BD-Tree: Output-Sensitive Collision Detection for Reduced Deformable Models

Doug L. James
Carnegie Mellon University

Dinesh K. Pai
Rutgers University

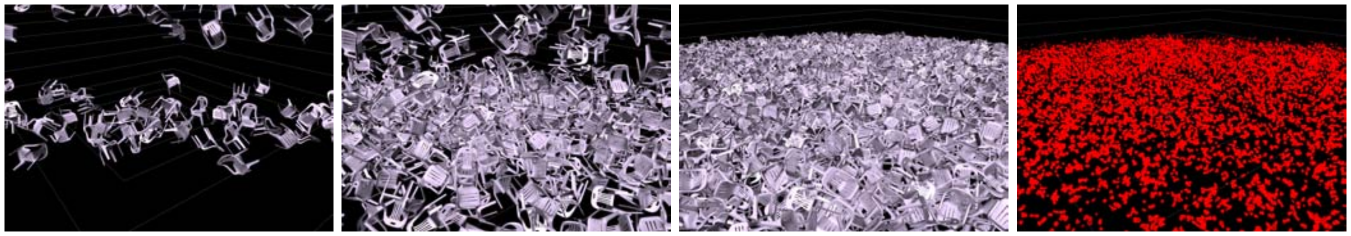


Figure 1: *BD-Tree simulation of 3601 plastic chairs crashing to the ground* with (far right) contacts approximated by small red spheres. Although the chairs flex and vibrate, collision processing was only twice as expensive as the rigid case.

Abstract

We introduce the Bounded Deformation Tree, or BD-Tree, which can perform collision detection with reduced deformable models at costs comparable to collision detection with rigid objects. Reduced deformable models represent complex deformations as linear superpositions of arbitrary displacement fields, and are used in a variety of applications of interactive computer graphics. The BD-Tree is a bounding sphere hierarchy for output-sensitive collision detection with such models. Its bounding spheres can be updated after deformation in any order, and at a cost independent of the geometric complexity of the model; in fact the cost can be as low as one multiplication and addition per tested sphere, and at most linear in the number of reduced deformation coordinates. We show that the BD-Tree is also extremely simple to implement, and performs well in practice for a variety of real-time and complex off-line deformable simulation examples.

CR Categories: I.3.5 [Computational Geometry and Object Modeling]: Physically based modeling

Keywords: output-sensitive, deformable, collision, sphere trees

1 Introduction

Interactive animation of large deformable models has become viable recently. A key reason is the realization that deformation can be separated from geometry, and described parsimoniously in *reduced deformation coordinates*. These can be computed efficiently with time complexity *sublinear* in the number of polygons. Another significant benefit of this separation is that modern GPUs are well suited for performing the unavoidable task of actually deforming the geometry prior to rendering; the host CPU only has to compute the reduced deformation coordinates and need not even have access to the full deformed geometry.

There are many examples of this approach. Freeform [Faloutsos et al. 1997] or multiresolution [Debunne et al. 2001] models use a small set of dynamic coordinates to deform complex geometry. Compressed animation sequences [Alexa and Müller 2000], can represent a shape animation as a small set of geometric principal component bases and a set of reduced animation coordinates; the latter can even be streamed in real time. Many skinning techniques, such as shape interpolation and Pose Space Deformation [Lewis et al. 2000], are also instances of this approach; the complex geometry is represented by a small set of keyshapes and the reduced deformation coordinates are functions of pose variables. Physically based deformation techniques such as DyRT [James and Pai 2002] represent the geometry as a static shape and a small set of mode shapes; the reduced deformation coordinates are the modal excitations. Many more examples exist, but in broad terms, the use of principal components, spectral decompositions, and multiresolution methods can lead to representations of this type.

Unfortunately, efficient sublinear methods have not emerged for collision detection (CD) with deformable objects; most deformable CD algorithms still involve computing the deformed position of each polygon to update a bounding volume hierarchy, with time complexity at least linear in the size of the models. This is unfortunate, because CD with a rigid bounding volume hierarchy can be done efficiently, in an output-sensitive way, with sublinear costs for typical situations without a large number of contacts (see Related Work). Exploiting temporal coherence can improve performance, but without assumptions about the deformation process the per-step costs of these approaches are inherently linear.

A sublinear algorithm would imply that we perform CD with a deformable model without explicitly computing its vertex positions. This may seem like an algorithmic conundrum. Fortunately, we show here that this is indeed possible for reduced deformation models based on the linear superposition of precomputed displacement fields, using a bounding sphere hierarchy that we call a **Bounded Deformation Tree**, or **BD-Tree**. Note that linear superpositions of displacement fields do not preclude modeling nonlinear deformation phenomena, as indicated above. The BD-Tree provides a foundation for a well balanced system for interactive deformable models, in which deformation computation and collision detection both have sublinear cost per step.

Our Contributions: We introduce the BD-Tree, which can perform output-sensitive collision detection for reduced deformable models at similar costs to standard algorithms for rigid bodies. The incremental cost, per tree node traversed, can be as low as *one* multiplication and *one* addition in the best case; in the worst case, it requires only $8M$ additional floating point operations per node, where M is the number of reduced deformation coordinates. This is typically a very small number, and independent of the size of the geometric model. Further, tree updates do not require access to deformed geometry, facilitating distributed processing with graphics hardware. Experimental results show that the algorithm works well in practice. The algorithm is also extremely easy to implement, and requires only small changes to existing sphere tree software.

