Collision Detection for Deforming Necklaces ⋆

Pankaj Agarwal [a] Leonidas Guibas [b] An Nguyen [b] Daniel Russel [b]
Li Zhang [c]

[a] *Department of Comp. Sci., Duke University, Durham, NC 27708*
[b] *Department of Comp. Sci., Stanford University, Stanford, CA 94305*
[c] *Compaq Systems Res. Center, 130 Lytton Avenue, Palo Alto, CA 94301*

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. In this paper, we exploit this linearity to develop geometric structures associated with necklaces that are useful 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 built on a necklace which can be used for collision and self-collision detection. Such a hierarchy is easy to compute and is suitable for maintenance when the necklace deforms, as our theoretical and experimental results show. Using this hierarchy, we achieve an upper bound of $O(n \log n)$ in two dimensions and $O(n^{2-2/d})$ in d-dimensions, $d \geq 3$, for collision checking. To our knowledge, this is the first sub-quadratic 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 also used to detect self-collisions of a necklace and is, in certain ways, complementary to the sphere hierarchy.

Key words: deformable chains, collision detection, bounding volume

⋆ A preliminary version of this paper appeared in the Proc. of the 18th ACM Symp. on Computational Geometry, 2002. This research was supported by NSF grants CCR-9910633 and ITR-0086013 and a Stanford Graduate Fellowship.

Email addresses: pankaj@cs.duke.edu (Pankaj Agarwal), guibas@cs.stanford.edu (Leonidas Guibas), anguyen@cs.stanford.edu (An Nguyen), drussel@cs.stanford.edu (Daniel Russel), l.zhang@compaq.com (Li Zhang).

1 Introduction

The computational study of geometry has been motivated in part by the desire to model physical systems whose description naturally involves geometric attributes, such as shape. Indeed, efficient algorithms and data structures for handling geometric data are needed in almost every computational field that deals with the physical world, including computer graphics, computer vision, robotics, geographic information systems, spatial databases, molecular biology, and scientific computing. Recently there has been increased interest in modeling time-varying phenomena and this has naturally led to the study of geometric objects under motion.

Early work on moving objects in computational geometry, initiated by Atallah [3], focused on bounding the number of combinatorial changes in geometric structures as the objects move along prescribed trajectories [1]. In the late 1980s algorithms were developed for detecting and computing the actual changes in geometric structures as objects undergo motion, e.g. [26,2]. However, these results assume an off-line model of motion in which the trajectories are given in advance, which is a relatively uncommon situation. More recently, the *Kinetic Data Structures Framework* (or KDS for short) [13,4] was introduced to allow on-line changes to the motion. In the KDS framework certificates are used to ensure the correctness of a certain computation of the attribute of interest. Motion can cause certificates to fail; at such events the KDS repair mechanism is invoked to fix the attribute computation so that the simulation can proceed. Though the KDS view has provided an elegant framework for the analysis of motion algorithms, certain requirements limit its applicability, especially the need to predict certificate failure times. This is hard to do in many real-world situations where a physical system is evolving according to an ordinary or partial differential equation, since the motion plans of the simulation elements are not known in closed form, and time-stepping methods must be used.

Another aspect of motion that has not been adequately modeled in previous work is that objects in the world are often organized into groups and hierarchies and the motions of objects in the same group are highly correlated. For example, though not all points in an elastic bouncing ball follow exactly the same rigid motion, the trajectories of nearby points are very similar and the overall motion is best described as the composition of a global rigid motion with a small local deformation. Similarly, the motion of an articulated figure, such as a man walking, is most succinctly described as a set of relative motions, say that of the upper right arm relative to the torso, rather than by giving the trajectory of each body part in world coordinates. All theoretical analysis to-date are based on the assumption of independently moving objects. By ignoring such motion coherence we run the danger of developing sub-optimal algorithms that do not exploit well the structure of the problem. A similar call for realistic input models in geometric algorithms was made in [7].

To begin addressing some of these issues, in this paper we propose to study a

model for deformable 'linear' objects such as macro-molecules, muscles, ropes, etc. Though our objects live in 2-D or 3-D, they have an essential one-dimensional character that we heavily exploit in our algorithms [1] . It is customary in engineering to model physical objects by breaking them up into 'elements' of various types — indeed this is the essence of the Finite-Element method [25]. Instead of a standard type of element (tetrahedra or hexahedra), we have chosen (potentially partially overlapping) spheres as our basic elements. The use of spheres simplifies substantially the basic geometric calculations and allows us to focus on the combinatorial issues that form our main interest. There is actually a literature on using spheres in engineering modeling [6] and for biomolecules spheres are obviously the right choice.

We call our linear objects *necklaces* and the spherical elements used to model them *beads*. The exact way in which a necklace moves and deforms depends on the physical model used and is application dependent. Our focus will be instead on tracking different geometric attributes of a necklace, such as its power diagram or a bounding sphere hierarchy that are useful in applications. For example, most forces in nature are short range and the power diagram provides useful proximity information among the beads. Collision and self-collision detection are important in most physical simulation settings and bounding volume hierarchies and the power diagram are both useful collision detection tools in such contexts. 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 position back to us. Though in general every single bead moves at every step, we assume that the time steps chosen by the simulator are such that at each step the motion of each bead is small when compared to the overall scale of the simulation. Unlike the kinetic data structure setting where we have explicit motion paths and can predict certificate failures, here we are called upon to repair a geometric structure after small displacements of its defining elements.

2 Bounding Volume Hierarchies

Bounding volume hierarchies for rigid objects have been extensively used as a data structure for collision detection. 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), arbitrarily oriented bounding boxes (OBBs) [12], k-DOPs [19], and spheres [24,17]; see [20] for a survey. The choice of

[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.

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. Once the bounding volume shape is chosen, we can proceed to build a bounding volume hierarchy for a given object A. This can be done by both bottom-up and top-down methods. In the bottom-up approach, small patches of the object A or its surface are each covered by an elementary bounding volume and these are then aggregated and the object geometry enclosed by each aggregate is further covered by a bounding volume, and so on in a hierarchical fashion, until A has been entirely enclosed in a single bounding volume. Once hierarchies for rigid objects A and B have been computed, they can be used for repeated intersection testing among the objects A and B in different positions and poses. For given placements of A and B, intersection testing proceeds by refining their respective hierarchies to only the coarsest level at which the primitive shapes in the two hierarchies can be shown to be pairwise disjoint — or until an intersection is detected.

