Algorithmica
DOI 10.1007/s00453-007-9019-4

Kinetic Collision Detection for Convex Fat Objects

Mohammad Ali Abam · Mark de Berg · Sheung-Hung Poon · Bettina Speckmann

Abstract We design compact and responsive kinetic data structures for detecting collisions between n convex fat objects in 3-dimensional space that can have arbitrary sizes. Our main results are:

(i) If the objects are 3-dimensional balls that roll on a plane, then we can detect collisions with a KDS of size $O(n \log n)$ that can handle events in $O(\log^2 n)$ time. This structure processes $O(n^2)$ events in the worst case, assuming that the objects follow constant-degree algebraic trajectories.
(ii) If the objects are convex fat 3-dimensional objects of constant complexity that are free-flying in $\mathbb{R}^3$, then we can detect collisions with a KDS of $O(n \log^6 n)$ size that can handle events in $O(\log^7 n)$ time. This structure processes $O(n^2)$ events in the worst case, assuming that the objects follow constant-degree algebraic trajectories. If the objects have similar sizes then the size of the KDS becomes $O(n)$ and events can be handled in $O(\log n)$ time.

Keywords Kinetic data structures · Collision detection · Fat objects

M.A. and S.-H.P. were supported by the Netherlands' Organisation for Scientific Research (NWO) under project no. 612.065.307. M.d.B. was supported by the Netherlands' Organisation for Scientific Research (NWO) under project no. 639.023.301.

M.A. Abam · M. de Berg · S.-H. Poon · B. Speckmann (✉)
Department of Mathematics and Computing Science, TU Eindhoven, P.O. Box 513,
5600 MB Eindhoven, The Netherlands
e-mail: speckman@win.tue.nl

M.A. Abam
e-mail: mabam@win.tue.nl

M. de Berg
e-mail: mdberg@win.tue.nl

S.-H. Poon
e-mail: spoon@win.tue.nl

1 Introduction

Collision detection is a basic computational problem arising in all areas of computer science involving objects in motion—motion planning, animated figure articulation, computer-simulated environments, or virtual prototyping, to name a few. Very often the problem of detecting collisions is broken down into two phases: a *broad phase* and a *narrow phase*. The broad phase determines pairs of objects that might possibly collide, frequently using (hierarchies of) bounding volumes to speed up the process. The narrow phase then uses specialized techniques to test each candidate pair, often by tracking closest features of the objects in question, a process that can be sped up significantly by exploiting spatial and temporal coherence. See [21] for a detailed overview of algorithms for such collision and proximity queries.

Algorithms that deal with objects in motion traditionally discretize the time axis and compute or update their structures based on the position of the objects at every time step. But since collisions tend to occur rather irregularly it is nearly impossible to choose the perfect time-step: too large an interval between sampled times will result in missed collisions, too small an interval will result in unnecessary computations (and still there is no guarantee that no collisions are missed). Event-driven methods, on the other hand, compute the event times of significant changes to a system of moving objects, store those in a priority queue sorted by time, and advance the system to the event at the front of the queue. The kinetic-data-structure framework initially introduced by Basch et al. [4] presents a systematic way to design and analyze event-driven data structures for moving objects.

A kinetic data structure (KDS) is designed to maintain or monitor a discrete attribute of a set of moving objects, where each object has a known motion trajectory or *flight plan*. A KDS contains a set of *certificates* that constitutes a proof of the property of interest. These certificates are inserted in a priority queue (*event queue*) based on their time of expiration. The KDS then performs an event-driven simulation of the motion of the objects, updating the structure whenever a certificate fails. A KDS for collision detection finds a set of geometric tests (elementary certificates) that together provide a proof that the input objects are disjoint—see the surveys by Guibas [13, 14] for more details.

Kinetic data structures and their accompanying maintenance algorithms can be evaluated and compared with respect to four desired characteristics. A good KDS is *compact* if it uses little space in addition to the input, *responsive* if the data structure invariants can be restored quickly after the failure of a certificate, *local* if each object involves a small number of certificates, and *efficient* if the worst-case number of events handled by the data structure (for a given class of motions) is not much larger than the worst-case number of combinatorial changes in the maintained attribute (for that class of motions).