1.1 Related Work

Many important methods have been developed for collision detection with rigid models. We refer the reader to these recent surveys: [Lin and Gottschalk 1998; Jimenez et al. 2001]. Sphere trees are a canonical example of a bounding volume hierarchy used for collision detection and proximity queries (e.g., [Quinlan 1994; Hubbard 1995; Brown et al. 2001; Guibas et al. 2002; Bradshaw and O'Sullivan 2004]). Collision detection with bounding volume hierarchies is output-sensitive and provides graceful degradation [Hubbard 1995; Dingliana and O'Sullivan 2000]. Hierarchy construction can be based on either spatial proximity between features within the *undeformed* model or on mesh topology, as well as inter-surface proximity (as in [Bridson et al. 2002] for cloth, and in [Warren and Weimer 2001] for subdivision hierarchies).

Many of these methods could be directly applied to deformable objects. The major cost for hierarchical collision detection algorithms is updating the hierarchy after every deformation, e.g., at each time step, before queries are performed. As a result, much attention has been drawn to hierarchies with easy to compute bounds (e.g., [van den Bergen 1997; Ganovelli et al. 2000]). Alternatively, intelligent methods for updating the bounding hierarchies and exploiting temporal coherence have been explored [Guibas et al. 2002; Brown et al. 2001; Larsson and Akenine-Möller 2001]. Larsson et al. [2003] propose a collision detection algorithm specialized for morphing that has features similar to ours (and is a special case). Because we use a more general deformation model (Eq. 1) our method can be applied in a more general setting (see second paragraph of Sec. 1), and we derive significantly less expensive bounds (Secs. 3.2 and 3.5).

Recently, hardware-accelerated collision detection methods have received increased attention; see [Manocha et al. 2002] for a recent overview. These general purpose methods are very useful, but tend to produce coarser approximations, since they are limited by the size of frame buffer memory, and not floating point precision. Note that the entire model has to be rendered for collision detection in these methods, and not just visible portions; a notable advantage of our approach is that it only requires touching a potentially small output-sensitive subset of model polygons.

2 Reduced Deformation Model

BD-Trees exploit spatially coherent motion that can be described as a combination of (hopefully smooth) displacement fields. Mathematically, we assume a general shape model based on the linear superposition of displacement basis functions that are known at the time of BD-Tree construction. The BD-Tree then tracks average motions associated with these displacement fields, and locally bounds their displacement deviations. As we will see, the fewer and more spatially coherent the displacement fields are, the better BD-Trees tend to perform.

Suppose we have N undeformed point locations, $\mathsf{p} = (\mathsf{p}_1, \dots, \mathsf{p}_N)^T$. Without loss of generality, we will assume that the deformed point locations, p', are approximated by a linear superposition of M displacement fields, given by the columns of U. The amplitude of each displacement field is given by corresponding reduced coordinates q, so that (see Figure 2)

$$\mathsf{p}' = \mathsf{p} + \mathsf{U}\mathsf{q} \qquad \text{or} \qquad \mathsf{p}'_i = \mathsf{p}_i + \sum_{j=1}^{M} \mathsf{U}_{ij}\mathsf{q}_j. \tag{1}$$

Note that the reduced coordinates q are determined by some other possibly nonlinear black box process; therefore, although the shape model is linear (1), the deformation process can be arbitrary. The columns of U represent displacement fields, or scaling functions, that could arise from, e.g., an interpolation process, multiresolution modeling, or modal analysis.

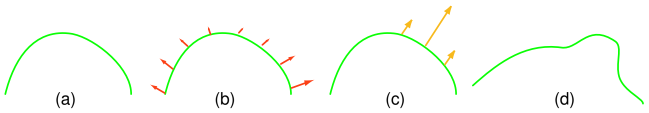


Figure 2: Example deformation: (a) Reference shape p (b) Displacement field U_{*1} (c) Field U_{*2} (d) Deformed shape p'.

3 Bounded Deformation Trees

This section describes the construction, derivation, post-deformation updating, and use of BD-Trees. Without loss of generality, we consider sphere trees constructed on polygonal models henceforth, and briefly discuss how these could be extended to bounding boxes. Throughout, we will describe the bounding sphere with radius, R, and center point, c (see Figure 5).

3.1 BD-Tree Construction: A Wrapped Hierarchy

BD-Tree construction involves first constructing a sphere tree on the undeformed model, after which it can be updated following deformation using the approach of §3.2. In the terminology of [Guibas et al. 2002], this undeformed BD-Tree is a *wrapped hierarchy*, wherein spheres enclose (or wrap) associated geometry, and not a *layered hierarchy* in which spheres merely enclose their child spheres (see Figure 3). Layered hierarchies are important in previous work on updating using hierarchical linear-time sphere refitting, e.g., see [Brown et al. 2001]. Unfortunately, layered bounds can fit more loosely than wrapped ones [Guibas et al. 2002]. By updating a wrapped hierarchy, BD-Trees can obtain tighter bounds than the deformable layered case (see Figure 4).