Unquestionably, bounding volume hierarchies have been among the most successful methods used for collision detection in large and complex virtual environments and physical simulations. However, if the object bounded by a hierarchy not only moves but also deforms, the use of hierarchies becomes more problematic — current methods provide only a set of bad alternatives. One can expand the bounding volumes to allow some limited shape variability, but then more intersection tests will be needed because the bounding volumes do not fit as tightly. Or one can recompute the bounding volume hierarchy at each time step, but the cost of that is significant. Space time bounding volumes, as proposed in [17] can also be used to accommodate deformation, albeit not very efficiently.

One of the key contributions of this paper is to propose a bounding volume hierarchy for our deforming necklaces that can be efficiently maintained under deformation and yet remains tightly fitting. We are not aware of much published prior work on deformable bounding volume hierarchies — with the exception of [21] that appeared in conference form concurrenty with this paper. While we focus on physical simulations in which all elements of the deformable model move by a small amount at each time step, [21] focusses on Monte Carlo simulations of kinematic chains where one of the joints undergoes a large angle change at each step.

3 Our Approach and Results

The two geometric tools we investigate in this paper for proximity maintenance and collison detection of deforming necklaces are a bounding sphere hierarchy and the power diagram. Given that our atomic elements are themselves balls, our choice of spheres as our bounding volumes is natural. Spheres do not bound as tightly as oriented bounding boxes (in the limit they have linear as opposed to quadratic convergence to an underlying smooth shape [12]), but intersection/containment tests

among them are especially simple and their symmetry makes rigid motions straightforward to implement.

The various hierarchies discussed above for static geometry aggregate bounding volumes based on spatial proximity. When an object undergoes large deformations, however, spatial proximity is variable over time and cannot be used as a reliable guide to aggregation. We have decided to base the hierarchy we will use on *topological proximity* in the object, because this notion of proximity is better preserved. For our linear necklace this gives us an especially simple rule for aggregation: we build a balanced binary tree on the sequence of beads $\{B_1, B_2, \cdots, B_n\}$ so that the intermediate aggregates correspond to the sets of leaves that are descendants of internal nodes in the tree.

The specific sphere hierarchy we use is based on bounding volumes which are the smallest spheres containing relevant parts of the original geometry — which are contiguous substrings of beads in our setting. This gives each bounding sphere a small combinatorial description: such a sphere is determined by at most four of the enclosed beads (3 in 2-D). As our necklace deforms, the geometry of this sphere hierarchy also changes continuously. However, the combinatorial descriptions of these bounding spheres (in other words, the lists of beads defining them) change at only discrete events. This discrete nature of the updates makes it possible to maintain the bounding spheres efficiently under motion, in contrast to non-combinatorially defined bounding volumes

In general, a bounding volume hierarchy is formed by creating a balanced recursive partitioning of the underlying geometry and computing a bounding volume enclosing each group. Once the partitioning is determined, there are two ways to form the bounding volume hierarchy. One is, as we do, to compute a tight bounding volume on the geometry in each group, which we call the 'wrapped hierarchy', and the other is to compute the bounding volume of the bounding volumes of the children subgroups, which we call the 'layered hierarchy'. Clearly, the wrapped hierarchy is always tighter than the layered hierarchy. In this paper, we first study the relationship between the two hierarchies. We show the somewhat surprising result that, in the worst case, 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. Furthermore, the bound holds in any dimension.

The most important application of bounding hierarchies is in collision and self-collision checking. While such methods work well in practice, in the worst case nothing better than the trivial quadratic bound was previously known. This bound arises in the case where 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, with a simple heuristic, the folklore self-collision checking method using the sphere hierarchy and local refinement as necessary achieves sub-quadratic 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 sub-quadratic worst-case bound for collision detection algorithms using bounding volume hierarchies [2] .

The power diagram is another tool that people often use to deal with balls. While it has been known that the closest pair of a set of disjoint balls defines an edge in the power diagram ([15]), that result does not apply directly to our problem since we allow adjacent spherical elements to overlap. We show that the power diagram can be used to compute the closest pair in a deforming necklace as well. It is interesting to note that the worst-case for the sphere hierarchy occurs for highly packed necklaces, while these are actually very favorable cases for the power diagram — in such cases the power diagram size is linear in all dimensions [10]. While both of these structures can be used for proximity determination and collision checking, one of the key experimental results of this paper is that the sphere hierarchy is far more stable than the power diagram under deformation.

In Section 4 we present the formal setting of our beads and necklaces. Section 5 discusses the precise bounding sphere hierarchy we have chosen to implement and the reasons for our choice. We present verification and repair algorithms for maintaining the hierarchy as the necklace deforms, and compare the tightness of the wrapped and layered hierarchies. Section 6 presents a number of combinatorial results about collision detection using the sphere hierarchy or the power diagram. Section 7 discusses computational experiments for a number of different necklace scenarios, both static and dynamic, that study the performance of our sphere hierarchy and the power diagram under realistic conditions. Finally Section 8 concludes the paper.

4 Beads and Necklaces — Definitions

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 E^d. We assume that only adjacent balls along the necklace may intersect and no ball is fully contained in another set of balls. In some contexts we make further assumptions. These include:

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

limited overlap: two consecutive balls B_i and B_{i+1} along a necklace are allowed to overlap, but the angle of their normals at a common point on their surfaces is bounded below by β.

[2] A similar bound was reported concurrently with our work for oriented bounding boxes in [21]. As already mentioned, they studied kinematic chains similar to our necklaces, although with a different motivation.

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].

We define the following terms to be used in later sessions. In a sphere hierarchy built on top of a necklace, the internal nodes of the hierarchy are called *cages*. When the cage is defined as the minimum enclosing sphere (MES) of the beads it contains, the geometry of the cage is fully determined by a constant number of the beads which we call the *basis beads*.

The sphere hierarchy forms a balanced binary tree built on top of the necklace, which each bead of the necklace forming a leaf in the tree. Let $L(t)$ be the set set of beads corresponding to leaves of the subtree rooted at node t. Note that $L(t)$ is a contiguous subchain of the original necklace, which we will call the *canonical subnecklace*. Each cage corresponds to an internal node, t, of the tree and bounds all the beads in $L(t)$. This is one instance where we heavily use the *a priori* known structure of the type of object we are modeling.