Kinetic Data Structures for Collision Detection One of the first papers on kinetic collision detection was published by Basch et al. [5], who designed a KDS for collision detection between two simple polygons in the plane. Their work was extended to an arbitrary number of polygons by Agarwal et al. [1]. Kirkpatrick et al. [20] and Kirkpatrick and Speckmann [19] also described KDS's for kinetic collision detection

between multiple polygons in the plane. These solutions all maintain a decomposition of the free space between the polygons into "easy" pieces (usually pseudo-triangles). Unfortunately it seems quite hard to define a suitable decomposition of the free space for objects in 3D, let alone maintain it while the objects move—the main problem being, that all standard decomposition schemes in 3D can have quadratic complexity. Hence, even though collision detection is the obvious application for kinetic data structures, there has hardly been any work on kinetic collision detection in 3D.

There are only a few papers that deal directly with (specialized versions of) kinetic 3D collision detection. Guibas et al. [15], extending work by Erickson et al. [12] in the plane, show how to certify the separation of two convex polyhedra moving rigidly in 3D using certain outer hierarchies. Basch et al. [3] describe a structure for collision detection among multiple convex *fat* objects that have almost the same size. The structure of Basch et al. uses $O(n \log^2 n)$ storage and processes $O(n^2)$ events and events can be processed in $O(\log^3 n)$ time. Coming and Staadt [11] kinetize the sweep-and-prune approach to find candidate pairs of objects that might collide. Their method has a quadratic worst-case bound and they give only experimental evidence for its performance. If all objects are spheres of similar sizes Kim et al. [17] present an event-driven approach that subdivides space into cells and processes events whenever a sphere enters or leaves a cell. This approach was later extended [18] to accommodate spheres with unknown trajectories but still similar sizes. There is only experimental evidence for the performance of this method. Finally, Guibas et al. [15] use the power diagram of a set of arbitrary balls in 3D to kinetically maintain the closest pair among them. The worst-case complexity of this structure is quadratic and it might undergo more than cubically many changes.

Results The main goal of our paper is to develop KDS's for 3D collision detection that have a *near-linear number of certificates* for *multiple* convex fat objects of *varying sizes*. As discussed above, none of the existing solutions achieves all these goals simultaneously. Our KDS's can be viewed as structures that perform the broad phase of the global collision-detection approach sketched above; one still has to detect collisions between the candidate pairs of objects produced by the KDS. Assuming the objects have constant complexity, this can trivially be done in constant time per pair; how to do this for complex objects is beyond the scope of this paper. Thus the challenge is to get a near-linear number of certificates, so that the number of candidate pairs is reduced from quadratic to near-linear.

We start in Sect. 2 with the special case of n balls of arbitrary sizes rolling on a plane. Here we present an elegant and simple KDS that uses $O(n \log n)$ storage and processes $O(n^2)$ events in the worst case if the objects follow constant-degree algebraic[1] trajectories. Processing an event takes $O(\log^2 n)$ time.

[1] In fact, the bound on the number of events holds in a more general setting: we maintain lists of certain x- and y-coordinates—for instance the coordinates of the tangency points of the disks with the plane on which they roll—whose values change according to the motions of the objects. The number of events is bounded by the number of changes (swaps) in these sorted lists. The $O(n^2)$ bound thus holds if we assume that any pair of coordinates swaps $O(1)$ times (which is for example the case if the motions are constant-degree algebraic). A similar remark holds for the other KDS's that we develop.