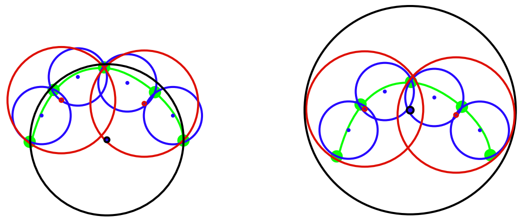


Figure 3: *The wrapped hierarchy* (left) has smaller spheres than the layered hierarchy (right). The base geometry is shown in green, with five vertices. Notice that in a wrapped hierarchy the bounding sphere of a node at one level need not contain the spheres of its descendents and so can be significantly smaller. However, since each sphere contains all the points in the base geometry, it is sufficient for collision detection.

BD-Tree construction has two stages:

1. **Build a hierarchy of the underlying geometry** using any effective technique, e.g., [Quinlan 1994; Hubbard 1995; Bradshaw and O'Sullivan 2004]. In our examples, we use an approach similar to [Quinlan 1994] for binary sphere tree construction, with a single leaf sphere bounding each triangle. Note that even though this method and most others also compute bounding spheres for each node in the hierarchy at the same time, this is not necessary at this stage.

2. **Build bounding spheres for tree nodes.** A simple method, used in our examples, is to start with the loose fitting bounding sphere produced during hierarchy construction, with center point c. It contains a set of polygons with associated vertex points $\{\mathsf{p}_i\}_{i\in\Lambda}$ where Λ is a list of vertex indices. We then reduce the radius of each sphere as much as possible to tightly *wrap the underlying polygons*, leaving the center c unchanged. Specifically, the undeformed sphere's radius, R, is

$$R \equiv \max_{i\in\Lambda} \|\mathsf{p}_i - \mathsf{c}\|_2. \tag{2}$$

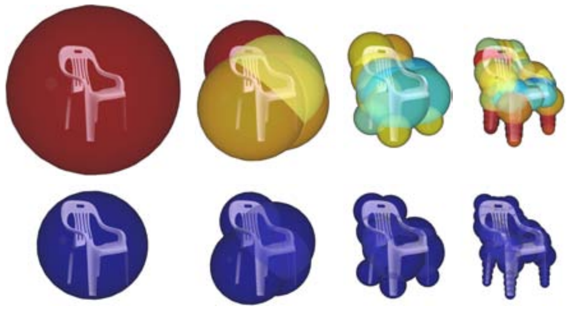


Figure 4: *Comparison of undeformed layered hierarchy (top) versus the wrapped BD-Tree (bottom).* Throughout the paper, dark red denotes at least 3 times greater sphere volume (or $\geq 1.44\times$ radius) *than the undeformed wrapped BD-Tree sphere*, and dark blue implies a negligible size increase.

Alternatively, one could just construct the smallest enclosing sphere for the underlying geometry of each node using, for instance, the efficient randomized algorithm of Welzl [1991].

3.2 Fast BD-Tree Updates

Given points $\{p_i\}_{i\in\Lambda}$ associated with a node in a hierarchy, how can we compute new bounding spheres as the object deforms? We propose to compute each sphere's updated center, c', and an updated conservative radius, R', as functions of the reduced coordinates, q. The update process is illustrated in Figure 5. A key point is that this update can be performed independently for each sphere, and also efficiently for many reduced models.

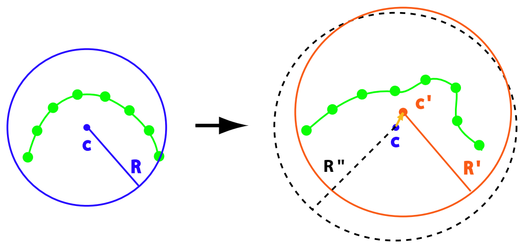


Figure 5: *BD-Tree bounding sphere during deformation* illustrating the change in the center $c \rightarrow c'$ evaluated as a weighted average of the points, $p_i \rightarrow p'_i$, and the conservatively enlarged radius $R \rightarrow R'$ that bounds the effect of each component of the reduced coordinate q using the triangle inequality. The inexpensive bound R'' of §3.5 is also shown.

3.2.1 Sphere Center Update ($c \rightarrow c'$)

Following deformation, the undeformed sphere center c is displaced by a weighted average of the contained points' displacements u, with weights $\{\beta_i\}_{i\in\Lambda}$. Specifically, c', is given by

$$c' = c + \sum_{i\in\Lambda} \beta_i u_i = c + \sum_{i\in\Lambda} \beta_i \left(\sum_{j=1}^{M} U_{ij} q_j \right) \tag{3}$$

$$= c + \sum_{j=1}^{M} \left(\sum_{i\in\Lambda} \beta_i U_{ij} \right) q_j \tag{4}$$

$$\equiv c + \sum_{j=1}^{M} \bar{U}_j q_j = \boxed{c + \bar{U} q \equiv c'} \tag{5}$$

where $\bar{U}_j \in \mathbb{R}^3$ is a β-weighted average of the j^{th} displacement field, U_{*j}, where $\sum_{i\in\Lambda} \beta_i = 1$ and $\beta_i \in [0,1] \forall i \in \Lambda$. In our examples, we track the mean displacement using $\beta_i = 1/|\Lambda|$. Note that the center, c', has linear dependence on the deformation parameters, q, and that computing the center involves only $6M$ flops for M displacement fields.

