4.7×10^4	5.6×10^5

Table 3. Detecting self-collision using various approaches. The naive methods tests collision between all pairs of beads, the power-diagram based algorithm tests the beads that share an edge, and the hierarchy based method tests pairs of cages that are in the separating set.

Name	*Necklace size*	*Quadratic collision time (μs)*	*Wrapped cascade col. time (μs)*	*Wrapped separating set size*	*Layered cascade col. time (μs)*	*Layered separating set size*	*Power diagram edges*
spiral, straight	1000	53000	240	1000	250	1000	3000
	1000	53000	720	2400	740	2400	3000
curled	1000	53000	280	6400	270	7200	3000
	108	610	28	110	28	110	731
villin	108	620	31	120	38	150	719
	108	620	35	140	38	150	708
	300	480	76	310	78	315	2200
random protein	300	470	83	600	100	600	2100
	300	490	93	380	106	420	2000

Table 4. Applying self-collision-detection algorithm from scratch at each step as the necklace deforms. We use molecular dynamic simulations and the rolling of a spiral as the test sets.

of 16 from 640 to 10240, the size of the separating set per bead of the necklace decreased from 2.34 to 1.07. This can be attributed to the fact that as the sampling increased, the curve became locally straighter and straighter (on the scale of the bead separation). On the other hand, when the scaling helix necklace size was similarly increased from 2 to 4, reflecting the greater curvature and packing in the neighborhood of each bead, the separating set increased from 2.34 to 4.58.

4.2 Dynamic properties

We used three data sets for testing the dynamic properties of the wrapped hierarchy: we performed molecular dynamics simulation on Villin and *random protein*, using large time steps (2.0×10^{-14}s) in order to cover greater conformational changes of the respective proteins. We also simulated the rolling of a straight line into a spiral; the spiral was evenly sampled along its length.

The wrapped hierarchy was very stable under the deformations of the underlying necklace in our tests. As illustrated in Table 5, there were only a couple basis changes per time step in the wrapped hierarchy for molecular dynamics simulation, whereas the power diagram had over 100 differing tetrahedron at each step of the same simulation. The cascade verification performed significantly better than the naive verification over all the simulations (despite their similar performance in the artificial test cases). Finally, we looked at the effect of varying the length of the time-step in Table 6. The wrapped hierarchy flattens out at around 15 basis changes for any time-step between 2 and 32 frames, while the amount of damage to the power diagram increased approximately logarithmically.

4.3 Taking advantage of the wrapped hierarchy

The great simplicity of the layered hierarchy makes its computation from scratch (which has to be done at each time step of a simulation) faster than updating the wrapped hierarchy. As a result, we have to to modify our techniques to properly take advantage of the wrapped hierarchy.

One advantage of the wrapped hierarchy over the layered hierarchy is that it provides an approximate bounding hierarchy when the beads are allowed to move (even if no verification is performed). Assuming all the beads in a cage move rigidly, the minimum enclosing sphere of the basis before a time-step is the minimum enclosing sphere after the time step. When the beads do not move rigidly but move coherently, the cage is still approximately valid. We call experiments which exploit this property (and make the assumption of its correctness) *lazy*. The update computation can be 10 times cheaper than the verification computation, as shown in Table 5. In addition, a cage in the wrapped hierarchy is defined without reference to any other cages, so individual cages can be computed independently.

Exploiting these two properties in some situations can yield simulations that are much faster than can be achieved with the layered hierarchy. Such an example of such a situation is shown in Table 7. There, two disjoint copies of the spiral were rolled independently, while checking for collisions between them (but not internal collisions). The cascade collision detection algorithm starts looking at cages from the top of the tree and cascades down until a separating set is found. Only the

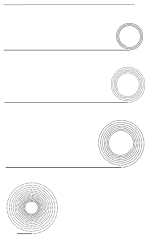
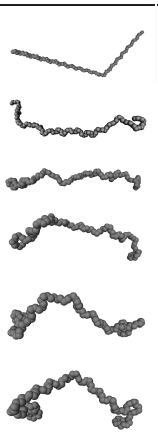
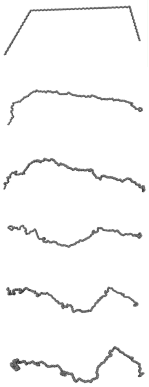
Name	*Picture*	*Power diagram cell changes*	*Wrapped Hierarchy*			
			basis changes	*lazy update (ms)*	*cascade verify (ms)*	*naive verify (ms)*
spiral		16	0	0.69	1.7	2.0
		1100	9	0.73	2.0	2.3
		1400	16	0.72	2.0	2.2
		1614	26	0.72	2.0	2.1
		1800	43	0.73	2.1	2.2
villin		172	0	0.08	0.20	0.2
		125	4	0.08	0.21	.2
		135	4	0.08	0.21	0.2
		130	4	0.08	0.23	0.2
		155	4	0.08	0.22	0.2
		136	6	0.08	0.24	0.2
random protein		611	0	0.22	0.60	0.6
		428	4	0.21	0.66	0.6
		461	11	0.22	0.66	0.6
		436	7	0.22	0.63	0.6
		438	11	0.22	0.66	0.6
		480	11	0.22	0.63	0.6

Table 5. Maintaining the changes in the wrapped hierarchy and the power diagram as the necklace deforms. For the spiral, there are 50 evenly spaced frames as it rolls up from a straight line to a 16 turn spiral, and in the molecular simulations, the adjacent frames are 2.0×10^{-14}s apart. We measure the changes in the structure at each time step and the cost of updating it at each time step.