In Sect. 3 we turn our attention to free-flying convex fat objects. Note that we do not assume the objects to be polyhedral. We first study fat objects that have similar sizes. We give an almost trivial KDS that has $O(n)$ size and processes $O(n^2)$ events; handling an event takes $O(\log n)$ time. This improves both the storage and the event-handling time of the KDS of Basch et al. [3] by several logarithmic factors. Next we consider the much more difficult general case, where the fat objects can have vastly different sizes. Here we present a KDS that uses $O(n \log^6 n)$ storage and processes $O(n^2)$ events; handling an event takes $O(\log^7 n)$ time. This is the first collision-detection KDS for multiple objects in 3D that has a near-linear number of certificates and does not require the objects to have similar sizes. Even though our KDS for this case uses $O(n \log^6 n)$ storage, it maintains only a linear number of candidate pairs of objects to test for collisions; the additional storage is used in various supporting data structures. Our structure is based on the following idea: we put a number of points—we call them guards—around each object in such a way that if two objects collide, one must contain a guard from the other. Because the objects are fat, we can show that a constant number of guards per object suffices. The idea of reducing problems on fat objects to problems on suitably chosen points has been used before—see e.g. [6, 10]. In our context, however, it is far from straightforward to apply since detecting collisions between objects and guards is nearly as difficult as detecting collisions between the objects themselves. Nevertheless, using several additional ideas, we show how to make this approach work.

2 Balls Rolling on a Plane

Assume that we are given a set $\mathcal{B}$ of n 3-dimensional balls which are rolling on a 2-dimensional plane T, that is, the balls in $\mathcal{B}$ move continuously while remaining tangent to T—see Fig. 1. In this section we describe a responsive and compact KDS that detects collisions between the balls in $\mathcal{B}$.

The basic idea behind our KDS is to construct a *collision tree* recursively as follows:

- If $|\mathcal{B}| = 1$, then there are obviously no collisions and the collision tree is just a single leaf.
- If $|\mathcal{B}| > 1$, then we partition $\mathcal{B}$ into two subsets, $\mathcal{B}_S$ and $\mathcal{B}_L$. The subset $\mathcal{B}_S$ contains the $\lfloor n/2 \rfloor$ smallest balls and the subset $\mathcal{B}_L$ contains the $\lceil n/2 \rceil$ largest balls from $\mathcal{B}$, where ties are broken arbitrarily. The collision tree now consists of a root node that has an associated structure to detect collisions between any ball from $\mathcal{B}_S$ and any ball from $\mathcal{B}_L$, and two subtrees that are collision trees for the sets $\mathcal{B}_S$ and $\mathcal{B}_L$, respectively.

To detect all collisions between the balls in $\mathcal{B}$ it suffices to detect collisions between the two subsets maintained at every node of the collision tree. Let $\mathcal{B}_S$ and $\mathcal{B}_L$ denote the two subsets maintained at a particular node. The remainder of this section focusses on detecting collisions between the balls in $\mathcal{B}_S$ and those in $\mathcal{B}_L$. In particular, we describe a KDS of size $O(|\mathcal{B}_S| + |\mathcal{B}_L|)$ that can handle events in $O(\log n)$ time—see Lemma 5. The structure processes $O((|\mathcal{B}_S| + |\mathcal{B}_L|)^2)$ events in the worst case,

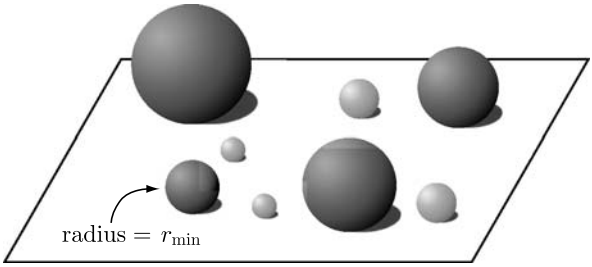

Fig. 1 Balls rolling on a plane—balls in $\mathcal{B}_S$ are *light gray*, balls in $\mathcal{B}_L$ are *dark gray*

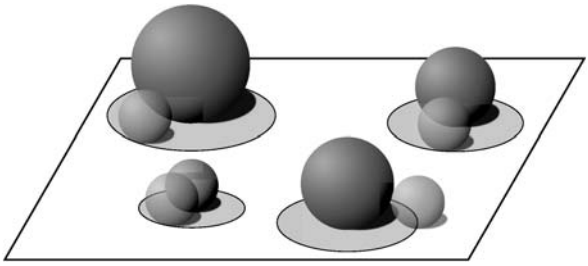

Fig. 2 The radius $r_{\min}$ of the smallest ball in $\mathcal{B}_L$ defines the threshold disks

assuming that the balls follow constant-degree algebraic trajectories. Since the same event can occur simultaneously at $O(\log n)$ nodes of the collision tree, we obtain the following theorem:

Theorem 1 *For any set $\mathcal{B}$ of n 3-dimensional balls that roll on a plane, there is a KDS for collision detection that uses $O(n \log n)$ space and processes $O(n^2)$ events in the worst case, assuming that the balls follow constant-degree algebraic trajectories. Each event can be handled in $O(\log^2 n)$ time.*

2.1 Detecting Collisions between Small and Large Balls

As mentioned above, we can restrict ourselves to detecting collisions between balls from two disjoint sets $\mathcal{B}_S$ and $\mathcal{B}_L$ where the balls in $\mathcal{B}_L$ are at least as large as the balls in $\mathcal{B}_S$. Recall that all balls are rolling on a plane T. Our basic strategy is the following: we associate a region D_i on T with each $B_i \in \mathcal{B}_L$ such that if the point of tangency of a ball $B_j \in \mathcal{B}_S$ and T is not contained in D_i, then B_j cannot collide with B_i. The regions associated with the balls in $\mathcal{B}_L$ need to have two important properties: (i) each point in T is contained in a constant number of regions and (ii) we can efficiently detect whenever a region starts or stops to contain a tangency point when the balls in $\mathcal{B}_L$ and $\mathcal{B}_S$ move. We first deal with the first requirement, that is, we consider $\mathcal{B}_L$ to be static. For a ball B_i let r_i denote its radius and let t_i be the point of tangency of B_i and T.

The Threshold Disk We define the distance of a point q in the plane T to a ball B_i as follows. Imagine that we place a ball $B(q)$ of initial radius 0 at point q. We then inflate $B(q)$ while keeping it tangent to T at q, until it collides with B_i. We define the distance of q and B_i, which we denote by $\mathrm{dist}(q, B_i)$, to be the radius of $B(q)$.

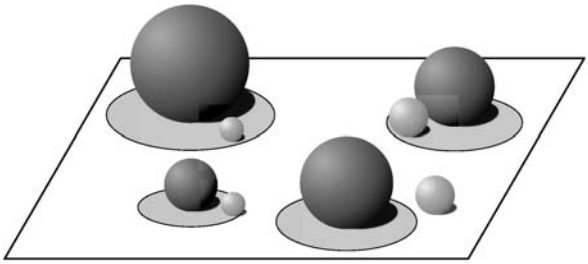

Fig. 3 Detecting collisions with the threshold disks

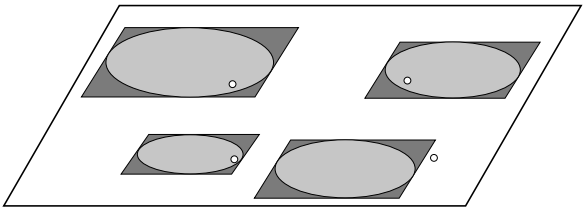

Fig. 4 Replacing threshold disks with threshold boxes

More precisely, $\text{dist}(q, B_i)$ is the radius of the unique ball that is tangent to T at q and tangent to B_i. It is easy to show that $\text{dist}(q, B_i) = d(q, t_i)^2/4r_i$ where $d(q, t_i)$ denotes the Euclidean distance between q and t_i.

Since we have to detect collisions only with balls from $\mathcal{B}_S$ and the balls in $\mathcal{B}_L$ are at least as large as those in $\mathcal{B}_S$, we can stop inflating when $B(q)$ is as large as the smallest ball in $\mathcal{B}_L$. Based on this, we define the threshold disk D_i of a ball $B_i \in \mathcal{B}_L$ as follows: a point $q \in T$ belongs to D_i if and only if $\text{dist}(q, B_i) \leq r_{\min}$ where $r_{\min}$ is the radius of the smallest ball in $\mathcal{B}_L$—see Fig. 2. Because $\text{dist}(q, B_i) = d(q, t_i)^2/4r_i$, D_i is a disk whose radius is $2\sqrt{r_i \cdot r_{\min}}$ and whose center is t_i.