3.2.2 Sphere Radius Update ($R \rightarrow R'$)

Following deformation, the bounding sphere radius is conservatively enlarged using the triangle inequality as follows:

$$\max_{i\in\Lambda} \|p'_i - c'\|_2 = \max_{i\in\Lambda} \|(p_i - c) + \sum_{j=1}^{M} (U_{ij} - \bar{U}_j) q_j\|_2 \tag{6}$$

$$\leq \max_{i\in\Lambda} \|p_i - c\|_2 + \sum_{j=1}^{M} \left(\max_{i\in\Lambda} \|U_{ij} - \bar{U}_j\|_2 \right) |q_j|$$

$$\equiv R + \sum_{j=1}^{M} \Delta R_j |q_j| = \boxed{R + \Delta R^T q^{ABS} \equiv R'} \tag{7}$$

where ΔR_j is the radius increment due to the j^{th} displacement field. Note that the radius, R', has linear dependence on the absolute value of the deformation parameters, q^{ABS}, and that computing the radius involves only $2M$ flops for M displacement fields.

3.3 Tightness of Conservative Bounding Spheres

A key question regarding BD-Trees is how well the conservative bound fits the deformed model. Certainly, the undeformed BD-Tree's wrapped hierarchy is much better than the usual layered hierarchy used with deformable models. Fortunately, it can be shown theoretically and in practice, that the conservative radius increase $\Delta R^T q^{ABS}$ still provides useful bounds. For example, Figure 6 shows that the deformed BD-Tree needn't be any worse than the hierarchically updated layered hierarchy even for large deformations.

To build intuition about why the conservative bound is effective, notice that for uniform translation of a sphere's points by each mode, $U_{ij} = t_j$, the bounding radius is conveniently invariant (since $U_{ij} = \bar{U}_j \forall i, j$ and so $\Delta R = 0$). This suggests that the BD-Tree will have tighter spheres when the displacement fields are locally constant (even if the displacements are large), and spheres may remain tighter near the leaves of the tree.

More specifically, for Lipschitz continuous displacement fields, we show in Appendix A that $\|\Delta R\|_2$ is bounded by a constant times the sphere's undeformed radius R; furthermore, the proportionality constant in question is monotonically decreasing for child spheres. Consequently, smaller spheres (alt. finer levels) should tend to have proportionately smaller radius increases, and thus the deformed BD-Tree will remain an effective bounding hierarchy. This trend is clearly observed in numerical experiments, e.g., the $\|\Delta R\|_2$ values for the plastic chair (normalized per-level averages) exhibit quick decay: 1.000, 0.684, 0.575, 0.399, 0.252, 0.167, 0.110, 0.080, 0.057, 0.041, 0.030, 0.022, 0.017, 0.016, 0.015, 0.013, 0.007, 0.004, 0.003.

3.4 Summary of BD-Tree Method

To summarize, with the BD-Trees, collision detection and proximity queries with deformable objects are performed as in standard sphere tree methods for rigid objects, with traversals proceeding from coarse to fine sphere nodes. The only differences are:

Precomputation: Build a sphere tree on the undeformed model, and tighten spheres to wrap underlying geometry. At each node in the hierarchy, in addition to the undeformed sphere center c and wrapped radius R, compute and store the matrices $\bar{U}$ and ΔR ($4M$ additional numbers per node).

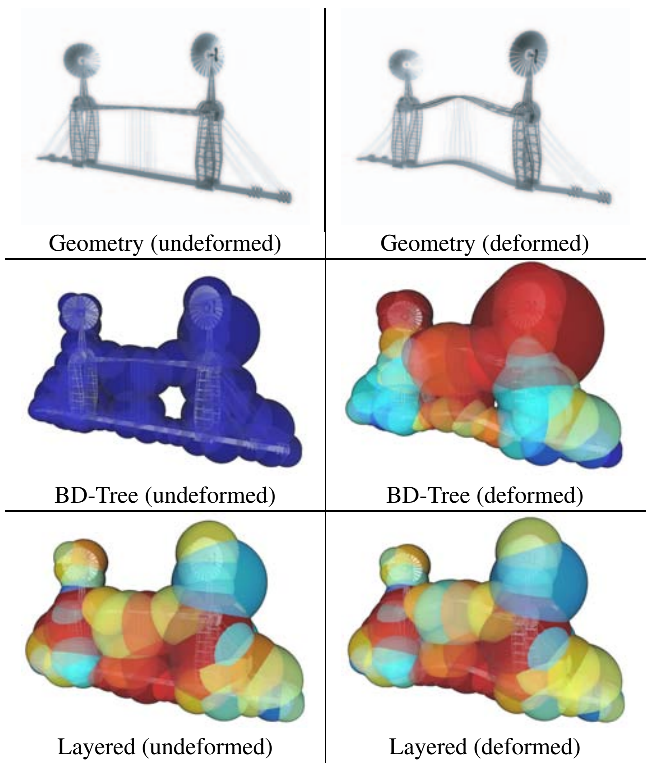


Figure 6: *BD-Tree for large deformations:* The BD-Tree and layered hierarchies are illustrated for the bridge model using 64 spheres on tree level 6. (Left) The undeformed bridge shows relatively larger spheres in the layered hierarchy versus the BD-Tree, with mean radius ratio of 1.31, and maximum of 1.77; (Right) A large deformation of the the nonlinear FEM model shows a layered hierarchy with similar tightness while the BD-Tree has a conservative fit. However, note that the colors indicate the conservative BD-Tree is still comparable in tightness to the layered hierarchy (cf. colors in Figure 4); in fact, both the mean and maximum radius ratios of the layered model (mean=1.31, max=1.79) are *larger* than for the BD-Tree (mean=1.25, max=1.56). Nevertheless, the BD-Tree has the largest (most conservative) sphere. Similar trends occur on other levels.