Number of frames	*changes*	*Wrapped*			*Power diagram changes*
		lazy update (ms)	*cascade verify (ms)*	*naive verification (ms)*	
1	7	0.25	0.66	0.62	462
2	15	0.22	0.66	0.62	608
4	12	0.22	0.63	0.61	869
8	16	0.22	0.66	0.63	974
16	15	0.22	0.65	0.63	1090
32	14	0.22	0.66	0.63	1238

Table 6. Stability of the collision-detection techniques over varying number of frames. The *random protein* model is used and all frame pairs start with frame 900 of 1000 in the simulation.

Conformation	*Layered*		*Wrapped*			
	seperating set	*time (ms)*	*seperating set*	*lazy time (ms)*	*full update time (ms)*	*Average frontier*
	6	0.9	6	0.006	0.7	1.7
	6	1.0	5	0.006	0.6	2.1
	6	1.0	6	0.01	0.7	2.3
	2400	2.0	1600	1.0	2.0	2.7
	3700	2.1	2300	2.0	2.0	3.0

Table 7. Collision detection between two rolling spirals, using the wrapped and layered hierarchies. The entire layered hierarchy is updated at each time step. In the wrapped hierarchy, at each time step, we either update all the cage bases that become invalid (*full update*) or we update only at those nodes that are visited by the collision-detection algorithm (*lazy update*). We also show the average size of the cascade verification frontier over all the nodes in the tree.

cages touched were recomputed for the wrapped hierarchy, resulting in a factor of 200 speedup when the curves we relatively far away from one another (as they were when fully extended). As the curves wound up they got closer, so the wrapped advantage decreased until the wrapped and layered computation times were nearly equal when the two spirals were almost touching.

5 Conclusions

This paper raises a new set of issues in geometric computing, by posing the problem of how to repair and maintain geometric structures under small motions or deformations of their defining elements. Efficient geometric structure repair is essential in complex physical simulations, virtual reality animations, as well as when tracking moving real world objects. More generally, additional research is needed on how to better integrate geometric algorithms with physical models of objects.

Even for our simple example of a deforming necklace, several basic questions remain open:

- Can we prove bounds on the number of combinatorial changes in the wrapped hierarchy, assuming a physical model of deformation and a given 'deformation energy budget' that limits the overall bending nd oscillations that can occur?
- How can we best integrate the power diagram and the sphere hierarchy so as to get the advantages of each?
- What properties of a physical model can be exploited to make hierarchy updates faster?

We hope to address some of these issues in the near future.

Acknowledgments The authors wish to thank John Hershberger, Menelaos Karavelas, Rachel Kolodny, and Man-Cho A. So for their helpful discussions and Erik Lindhahl for the molecular simulation data.

References

[1] P. K. Agarwal, J. Basch, L. J. Guibas, J. Hershberger, and L. Zhang. Deformable free space tiling for kinetic collision detection. *Intl. J. Robotics Research*, 21:179–197, 2002.

[2] G. Barequet, B. Chazelle, L. J. Guibas, J. S. B. Mitchell, and A. Tal. BOXTREE: A hierarchical representation for surfaces in 3D. *Computer Graphics Forum*, 15(3):387–396, 1996.

[3] J. Basch, L. J. Guibas, and J. Hershberger. Data structures for mobile data. *Journal of Algorithms*, 31:1–28, 1999.

[4] P. B. Callahan and S. R. Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *Journal of ACM*, 42:67–90, 1995.

[5] H.-L. Cheng, H. Edelsbrunner, and P. Fu. Shape space from deformation. *Comput. Geom. Theory Appl.*, 19:191–204, 2001.

[6] S. De and K. J. Bathe. The method of finite spheres. *Computational Mechanics*, 25:329–345, 2000.

[7] H. Edelsbrunner. *Geometry and Topology for Mesh Generation*. Cambridge University Press, New York, 2001.

[8] H. Edelsbrunner, G. Rote, and E. Welzl. Testing the necklace condition for shortest tours and optimal factors in the plane. *Theoret. Comput. Sci.*, 66:157–180, 1989.

[9] J. Erickson. Nice point sets can have nasty delaunay triangulations. In *Proc. 17th Annual ACM Symposium on Computational Geometry*, pages 96–105, 2001.

[10] J. Erickson. Dense point sets have sparse Delaunay triangulations. In *Proc. 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 125–134, 2002.