5 The Wrapped Hierarchy

5.1 Definition and properties

We define the *wrapped hierarchy* of a necklace to be the sphere hierarchy corresponding to the balanced tree described above, where the sphere corresponding to each internal node is the *minimum enclosing sphere* (MES) of the beads in the canonical sub-necklace associated with that node. We call these bounding spheres corresponding to internal nodes *cages*. Note that this allows the cages of the children of a node in the hierarchy to stick out of the cage of the parent. We call the sphere hierarchy defined by making the cage of a parent to be the MES of the cages of its two children the *layered hierarchy* [24]. 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. An example of each type of hierarchy is shown in Figure 1.

The key property of the wrapped hierarchy that is of interest to us is that each cage, being a minimum enclosing sphere of a set of beads, is fully determined by a small number (two, three, or four in 3-D) of the basis beads in the associated canonical sub-necklace. Note that the cage of an internal node is also the minimum enclosing sphere of its basis beads. When a necklace deforms, the basis of each cage remains constant for a period. At certain discrete events the basis of a cage changes typically by a pivoting step in which (1) an old basis bead leaves the basis, and (2) a new bead

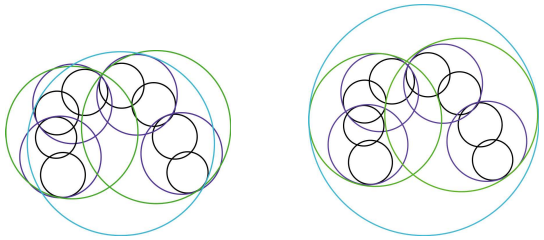

Fig. 1. Wrapped (left) and layered (right) sphere hierarchies. The base beads are black. Notice that each cage in the wrapped hierarchy is supported by 2 or 3 beads.

from the enclosed sub-necklace enters the basis. At times only one of these events may happen, but the total number of basis beads will always remain between two and four. Thus, although during continuous necklace deformation the cages deform continuously, their combinatorial descriptions stay constant and change only at discrete events. This combinatorialization of a continuous phenomenon is an insight analogous to what is exploited in kinetic data structures.

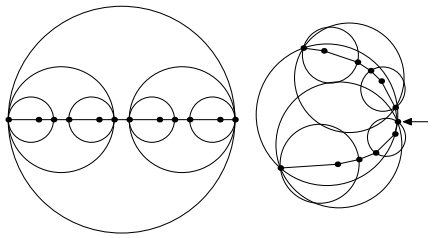

Fig. 2. A combinatorially defined sphere hierarchy is stable under deformation. Only the top level cage differs between the two conformations.

We expect that under smooth deformation the combinatorial description of the cages (i.e. their basis beads) will stay constant 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 2 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.

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. In the following, we consider the case where all the beads are points (or equivalently equal radius spheres), and in arbitrary position. We have the following result:

Theorem 5.1 *For any given set of n points in any dimension and any binary tree with depth $\lceil \log_2 n \rceil$ on the points, if we denote by τ_1, τ_2 the radii of the root spheres for the wrapped and layered hierarchies of the point set, respectively, then $\tau_2 \leq \tau_1 \sqrt{\lceil \log_2 n \rceil}$. The bound is almost perfectly tight, as we can construct a set of points so that $\tau_2 \geq \tau_1 \sqrt{\lfloor \log_2 n \rfloor}$.*

We denote the minimum enclosing sphere (MES) of a set of beads S by $M(S)$ and the MES corresponding to a node t by $M(t)$. The upper bound result is implied by the following lemma.

Lemma 5.2 *Let O and R be the center and the radius of $M(S)$, and O_ℓ and R_ℓ the center and the radius of a ball on level ℓ in the layered hierarchy of S. Let $d_\ell = |OO_\ell|$. Then $R_\ell^2 \leq \ell(R^2 - d_\ell^2)$.*

Proof. Without lost of generality, let us assume that $R = 1$. We prove the lemma using induction. The lemma is clearly true for the 0-th level, as $R_0^2 \leq (1 - d_0)^2 \leq 1 - d_0^2$. We assume that the lemma holds for all balls at level ℓ in the layered hierarchy and show that the lemma still holds for balls at level $\ell + 1$.

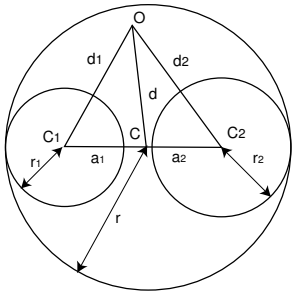

Fig. 3. A ball in the layered hierarchy and its two children. Lemma 5.2 proves that the farther the centers of the children are from the center of their parent, the smaller their radii must be in comparison.

Let C and r be the center and the radius of a ball at level $\ell + 1$ in the layered hierarchy, and let C_1, C_2 and r_1, r_2 the centers and radii of its two children. Let $d = |OC|$, $d_1 = |OC_1|$, and $d_2 = |OC_2|$, see Figure 3. By induction assumption, $r_1^2 \leq \ell(1-d_1^2)$ and $r_2^2 \leq \ell(1-d_2^2)$. We would like to show that $r^2 \leq (\ell+1)(1-d^2)$. This is clearly true when the parent ball at C is identical to one of its children balls. We consider the general case when the parent ball is bigger than both of its children.

From $\cos(\angle OCC_1) = -\cos(\angle OCC_2)$ and the law of cosines, we obtain:

$$\begin{aligned}
\frac{a_1^2 + d^2 - d_1^2}{2a_1 d} &= -\frac{a_2^2 + d^2 - d_2^2}{2a_2 d} \\
a_2(a_1^2 + d^2 - d_1^2) &= -a_1(a_2^2 + d^2 - d_2^2) \\
(a_1 + a_2)d^2 &= (a_2 d_1^2 + a_1 d_2^2) - (a_2 a_1^2 + a_1 a_2^2) \\
d^2 &= \frac{a_2 d_1^2 + a_1 d_2^2}{a_1 + a_2} - a_1 a_2 \,. \qquad (1)
\end{aligned}$$

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,

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

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 5.1 as the depth of the tree is bounded by $\lceil \log_2 n \rceil$. Furthermore, we can show that the inequality in Lemma 5.2 can be made tight, and we can construct a set of points to attain the upper bound.

Lemma 5.3 *There is a set of n points in the plane such that $\tau_2 \geq \tau_1\sqrt{\lfloor \log n \rfloor}$, where τ_1, τ_2 denote the radius of the root ball in the wrapped and layered hierarchy, respectively.*

Proof. It suffices to consider the case $n = 2^k$. We will construct a collection of points such that their wrapped hierarchy has radius 1 and their layered hierarchy has radius $\sqrt{k}$. The construction is done incrementally. We first fix any point O and place a point at O_0 such that $|OO_0| = 1$.