Run time updates: The first time a node is visited during tree traversal per time-step, update the center $\mathsf{c}' = \mathsf{c} + \bar{\mathsf{U}}\mathsf{q}$ and the radius $R' = R + \Delta R^T \mathsf{q}^{ABS}$. Use the updated sphere (c', R') for all subsequent collision checks at that time-step.

Traversal: A key point is that the bounding spheres can be updated in any order, and the update costs are independent of the number of points in the bounding sphere. In particular, the bounding sphere for a coarse node can be updated before its descendents are, and this can be done in a lazy manner, just prior to testing sphere overlap. Therefore, the asymptotic complexity of collision detection for reduced deformable objects is similar to the rigid case. The cost of each sphere-sphere overlap test is increased by the amortized update cost of $8M$ flops per deformed sphere.

BD-Trees are not optimized for time-critical collision response in the sense of [Hubbard 1995] since the hierarchy is not layered. Nevertheless, we can reduce collision processing costs by using bounded-depth traversals. During precomputation, we designate BD-Tree spheres on some suitably fine level L as contact proxies. Coarser tree nodes are then enlarged slightly to wrap their descendent contact spheres. Although the contact spheres also grow conservatively, for smooth displacement fields they stay wrapped without special precautions, since child spheres tend to enlarge more slowly than their parent spheres (as discussed in §3.3).

3.5 BD-Tree Extensions

Exploiting Temporal Coherence: Although the aforementioned $8M$ flop (center and radius) update is fast enough for our examples, we can find effective bounds *much* faster for small changes in q, say $\mathsf{q}'' = \mathsf{q} + \delta\mathsf{q}$, after obtaining (c', R') as above. We leave the center of the bounding volume at c', and use a simple $\mathscr{O}(1)$ per-sphere approximation to find the enlarged radius R'' as follows:

$$\max_{i\in\Lambda} \|\mathsf{p}_i'' - \mathsf{c}'\|_2 = \max_{i\in\Lambda} \|(\mathsf{p}_i + \sum_j \mathsf{U}_{ij}\mathsf{q}_j'') - (\mathsf{c} + \sum_j \bar{\mathsf{U}}_j \mathsf{q}_j)\|_2 \tag{8}$$

$$= \max_{i\in\Lambda} \|(\mathsf{p}_i - \mathsf{c}) + \sum_j (\mathsf{U}_{ij} - \bar{\mathsf{U}}_j)\mathsf{q}_j + \sum_j \mathsf{U}_{ij}\delta\mathsf{q}_j\|_2 \tag{9}$$

$$\leq R' + \max_{i\in\Lambda} \|\sum_j \mathsf{U}_{ij}\delta\mathsf{q}_j\|_2 \quad \leq \quad R' + (\max_{i\in\Lambda} \|\mathsf{U}_{i:}\|_2)\|\delta\mathsf{q}\|_2 \tag{10}$$

$$= \boxed{R' + \delta R \|\delta\mathsf{q}\|_2 \equiv R''} \tag{11}$$

where in (10) we used the submultiplicative property of the matrix norm, $\|\mathsf{U}_{i:}\mathsf{q}\|_2 \leq \|\mathsf{U}_{i:}\|_2 \|\mathsf{q}\|_2$. By precomputing $\delta R = \max_i \|\mathsf{U}_{i:}\|_2$, where $\mathsf{U}_{i:} \in \mathbb{R}^{3\times M}$, for each bounding volume, and evaluating $\|\delta\mathsf{q}\|_2$ once at runtime, *the amortized radius correction cost for each traversed sphere is a single flop!* Clearly, this is the lowest cost possible for any update that depends on the input.

Updating Bounding Boxes: The same principles of bounded deformation can also be used to augment bounding box hierarchies [Gottschalk et al. 1996; van den Bergen 1997; Larsson and Akenine-Möller 2001]. Similar to sphere BD-Trees, the tree is constructed on the undeformed model (using any suitable technique), and each bounding box retains the same center (or origin) c and associated set of points specified by the index list Λ; the bounds are then tightened (if possible) to closely wrap underlying polygons. Post-deformation updating is similar to the sphere case. The update formula for the deformed center, c', is unchanged from (5) used earlier, so that each bound tracks some average point motion as before. For axis-aligned bounding boxes, the conservative radius bounds update of (7) is simply replaced by a *per-axis bound* by replacing the 3-vector Euclidean norm, $\|\cdot\|_2$ by the absolute value of the x (or y or z) component, $|\cdot|_x \equiv |(\cdot)_x|$ throughout (6–7). Since three axis updates are required, box updates require $12M$ flops total, as opposed to $8M$ flops for spheres. It is possible to derive updates for oriented bounding boxes [Gottschalk et al. 1996] by considering per-axis bounds in rotated coordinates, e.g., $|\cdot|_x \to |\cdot|_{\tilde{x}}$. Although the conservative BD-Tree bound update and fixed orientation may offset the benefit of oriented bounds, they could be useful in (close proximity) situations where the object is only mildly deformed.