[11] J. Erickson. Local polyhedra and geometric graphs. In *Proc. 14th ACM-SIAM Sympos. on Discrete Algorithms*, pages 171–180, 2003.

[12] J. Erickson, L. J. Guibas, J. Stolfi, and L. Zhang. Separation sensitive collision detection for convex objects. In *Proc. 10th ACM-SIAM Sympos. on Discrete Algorithms*, pages 327–336, 1999.

[13] A. Fabri, G.-J. Giezeman, L. Kettner, S. Schirra, and S. Schönherr. The CGAL kernel: A basis for geometric computation. In M. C. Lin and D. Manocha, editors, *Proc. 1st ACM Workshop on Appl. Comput. Geom.*, volume 1148 of *Lecture Notes Comput. Sci.*, pages 191–202. Springer-Verlag, 1996.

[14] S. Gottschalk, M. Lin, and D. Manocha. OBB-Tree: A hierarchical structure for rapid interference detection. In *SIGGRAPH 96 Conference Proceedings*, pages 171–180, 1996.

[15] L. J. Guibas. Kinetic data structures — a state of the art report. In *Proc. 3rd Workshop on Algorithmic Foundations of Robotics (WAFR)*, pages 191–209, 1998.

[16] L. J. Guibas, F. Xie, and L. Zhang. Kinetic data structures for efficient simulation. In *Proc. IEEE International Conference on Robotics and Automation (Vol. 3)*, pages 2903–2910, 2001.

[17] L. J. Guibas and L. Zhang. Euclidean proximity and power diagrams. In *Proc. 10th Canadian Conference on Computational Geometry*, pages 90–91, 1998.

[18] J. E. Hopcroft, J. T. Schwartz, and M. Sharir. Efficient detection of intersections among spheres. *Internat. J. Robot. Res.*, 2(4):77–80, 1983.

[19] P. M. Hubbard. Collision detection for interactive graphics applications. *IEEE Trans. on Visualization and Computer Graphics*, 1:218–320, 1995.

[20] B. G. K. Fischere. The smallest enclosing ball of balls: Combinatorial structure and algorithms. In *Proc. 19th Annual ACM Symposium on Computational Geometry*, pages 292–301, 2003.

[21] D. Kirkpatrick, J. Snoeyink, and B. Speckmann. Kinetic collision detection for simple polygons. In *Proc. 16th Sympos. Comput. Geom.*, pages 322–330, 2000.

[22] D. Kirkpatrick and B. Speckmann. Kinetic maintenance of context-sensitive hierarchical representations for disjoint simple polygons. In *Proc. 18th Sympos. Comput. Geom.*, pages 179–188, 2002.

[23] J. Klosowski, M. Held, J. S. B. Mitchell, K. Zikan, and H. Sowizral. Efficient collision detection using bounding volume hierarchies of k-DOPs. *IEEE Trans. Visualizat. Comput. Graph.*, 4(1):21–36, 1998.

[24] M. C. Lin and S. Gottschalk. Collision detection between geometric models: a survey. In *Proc. of IMA Conference on Mathematics of Surfaces*, pages 37–56, 1998.

[25] I. Lotan, F. Schwarzer, D. Halperin, and J.-C. Latombe. Efficient maintenance and self-collision testing for kinematic chains. In *Proc. 18th Annual ACM Symposium on Computational Geometry*, pages 43–52, 2002.

[26] N. Megiddo. Linear-time algorithms for linear programming in R^3 and related problems. In *Proc. 23rd Annu. IEEE Sympos. Found. Comput. Sci.*, pages 329–338, 1982.

[27] M. K. Ponamgi, D. Manocha, and M. C. Lin. Incremental algorithms for collision detection between general solid models. In *Proc. ACM Siggraph Sympos. Solid Modeling*, pages 293–304, 1995.

[28] C. M. Procopiuc, P. K. Agarwal, and S. Har-Peled. Star-tree: An efficent self-adjusting index for moving points. In *Proc. 4th Workshop on Algorithm Engineering and Experiments*, pages 178–193, 2002.

[29] S. Quinlan. Efficient distance computation between non-convex objects. In *Proc. IEEE International Conference on Robotics and Automation*, pages 3324–3329, 1994.

[30] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, Reading, MA, 1990.

[31] S. Šaltenis, C. S. Jensen, S. T. Leutenegger, and M. A. Lopez. Indexing the positions of continuously moving objects. In *Proc. ACM SIGMOD International Conference on Management of Data*, pages 331–342, 2000.

[32] E. Welzl. Smallest enclosing disks (balls and ellipsoids). In H. Maurer, editor, *New Results and New Trends in Computer Science*, volume 555 of *Lecture Notes Comput. Sci.*, pages 359–370. Springer-Verlag, 1991.

[33] J. Westbrook, Z. Feng, L. Chen, H. Yang, and H. M. Berman. The Protein Data Bank and structural genomics. *Nucleic Acids Research*, 31(1):489–491, 2003.

[34] Y. Zhou and S. Suri. Analysis of a bounding box heuristic for object intersection. *Journal of ACM*, 46:833–857, 1999.