Suppose that we have constructed the set S_ℓ, the first 2^ℓ points. Let O_ℓ be the center of the ball 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) $|O_\ell O_{\ell+1}| = 1/\sqrt{k}$. We can then construct S'_ℓ by flipping all the points in S_ℓ about the line $OO_{\ell+1}$. Finally we set $S_{\ell+1} = S_\ell \cup S'_\ell$. See Figure 4.

It is straightforward to show that in the above incremental construction, the ball covering S_ℓ in the layer hierarchy has radius $\ell/\sqrt{k}$, and $|OO_\ell| = \sqrt{1-\ell/k}$. Thus

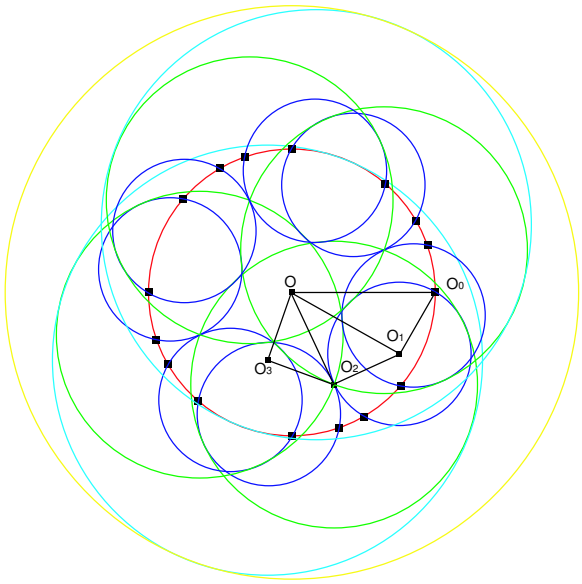

Fig. 4. The construction of a set of 16 points on a circle of radius 1 such that the root circle of their layered hierarchy has radius $\log(\sqrt{16}) = 2$. The point O_0 is chosen arbitrarily. Right triangles OO_0O_1, OO_1O_2, OO_2O_3 are then constructed such that $|O_0O_1| = |O_1O_2| = |O_2O_3| = 1/2$. 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 refection over the point O. The circles in the layered hierarchy are also shown.

we can always find the point $O_{\ell+1}$, as long as $\ell < k$. It is clear that the root ball of the wrapped hierarchy has unit radius as all the points constructed are on the same circle, and the root ball of the layered hierarchy has radius $\sqrt{k}$.

Theorem 5.1 is the combination of Lemmas 5.2 and 5.3. □

5.2 Construction and maintenance

It is straightforward to construct the wrapped hierarchy by directly computing the minimum enclosing ball for the canonical set of each cage. There is a complex algorithm for computing MES of a set of points or equal radius spheres in $O(n)$ time in the worst case [22] and a simple one running in expected $O(n)$ time [27]. If the basis beads are balls of different radii, the algorithm is slightly more complicated but with the same overall running time [18]. Therefore, it takes $O(n \log n)$ time to construct initially the wrapped hierarchy of the necklace of n beads.

To maintain the wrapped hierarchy, we need to verify the correctness of the hierarchy, i.e. the correctness of the basis beads at each internal node, after a simulation time step, and update those which are no longer correct. This task is simplified by the following observation:

Lemma 5.4 *Let $\mathcal{B}$ and $\mathcal{S}$ be sets of beads with $\mathcal{B} \subseteq \mathcal{S}$. If all beads $B \in \mathcal{S}$ are contained in $M(\mathcal{B})$, then $M(\mathcal{S}) = M(\mathcal{B})$. Thus, the basis for $M(\mathcal{S})$ is the same as the basis for $M(\mathcal{B})$.*

Note that the basis can shrink, so if $\mathcal{B}$ was the basis of $M(\mathcal{S})$ before an update, and after the update $M(\mathcal{S}) = M(\mathcal{B})$, then some (not necessarily proper) subset of $\mathcal{B}$ is the new basis for $M(\mathcal{S})$.

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 nodes of a subtree under a node c. We first compute the minimum enclosing sphere M of the basis beads from the previous time step. According to Lemma 5.4, in order to check the correctness of c, it is sufficient to check that all the beads in $L(c)$, are contained in $M(c)$. 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 c. At each step we take a node d out of the frontier and check if $M(d)$ is contained in M. If it is not, there are two cases — if the node is an internal node we add its children to F; otherwise it is a leaf, so we know that the node has escaped and the basis of node c 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 the basis for $M(c)$ is still valid. 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 if 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.

When beads escape from an enclosing cage c, a basis update must be performed. At least one of the escaped beads must be a basis bead for the new $M(c)$. The LP-type algorithm [27] 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 size of the canonical subchain.

6.1 Separating necklaces using the wrapped hierarchy

Bounding volume hierarchies are often used for collision detection algorithms between two objects or for self-collision detection. Typically, such an algorithm tries to derive a non-intersection proof by finding a sufficient set of *separating bounding volume pairs* while walking down the hierarchy. It reports a collision if it fails to produce such a set of pairs. More precisely, suppose that B is a set of n disjoint objects $\{B_1, B_2, \ldots, B_n\}$ (beads in our hierarchy) and T is a tree built on the elements of B. Each internal node t in T corresponds to a bounding volume containing the objects at the leaves of the subtree rooted at t. This way, we create a collection P of objects, among which n are basic objects. A set C of pairs $\{(c_i, c_i')\}$, where $c_i, c_i' \in P$, is called a collection of separating pairs for B if:

(1) for any i, c_i and c_i' are disjoint, and
(2) for any i, j, where $i \neq j$, there exists a k so that $B_i \in c_k$ and $B_j \in c_k'$.