4 Results

Despite the approximate bounds used by the BD-Tree, it performs well in practice. We illustrate this behavior using three examples shown here and in our accompanying video; geometric statistics are given in Table 1. All sphere tree implementations are unoptimized Java code, and timings are for a 3.0 GHz Intel Xeon CPU.

Model	#Vertices	#Faces	M	#Spheres	#Levels
Chair	9421	17914	10	35827	19
Bridge	135304	240316	10	480631	26
Fish	12531	24776	3	49551	18
Curtain	6184	12020	12	24039	17

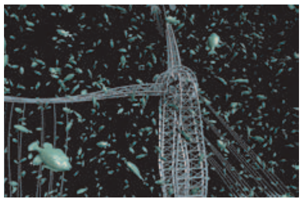
Table 1: *Geometric statistics* for our models.

To study performance on large deformations in a controlled setting, we use an instrumented example of a ball thrown at a compressed geometry animation [Alexa and Müller 2000] of a cloth curtain blowing in the wind. The animation has 110 frames (30 FPS; see accompanying video). We compare BD-Tree performance to hierarchically updated layered sphere trees (see Table 2). Although more sphere overlap tests (#Tests) result from conservative BD-Tree spheres ($<R'/R>=1.51$), the *total collision detection cost* (Total Time) is orders of magnitude smaller than the layered hierarchical update approach. Also note that the exact speedup will depend on mesh complexity, and higher (lower) polygon counts will tend to produce greater (smaller) speedups.

Method	#Tests	Total Time	<Frame Time>
BD-Tree	3890	2.4 ms	0.022 ms
Layered	2150	5400 ms	49 ms

Table 2: *Performance statistics* for curtain-ball example.

Our deformable multibody dynamics examples involve numerous falling plastic chairs, and fish falling on a bridge superstructure. These examples exhibit qualitatively similar performance behaviors. Each bridge and chair BD-Tree sphere ($M = 10$) can be updated in approximately 0.75 microseconds. All deformations are computed using NVIDIA Quadro FX 3000 vertex shader hardware (similar to [James and Pai 2002]), and collision detection and response is performed on the CPU.



In the bridge example, we perform offline computation of collisions between about 12000 linear modal fish models, and a compressed animation of a nonlinear FEM simulation of a large swaying bridge. To avoid the bottleneck of deforming/rendering 300 million fish triangles, we perform *view frustrum culling using the root sphere of each BD-Tree*; off-screen collisions are still resolved in an output-sensitive manner. Root spheres are also used to update a uniform subdivision at each time-step to address any broad phase collision bottleneck [Lin and Gottschalk 1998]. To reduce fish collision processing costs, we use bounded-depth (level 7) traversals (see §3.4) to exploit BD-Tree output-sensitivity (see Figure 7). Forces are computed using a simple Kelvin-Voigt penalty law. For improved visual quality, the bridge's (skinny) leaf-sphere triangles are reconstructed for contact geometry; this triangle-phase collision response (and rendering) dominate simulation costs.

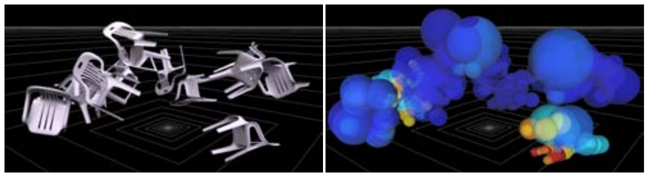


Figure 7: *Output-sensitive deformable collision processing:* (Left) simulation of 12 chairs crashing to the ground; (Right) updated spheres illustrating the extent of sphere tree traversal. Only a small fraction of BD-Tree spheres are updated at any time step. The tree traversal depth can be bounded to lower collision processing costs (shown for maximum depth of 8).

Our final example involves 3601 deformable plastic chairs crashing to the ground (see Figure 1), and serves to illustrate performance of the algorithm for multiple contacts. Each chair is approximated as a linear modal vibration model (with strain limiting), and collisions are resolved at tree level 10. The simulation has 5000 frames (30 FPS), with 5 time-steps used per-frame (25000 time-steps) to resolve high-energy collisions. The mean collision processing cost per time-step is 2.1 seconds, whereas the mean hardware rendering cost (with view frustrum culling) is 2.0 seconds per frame, for a total cost of 14.7 hours for simulation and 2.8 hours for rendering. During the animation 1.6 billion sphere collision events were processed, or approximately 30000 collisions per second of computation. We compared this result to a simulation with rigid chairs ($\mathsf{q} = 0$), and observed comparable performance: mean time-step cost was 1.1 seconds, for total simulation cost of 7.5 hours (or 10.9 hours if BD sphere updates were redundantly performed), 1.1 billion collision events, for a mean collision processing rate of 41000 collisions per second of computation. The deformable contact phenomena are clearly more intensive, having approximately half a billion more collisions. Nevertheless, *the total processing cost of the massive deformable plastic chair simulation is only twice that of the rigid case.* Finally, we did not compare our method to the layered hierarchy; however, the chair's hierarchical layered update takes 55 ms, and suggests that *about 2 months would be required for updates alone*–far exceeding the 14.7 hours required for the entire BD-Tree simulation.