Such a separating pair set serves as a proof of the non-collision of the set B, i.e. it exists if and only if all the objects in B are mutually disjoint. The minimum number of pairs needed to separate a set of objects is crucial as it is a lower bound of the cost of a collision detection algorithm taking such an approach. In our problem, the objects are beads and the bounding volumes are cages, and the hierarchical structure is the wrapped hierarchy. We also need to relax requirement 2 to $i \neq j-1, j, j+1$ as we allow adjacent beads to intersect each other. There has been some prior research on separating a set of balls. When we are allowed to group balls arbitrarily, there always exist a set of $O(n)$ separating pairs for n disjoint balls [16,5]. However, to our knowledge, the separating pairs for predefined hierarchical structures have not been studied combinatorially before. Here, we will show that for the wrapped hierarchy in d dimensions, there always exists a separating pairs collection (a *separating family*) with $O(\max\{n \log n, n^{2-2/d}\})$ members, if there is no collision between any pair of two non-adjacent beads. In the following, we do not distinguish a node in the tree and the cage built for that node.

Theorem 6.1 *Let $\mathcal{B} = \{B_1, B_2, \cdots, B_n\}$ be a sequence of n beads in d-dimensions (for $d \geq 2$), satisfying the uniformity and limited overlap assumptions. Then there exists a separating family for $\mathcal{B}$ of size $O(\max\{n \log n, n^{2-2/d}\})$ among the cages in its wrapped hierarchy.*

Proof. By the uniformity and limited overlap assumption, we have that there exists a constant $\alpha \geq 1$ such that the distance between the centers of two adjacent beads is in the interval $[1, \alpha]$. We give an algorithm for constructing a separating family Σ. Throughout the proof, $\log$ is understood as $\log_2$.

Fix an integer i such that $0 \leq i \leq \log n - 1$. Set $r_i = 2^i$. Let D_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 D_j is at most $R_i = r_i\alpha$. Let $\mathcal{D}^i = \{D_1, D_2, \ldots\}$ be the resulting set of balls. For each $D_j \in \mathcal{D}^i$, let K_j be the set of points that are distance at most $8R_i$ from a point in D_j, i.e., $K_j = \{x \mid \exists y \in D_j \quad |xy| \leq 8R_i\}$. K_j is a ball with radius at most $9R_i$ and concentric with D_j. For any ball $D_k \in \mathcal{D}^i$ that is contained in K_j and is disjoint from D_j, we report the pair (D_j, D_k). We define Σ^i_j to be the set of pairs reported for D_j. We repeat this process for all balls in $\mathcal{D}^i$ and set $\Sigma^i = \bigcup_j \Sigma^i_j$. Set $\Sigma = \bigcup_{i=0}^{\log n-1} \Sigma^i$.

We claim that Σ is a separating family for 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 D^i_ℓ and D^i_m the i-th level (the level increases bottom up starting from 0) cages that contain B_ℓ and B_m, respectively. Let $t = \max\{i \mid D^i_\ell \cap D^i_m = \emptyset\}$. It is easy to see that every point in D^t_ℓ is within distance $4R_{t+1} = 8R_t$ from the center of D^t_m because D^{t+1}_ℓ and D^{t+1}_m intersect. Therefore $(D^t_\ell, D^t_m) \in \Sigma$. Hence, Σ is a separating family.

Next, we bound the size of Σ. Note that

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

because each ball in this set corresponds to the cage at a node in the wrapped hierarchy of B whose depth is less than $(1/d)\log n$. Hence,

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

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

Every pair (D_j, D_k) in Σ^i_j "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 D_k is contained in K_j. Therefore, Σ^i covers $2^{2i}|\Sigma^i|$ pairs of beads with each pair is within distance $9R_i$. But by a packing argument, there are at most $(9R_i)^d = O(2^{di})$ beads whose centers are within distance $9R_i$ from the center of 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 theorem. □

Remark. The above bound is tight in the worst case. Consider an $n^{1/d} \times \cdots \times n^{1/d}$ d-dimensional grid. We can form a necklace B by tracing and connecting the segments parallel to the x-axis in the grid. By the construction, B contains $n^{1-1/d}$ x-axis aligned segments, each with length $n^{1/d}$. Any separating family of B in its wrapped hierarchy has size $\Omega(\max\{n \log n, n^{2-2/d}\})$. For two parallel segments at distance δ where $\delta \in [i, i+1)$, we need $\Theta(n^{1/d}/i)$ pairs to cover the beads on them. According to the bounds on the number of lattice points in spherical shells, we know that there are $\Omega(n^{1-1/d} i^{d-2})$ such pairs of segments. So the number of pairs in any separating family is

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

From Theorem 6.1, it follows that there always exists a collection of sub-quadratically many ($O(n \log n)$ in two dimensions and $O(n^{2-2/d})$ in d-dimensions for $d \geq 3$) separating pairs if any two non-adjacent balls are disjoint. The above constructive proof also suggests the following simple heuristic for collision detection using wrapped hierarchies: when two cages intersect, we always split the one containing more beads and break ties arbitrarily. This way, the cages we compare always have similar number of beads inside them, and their parent cages intersect. Therefore, the above proof applies — the number of pairs examined by the algorithm is bounded by Theorem 6.1. That is, such a collision detection algorithm takes time $O(n \log n)$ in two dimensions and $O(n^{2-2/d})$ in higher dimensions, in particular, it is $O(n^{4/3})$ in three dimensions.

6.2 Collision detection with the power diagram

Theorem 6.1 gives us a sub-quadratic bound on the running time of the collision detection method using the wrapped hierarchy. While the bound is $O(n \log n)$ in two dimensions, it is $\Theta(n^{4/3})$ in three dimensions. We observe that the wrapped sphere hierarchy is good for self-collision detection of a deforming necklace when the necklace is not too entangled. When the string of beads is highly packed, however, many cages in the hierarchy overlap with each other. We then have to traverse deeply down the hierarchy before being able to locate disjoint cages. According to a recent result by Jeff Erickson [10], the Delaunay triangulation has linear complexity for a dense set of points. Although that result does not directly apply to the power diagrams, we may still expect that a similar result holds for the power diagram of a dense set of balls with comparable sizes. It was shown in [15] that the closest pair of balls are neighbors in the power diagram if all the balls are disjoint, and therefore we may use the power diagram for collision detection of a set of disjoint balls. In our problem, however, consecutive beads may overlap and that result no longer applies. In the following, we first show that we can still use the power diagram to

perform collision detection, even when we allow some partial ball overlaps. Then, we discuss briefly how to maintain the power diagram under our motion model.

We prove our result about the power diagram in a more general setting. Let $\mathcal{B}$ be a collection of balls and P be a set of pairs of balls. $\mathcal{B}$ is disjoint with respect to P if every pair of balls are disjoint, unless it is in P. For example, a necklace is disjoint with respect to the set $P = \{(B_i, B_{i+1}) | 1 \leq i \leq n-1\}$. We define the closest pair of balls in $\mathcal{B}$ to be the pair with the minimum Euclidean distance among all the pairs *not* in P.

For any given P, a ball A is a neighbor of B if the pair (A, B) is in 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*. Note that when the balls are disjoint, there are no improper balls. For a necklace, only the first and last beads are improper. We then have the following result.

Theorem 6.2 *The closest pair of $\mathcal{B}$ must either (1) share an edge in the power diagram of $\mathcal{B}$, or (2) have a common neighbor, or (3) have an improper ball as a neighbor.*

Before we prove the theorem, let us briefly mention some terminology often used in the power diagram literature. Two balls are *orthogonal* if they intersect each other and the angle between the tangent planes at any of the common points is $90°$. When the two balls penetrate each other even deeper, they *intersect more than orthogonally*. It is well know that given two balls B_i and B_j, there is an edge between B_i and B_j in the power diagram if there is no ball intersecting both B_i and B_j more than orthogonally.

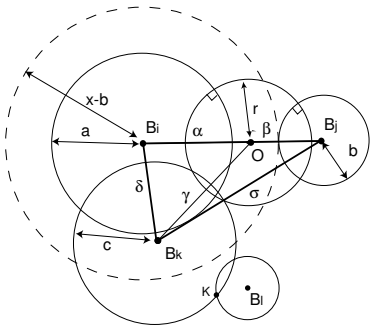

Fig. 5. The power diagram can be used for collision detection in necklaces. The ball B_k cannot intersect ball O more than orthogonally, certifying the power diagram edge B_iB_j. x is the distance between B_i and B_j.

Proof. Suppose that (B_i, B_j) is the closest pair of balls in $\mathcal{B}$ with respect to P. Assume that (B_i, B_j) does not satisfy (2) and (3). We would like to show that B_iB_j is an edge in the power diagram of $\mathcal{B}$.

Let O be the minimum ball orthogonal to both balls B_i and B_j, and let r, a, b be the radii of O, B_i, and B_j respectively (Figure 6.2). Consider another ball B_k, where $k \neq i, j$, in $\mathcal{B}$. We would like to show that B_k does not intersect O more than orthogonally. According to [15], 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. Let r be the radius of B_k. We denote the length of the line segments $OB_i, OB_j, OB_k, B_iB_k, B_jB_k$ by $\alpha, \beta, \gamma, \delta$, and σ, respectively, and the length of B_iB_j by $x = \alpha + \beta$. We list the following conditions those quantities have to satisfy:

(1) $\alpha^2 = r^2 + a^2$, and $\beta^2 = r^2 + b^2$ (by orthogonality).
(2) $\sigma \geq x - a + c$ (by the fact that that the distance between ball B_i and ball B_j should be no more than the distance between ball B_j and ball B_k).
(3) $\delta \geq c - a$ (by the triangle inequality $\delta \geq \sigma - x$, and condition 2).
(4) $\delta \geq x - b - c$ (since B_k is proper, there is another neighbor ball B_l of B_k so that B_i, B_l are disjoint, and so the distance between ball B_i and ball B_l is no less than the distance between ball B_i and ball B_j. If K is the intersection of balls B_k and B_l, then $\delta + c \geq |B_iK| \geq x - b$).

Given the above relations, we would like to derive that the ball B_k does not intersect ball O' more than orthogonally, i.e. $\gamma^2 \geq c^2 + r^2$.

From Equation (1) in the proof of Lemma 5.2, we have

$$\gamma^2 = \frac{\beta\delta^2 + \alpha\sigma^2}{x} - \alpha\beta\,.$$

We substitute $x - a + c$ for σ and the larger of $c - a$ or $x - b - c$ for δ to find a lower bound for γ^2.

(1) When $c - a \geq x - b - c$, i.e. $c \geq (x + a - b)/2$, we obtain

$$\begin{aligned}
\gamma^2 &\geq \frac{\beta(c-a)^2 + \alpha(c+x-a)^2}{x} - \alpha\beta \\
&= c^2 + \frac{2c}{x}(-\beta a + \alpha(x-a)) + \frac{1}{x}(\beta a^2 + \alpha(x-a)^2) - \alpha\beta \\
&= c^2 + 2c(\alpha - a) + a^2 + \alpha x - 2\alpha a - \alpha\beta \\
&= c^2 + 2c(\alpha - a) + (\alpha - a)^2 \\
&\geq c^2 + (\alpha - a)(x + a - b) + (\alpha - a)^2 \\
&= c^2 + (\alpha - a)(\alpha + a) + (\alpha - a)(\beta - b) + (\alpha - a)^2 \\
&= c^2 + r^2 + (\alpha - a)(\beta - b) + (\alpha - a)^2 \\
&\geq c^2 + r^2.
\end{aligned}$$

(2) When $c < (x + a - b)/2$, similarly, we get

$$\gamma^2 \geq \frac{\beta(x-b-c)^2 + \alpha(x-a+c)^2}{x} - \alpha\beta\,.$$

Manipulating the right hand side of the above inequality, we obtain:

$$\gamma^2 \geq c^2 + \frac{2c}{x}(\alpha(\alpha-a) - \beta(\beta-b)) + (\alpha-a)^2 + (\beta-b)^2 + r^2\,.$$

If $\alpha(\alpha-a) \geq \beta(\beta-b)$, it is clear that $\gamma^2 \geq c^2 + r^2$. If not, using $c > (x+a-b)/2$, we have that

$$\begin{aligned}\gamma^2 > & c^2 + r^2 + \frac{1}{\alpha+\beta}((\alpha+a) + (\beta-b))(\alpha(\alpha-a) - \beta(\beta-b)) \\ & + (\alpha-a)^2 + (\beta-b)^2\,.\end{aligned}$$

Simplifying the right hand side of the above inequality, we obtain

$$\gamma^2 > c^2 + r^2 + (\alpha-a)^2 + (\beta-b)(\alpha-a).$$

In all cases, $\gamma^2 \geq c^2 + r^2$. □

Theorem 6.2 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 [14]. 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. The discrete time step situation can be converted to the KDS setting by creating a motion which interpolates between the initial and final locations of the balls and then apply the KDS 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, we can continuously change the size of beads so that each bead moves on a straight line in the standard lifting space. Due to lack of space, we do not elaborate on these options here.