5 Summary and Discussion

We have introduced Bounded Deformation Trees (BD-Trees), and shown that this precomputed collision hierarchy makes it possible to efficiently bound deformations of reduced deformable models. BD-Trees provide effective output-sensitive collision detection that can be much faster than hierarchical updating.

There are several limitations of the approach. First, the simple conservative radius bound means that the radius *always* increases under deformation (cf. [Larsson and Akenine-Möller 2003]), even when the geometry actually shrinks in size, e.g., if an object shrinks to a point, the radius can double. Another point is that the approach is primarily useful for *reduced* deformable models ($M \ll N$), and deformations well-approximated by the reduced Uq displacement model. Using BD-Trees with unreduced models, e.g., $\mathsf{U} = I$, will lead to far too conservative bounds. Some (dense) displacement models can have high-rank (high M) (e.g., [James and Pai 1999] has $M = \mathcal{O}(N)$ in general) and this can lead to computational and memory overhead. For large enough M, there are diminishing returns and updating a layered hierarchy becomes relatively less costly. Finally, the BD-Tree's output-sensitive speedups only help when the query's "output" is small; in the worst case, all or most of the spheres may be queried, e.g., during ray-tracing, and therefore traditional brute-force post-deformation updating methods would have similar effectiveness.

We believe that BD-Trees are a fruitful starting point for future work on deformable collision detection. BD-Trees are but one example of intelligent independently-updateable deformation bounds for output-sensitive collision processing. In addition to augmenting other types of bounds, e.g., bounding boxes, bound types could be mixed, and various top-down/bottom-up updating strategies explored. Tree construction methods could be optimized for deformed BD-Tree tightness. Optimized collision codes often exploit temporal coherence in narrow-phase queries, e.g., using generalized front tracking on bounding volume test trees [Lin and Gottschalk 1998], and this could be explored for BD-Trees. Finally, BD-Trees provide a unique opportunity to explore continuous collision detection strategies for deformable models.

A ΔR for Smooth Displacement Fields

In this appendix, we show that in the important case of smooth displacement fields, the magnitude of the radius growth factors ΔR in the sphere tree can be bounded by a function that decreases with R, and thus maintains a useful bounding hierarchy under deformation.

Consider the important case where $\mathsf{U}_{ij}, i \in \Lambda$ is Lipschitz continuous on the point set $\{\mathsf{p}_i\}_{i\in\Lambda}$, so that

$$\|\mathsf{U}_{ij} - \mathsf{U}_{kj}\|_2 \leq L_j^{\Lambda}\|\mathsf{p}_i - \mathsf{p}_k\|_2, \quad i,k \in \Lambda,$$

where L_j^{Λ} *is the Lipschitz constant*[1], such that for smoother displacement fields one expects smaller L_j^{Λ} values.

From the definition of ΔR_j (with β representing an affine combination) we obtain the bound

$$\begin{aligned} \Delta R_j &= \max_{i\in\Lambda} \|\mathsf{U}_{ij} - \bar{\mathsf{U}}_j\|_2 = \max_{i\in\Lambda} \|\mathsf{U}_{ij} - \sum_{k\in\Lambda} \beta_k \mathsf{U}_{kj}\|_2 \quad (12) \\ &\leq \max_{i,k\in\Lambda} \|\mathsf{U}_{ij} - \mathsf{U}_{kj}\|_2 = \|\mathsf{U}_{Ij} - \mathsf{U}_{Kj}\|_2 \quad (13) \end{aligned}$$

where I and K yield the maximum. Note that, in general, the growth factor $\|\mathsf{U}_{Ij} - \mathsf{U}_{Kj}\|_2$ can be larger than δR. Applying the definition of Lipschitz continuity,

$$\Delta R_j \leq \|\mathsf{U}_{Ij} - \mathsf{U}_{Kj}\|_2 \leq L_j^{\Lambda}\|\mathsf{p}_I - \mathsf{p}_K\|_2 \quad (14)$$

and since points in the undeformed sphere are closer than its diameter, $2R$, we obtain

$$\boxed{\Delta R_j \leq 2L_j^{\Lambda} R}. \quad (15)$$

Note also that the Lipschitz constant L_j^{Λ} is never larger than its parent sphere's constant $L_j^{\Lambda'}$, since $\Lambda \subset \Lambda'$ implies that $L_\Lambda \leq L_{\Lambda'}$. **Consequently, $\|\Delta R\|_2$ is bounded by a constant times the radius R of the undeformed sphere, and the constant is monotonically decreasing for child spheres.**

Acknowledgements: A special thanks to the anonymous reviewers; Chris Twigg for video production; modeling assistance from Chris, Jernej Barbic, and Robert Y. Wang; and NVIDIA for graphics hardware. Supported in part by NSF grants CAREER-0430528, IIS-0308157, EIA-0215887, and ITR-0205671.