7 Experimental Results

⟨⟨**Not sure how to address the reviewers comment on this section**⟩⟩

We ran a variety of experiments to test the static and dynamic properties of the wrapped hierarchy. The sphere hierarchy was much more stable than the power diagram under motion, as expected. Validation of the hierarchy after each integrator step, as well as collision detection using it, were cheap in the situations where the necklace was loosely packed, but became expensive when the necklace was closely packed. All the timings were done on a 700Mhz PIII and use Bernd Gärtner's [11] implementation of Welzl's algorithm for computing minimum enclosing sphere of a set of points. The properties of interest were:

(1) the cost of construction, as measured by the time taken,
(2) the cost of verifying the hierarchy, using both the naive method (direct verification of bead inclusion in the cages) and the cascade verification, measured by both the time taken and the number of sphere inclusion tests performed,
(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, and
(4) the frequency of basis changes as the shape deforms.

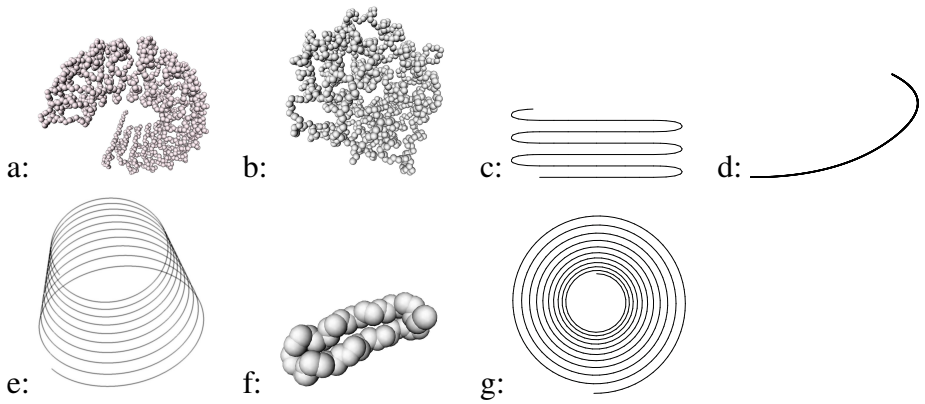

Fig. 6. The necklaces used for testing. *(a)* is the backbone of the protein 1a4yA0, *(b)* is the backbone of the protein 1cem00, *(c)* is a 2-D grill which is a bad case for self-collision detection, *(d)* is a spline sampled at a variety of resolutions, *(e)* is a helix which has a quadratic complexity power diagram, *(f)* is a beta hairpin backbone fragment, and *(g)* is a 2-D spiral. *(f)* and *(g)* were also used to test stability under motion.

A variety of static curves were used for testing the first three properties. Two of them, *(a)* and *(b)*, are protein backbones (1a4yA0 and 1cem00) consisting of slightly over 1000 atoms, each of whom was treated as having a radius of 2Å. The first protein has a great deal of secondary structure and is horse-shoe shaped, while the second is basically globular. The third, *(c)*, is a 2-D grill-like curve which is a bad case for collision checking, according to Theorem 6.1. We also sampled a simple spline, *(d)*, consisting of 2 quadratic segments, with 100 to 800 points. Our

next shape *(e)* was a helix. The helix has constant height and radius, but it has variable number of turns and sample points. We tried two cases: 'static' helixes with 10 turns and with 100 to 12,800 sample points, and 'scaling' helixes with n sample points and $\sqrt{n}$ turns, where n varies between 100 and 25,000. This shape is a bad case for the power diagram [9]. Some of the empirical results discussed below are summarized in Figure 7.

Empirically, the cost of construction depends solely on the number of beads used. It took approximately 42ms to build the wrapped hierarchy for a protein backbone with 1380 beads. In general, the time dependence on the number of beads agrees well with the expected $O(n \log n)$ cost.

Similarly, the naive verification cost is mostly independent of the geometry. In contrast, the cascade verification algorithm depends quite heavily on the necklace shape. As a consequence, which verification algorithm was faster varied from model to model. Cascade verification performed best when the curve was smooth compared to the number of samples, i.e. when the curve was crossing most of the spheres in a close to diametral fashion and there were only two basis beads. For example, the two algorithms performed similarly for the simple spline, *(d)*, when the sampling was at 100 points, but they diverged as the sampling increased, and cascade verification was twice as fast when *(d)* was sampled with 800 points. This high sampling was typical of the good cases for cascade verification in that the only descendant cages which stick out are those that contain the end-points of the canonical sub-necklace. As a result, the average verification cost per cage is constant and independent of the number of beads (since it is twice the average distance to the leaves of the tree).

At the other extreme, the naive algorithm was twice as fast as the cascade verification on the scaling helix *(e)*. On such a tightly coiled shape, most child cages stick out from their parents, consequently, cascade verification must explore the whole subtree before getting to the beads, resulting in almost twice as many intersection tests. The other experiments were somewhere in between the two extremes, and the two algorithms were generally fairly equivalent in time and work. The minimum times for verification ranged from 6ms for the smooth curve sampled with 800 points to .8s for the scaling helix with 25,600 beads.

When the objects are deforming, we can save the 'frontier' from the earlier time step in the cascade verification algorithm and use that as a starting point for the verification in the next time step, allowing a smooth transition between the two algorithms. Even in the worst case of the scaling helix with 25,000 beads, the frontier was less than 16 descendant nodes per cage being tested.

Collision checking performance was tested using the self-collision algorithm. The collision checking algorithm is fairly similar to the verification algorithm, so it is not surprising that the good and bad cases were similar. The spline, *(d)*, was the

Model	Nklc Size	S. Col. Tests	S. Col. Front.	Casc. Tests	Casc. Front.	Naive Tests
(a)	1380	9.8	5.7	7.6	5.8	13
(b)	1089	7.2	5.0	6.4	4.8	10
(c)	63	6.5	4.9	6.2	4.4	8.5
(d)	100	.98	1.4	7.0	4.2	8.6
	400	1.0	2	7.0	4.3	11
	1600	1.0	1.5	7.0	4.3	14
(e)	400	9.7	7.2	14	15	3.9
static	1600	3.5	3.3	14	7.7	14
	6400	1.7	2.0	13	6.9	16
(e)	100	19	11	12	5.7	8.6
scaling	400	28	22	16	8.1	11
	1600	41	32	19	10	14
	6.4k	166	130	23	12	16
	25k	672	530	28	15	19
(f)	51	6.0	3.3	5.6	3.5	7.2
(g), start	500	1.0	1.5	3.1	2.5	9.1
end	500	8.1	5.7	8.1	4.6	9.1