[1]For finite point sets, L_j^{Λ} can be taken as $\max_{ik\in\Lambda, i\neq k} \frac{\|\mathsf{U}_{ij}-\mathsf{U}_{kj}\|_2}{\|\mathsf{p}_i-\mathsf{p}_k\|_2}$.

References

ALEXA, M., AND MÜLLER, W. 2000. Representing Animations by Principal Components. *Computer Graphics Forum 19*, 3 (Aug.), 411–418.

BRADSHAW, G., AND O'SULLIVAN, C. 2004. Adaptive Medial-Axis Approximation for Sphere-Tree Construction. *ACM Transactions on Graphics 23*, 1, 1–26.

BRIDSON, R., FEDKIW, R. P., AND ANDERSON, J. 2002. Robust Treatment of Collisions, Contact, and Friction for Cloth Animation. *ACM Transactions on Graphics 21*, 3 (July), 594–603.

BROWN, J., SORKIN, S., BRUYNS, C., LATOMBE, J., MONTGOMERY, K., AND STEPHANIDES, M. 2001. Real-Time Simulation of Deformable Objects: Tools and Application. In *Proceedings of Computer Animation 2001*.

DEBUNNE, G., DESBRUN, M., CANI, M.-P., AND BARR, A. H. 2001. Dynamic Real-Time Deformations Using Space & Time Adaptive Sampling. In *Proc. of ACM SIGGRAPH 2001*, 31–36.

DINGLIANA, J., AND O'SULLIVAN, C. 2000. Graceful Degradation of Collision Handling in Physically Based Animation. *Computer Graphics Forum 19*, 3 (Aug.), 239–248.

FALOUTSOS, P., VAN DE PANNE, M., AND TERZOPOULOS, D. 1997. Dynamic Free-Form Deformations for Animation Synthesis. *IEEE Trans. on Vis. and Comp. Graph. 3*, 3, 201–214.

GANOVELLI, F., DINGLIANA, J., AND O'SULLIVAN, C. 2000. BucketTree: Improving collision detection between deformable objects. In *Spring Conference in Computer Graphics (SCCG2000)*, 156–163.

GOTTSCHALK, S., LIN, M. C., AND MANOCHA, D. 1996. OBB-Tree: A Hierarchical Structure for Rapid Interference Detection. In *Proceedings of ACM SIGGRAPH 96*, 171–180.

GUIBAS, L., NGUYEN, A., RUSSEL, D., AND ZHANG, L. 2002. Collision Detection for Deforming Necklaces. In *Proceedings of the Eighteenth Annual Symposium on Computational Geometry*, ACM Press, 33–42.

HUBBARD, P. M. 1995. Collision Detection for Interactive Graphics Applications. *IEEE Transactions on Visualization and Computer Graphics 1*, 3, 218–230.

JAMES, D. L., AND PAI, D. K. 1999. ARTDEFO: Accurate Real Time Deformable Objects. In *Proceedings of ACM SIGGRAPH 99*, Computer Graphics Proceedings, 65–72.

JAMES, D. L., AND PAI, D. K. 2002. DyRT: Dynamic Response Textures for Real Time Deformation Simulation With Graphics Hardware. *ACM Transactions on Graphics 21*, 3, 582–585.

JIMENEZ, P., THOMAS, F., AND TORRAS, C. 2001. 3D Collision Detection: A Survey. *Computers and Graphics 25*, 2 (Apr.), 269–285.

LARSSON, T., AND AKENINE-MÖLLER, T. 2001. Collision Detection for Continuously Deforming Bodies. In *Eurographics 2001*, A. Chalmers and T.-M. Rhyne, Eds., Eurographics, 325–333.

LARSSON, T., AND AKENINE-MÖLLER, T. 2003. Efficient collision detection for models deformed by morphing. *The Visual Computer 19*, 2, 164–174.

LEWIS, J. P., CORDNER, M., AND FONG, N. 2000. Pose Space Deformations: A Unified Approach to Shape Interpolation and Skeleton-Driven Deformation. In *Proceedings of ACM SIGGRAPH 2000*, 165–172.

LIN, M. C., AND GOTTSCHALK, S. 1998. Collision detection between geometric models: A survey. In *Proc. of IMA Conference on Mathematics of Surfaces*, 37–56.

MANOCHA, D., LIN, M. C., DOGGETT, M., GREENE, N., HOFF, K., KILGARD, M., AND KRISHNAN, S. 2002. Interactive Geometric Computations Using Graphics Hardware. In *SIGGRAPH 2002 Course Notes*, ACM SIGGRAPH.

QUINLAN, S. 1994. Efficient Distance Computation between Non-Convex Objects. In *IEEE Intern. Conf. on Robotics and Automation*, IEEE, 3324–3329.

VAN DEN BERGEN, G. 1997. Efficient Collision Detection of Complex Deformable Models using AABB Trees. *Journal of Graphics Tools 2*, 4, 1–14.

WARREN, J., AND WEIMER, H. 2001. *Subdivision Methods for Geometric Design: A Constructive Approach*, first ed. Morgan Kaufmann Publishers, October.

WELZL, E. 1991. Smallest enclosing disks (balls and ellipsoids). In *New Results and New Trends in Comp. Sci.*, H. Maurer, Ed., vol. 555 of *Lecture Notes Comp. Sci.* Springer-Verlag, 359–370.