Fig. 7. "Nklc Size" is the number of beads in the necklace. "S. Col. Tests" is the average number of intersection tests for self-collision per hierarchy node, and "S. Col. Front." is the average size of the frontier per node. "Casc. Tests" is the number of comparisons for cascade verification per node, and "Casc. Front" is the frontier size per node. Finally, "Naive Tests" is the average number of comparisons for naive verification, namely the average number of leaves per node. The average costs per node are generally small except when the curve is very tightly packed (for example model (e) with a large number of points).

fastest to check, as the algorithm was able, in most cases, to simply verify that the two siblings did not intersect due to the small size of the beads compared to their separation. However, even if the beads did intersect, there would still be an average constant cost per node for reasons similar to verification. The separating set scaled linearly with the sampling of *(d)*, from 35 for a sampling of 100 beads to 575 for a sampling of 1600. Performance on the grill curve, *(c)* was as expected. While the separating set size for *(d)* sampled at 100 points was only 35, the set size for the

grill was 309, and the verification took correspondingly longer, at .6ms for the grill.

As before, the scaling helix was far and away the worst case. Self collision on the 25,600 bead sampling took 18 seconds, and the 100 bead sampling took 2ms. The separating sets were similarly enormous, being almost 10 million pairs for the high sampling rate and 1000 pairs for the 100 bead sample. This set is too large to hope for gains from a caching scheme analogous to that proposed for the verification stage. However, this is a pathologically bad case, and in many applications such situations can be ruled out.

A more interesting case was the static helix. When sampled with 200 points, self-collisions took 3.25ms, involved 14 checks per node and had a separating set of 2000 pairs. As the sampling rate increased, the number of checks per node decreased. At the highest sampling rate with 12,800 beads, it took 20ms and only involved 1.3 intersection tests per cage and the separating set was only 6,800 pairs. The reason for this is that as the sampling increases, amount of bending per bead decreases so the curve looks straighter and straighter on the scale of most of the spheres. As a result, a greater fraction of the collision tests became trivial ones as in the smooth curve case.

We had two data sets for testing the properties under dynamics. The first, *(f)*, was a 51 atom backbone fragment from a beta hairpin, simulated through 300 time steps using a molecular dynamics package for protein folding (Tinker) [23]. The initial configuration was with the backbone folded along itself, a situation which did not visibly change during the time steps considered. The radii of all the atoms were treated as 2Å for the purposes of building the wrapped hierarchy. The other dynamic case was a straight line segment rolling up into a spiral over 100 frames. The final form is shown in *(g)*. The spiral was evenly sampled at 500 points along its length.

The wrapped hierarchy was very stable under the deformations of the underlying necklace in our tests. For the hairpin simulation *(f)*, the cascade verification performed similarly to the naive verification, both averaging approximately 7 containment tests for each of the 37 nodes, and the separating set had a size of about 30. There were 19 basis changes over the 300 frames of the simulation. For comparison, we also built the power diagram and looked at its stability. The power diagram was much more complex with approximately 300 edges and 450 faces. The total number of changes was close to 1500.

For the spiral tests set *(g)*, the wrapped hierarchy had 487 cages. Since the sequence involved a change from a straight line (an optimal configuration for our algorithm), to a spiral (a near worst case configuration), the wrapped hierarchy verification performance varied quite considerably. In the first frame, cascade verification was twice as fast as naive and made 3 comparisons per node, while in the last frame cascade verify was slightly slower and took 8 inclusion tests per cage. Self-collision

fared even worse, with the separating set size going from 243 pairs to 2,868 pairs. There were a total of 500 basis changes. In comparison, the power diagram had 1000 edges in the first frame, and 498 faces, compared to 1,425 edges and 926 faces in the last frame. The total number of events in the power diagram was over 50,000.

The experiments show that the wrapped hierarchy is quite stable under deformation and efficient to maintain for elongated ot relatively unpacked shapes, while the verification and collision detection become expensive when the curve is tangled and becomes globular. The cost of self-collision checking and hierarchy verification seems to depend on the amount of bending at each bead, but we have so far been unable to quantify the dependence. The experiments also show that the power diagram is more complex than the wrapped hierarchy for the cases we tested — while the verification and collision checking cost for the power diagram is low, the maintenance cost is significantly higher. When dealing with curves that are loosely packed, the greater simplicity of the wrapped hierarchy clearly wins; the verdict it is less clear with necklaces that are more tightly packed. We can hope that a hybrid method exists that combines the best features of the sphere hierarchy and power diagram approaches.

8 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 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, many basic questions remain open:

- What are the trade-offs between the wrapped and layered hierarchies?
- 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 that can occur?
- how can we best integrate the power diagram and the sphere hierarchy so as to get the advantages of each?

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. Leonidas

Guibas, An Nguyen, and Daniel Russel acknowledge support under NSF grants CCR-9910633 and ITR-0086013, as well as from a Stanford Graduate Fellowship.

References

[1] P. K. Agarwal and M. Sharir. Arrangements and their applications. In J. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 49–119. Elsevier Science Publishers, 2000.

[2] G. Albers, L. J. Guibas, J. S. B. Mitchell, and T. Roos. Voronoi diagrams of moving points. *Internat. J. Comput. Geom. Appl.*, 8:365–380, 1998.

[3] M. J. Atallah. Some dynamic computational geometry problems. *Comput. Math. Appl.*, 11(12):1171–1181, 1985.

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

[5] 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.

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

[7] M. de Berg, M. J. Katz, A. F. van der Stappen, and J. Vleugels. Realistic input models for geometric algorithms. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 294–303, 1997.

[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] B. Gärtner. Fast and robust smallest enclosing balls. In *Proc. 7th Annual European Symposium on Algorithms (ESA)*, pages 325–338. Springer Verlag, Lecture Notes in Computer Science 1643, 1999.

[12] 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.

[13] 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.

[14] 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.

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

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

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

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] J. W. Ponder and F. M. Richards. An efficient newton-like method for molecular mechanics energy minimization of large molecules. *J. Comput. Chem.*, 8:1016–1026, 1987.

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

[25] J. N. Reddy. *An Introduction to the Finite Element Method*. McGraw-Hill, 1993.

[26] E. Schömer and C. Thiel. Efficient collision detection for moving polyhedra. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages 51–60, 1995.

[27] 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.