Clearly a ball $B_j \in \mathcal{B}_S$ cannot collide with a ball $B_i \in \mathcal{B}_L$ as long as t_j is outside D_i—see Fig. 3. In the following, we prove that a point $q \in T$ can be contained in at most a constant number of threshold disks. We start by proving a more general result, which we will need later when we replace the threshold disks by threshold boxes. For a given constant $c \geq 0$, let $c \cdot D_i$ denote the disk with radius $c \cdot \text{radius}(D_i)$ and center t_i.

Lemma 2 *The number of threshold disks D_j that are at least as large as a given threshold disk D_i and for which $c \cdot D_i \cap c \cdot D_j \neq \emptyset$, is at most* $(8c^2 + 2c + 1)^2 + 1$.

Proof Let $\mathcal{D}(i)$ be the set of all threshold disks D_j that are at least as large as D_i and for which $c \cdot D_i \cap c \cdot D_j \neq \emptyset$. First we prove that there are no two balls B_j and B_k such that $r_k \geq r_j > 16c^2 r_i$ and $D_j, D_k \in \mathcal{D}(i)$. Assume, for contradiction, that there are two balls B_j and B_k with this property. Since B_j and B_k are disjoint, we have $d(t_j, t_k) \geq ((r_i + r_j)^2 - (r_i - r_j)^2)^{1/2} = 2\sqrt{r_j \cdot r_k} > 8c\sqrt{r_k \cdot r_i}$. We also know that $d(t_j, t_k) \leq d(t_j, t_i) + d(t_i, t_k) \leq 8c\sqrt{r_k \cdot r_i}$ which is a contradiction. Hence, there is at most one ball B_j such that $r_j > 16c^2 r_i$ and $D_j \in \mathcal{D}(i)$.

It remains to show that the number of balls B_j whose radii are not greater than $16c^2 r_i$ and for which $D_j \in \mathcal{D}(i)$ is at most $(8c^2 + 2c + 1)^2$. Let B_j be one of these

balls and let x be a point in $c \cdot D_j \cap c \cdot D_i$. Since

$$d(t_i, t_j) \leq d(t_i, x) + d(t_j, x) \leq 2c\sqrt{r_i \cdot r_{\min}} + 2c\sqrt{r_j \cdot r_{\min}} \leq (2c + 8c^2)r_i,$$

t_j must lie in a disk whose center is t_i and whose radius is $(2c+8c^2)r_i$. We also know that $d(t_j, t_k) \geq 2\sqrt{r_j \cdot r_k} \geq 2r_i$ for any two such balls B_j and B_k. Thus the set $\mathcal{D}'(i)$ of disks centered at t_j with radius r_i for all $D_j \in \mathcal{D}(i)$ are disjoint. Note that any disk in $\mathcal{D}'(i)$ lies inside the disk centered at t_i with radius $((2c + 8c^2) + 1)r_i$. Thus $|\mathcal{D}'(i)| \leq (\pi(2c + 8c^2 + 1)^2 r_i^2)/(\pi r_i^2) = (2c + 8c^2 + 1)^2$ which implies $|\mathcal{D}(i)| \leq (2c + 8c^2 + 1)^2 + 1$. □

Lemma 3 *Each point $q \in T$ is contained in at most a constant number of threshold disks.*

Proof Let D_i be the smallest threshold disk containing q. Lemma 2 with $c = 1$ implies that the number of disks not smaller than D_i and intersecting D_i is constant. Hence the number of threshold disks containing q is constant. □

The Threshold Box The threshold disks have the important property that each point in T is contained in a constant number of disks. But unfortunately, as the balls in $\mathcal{B}_L$ and $\mathcal{B}_S$ move, it is difficult to detect efficiently whenever a tangency point enters or leaves a threshold disk. Hence we replace each threshold disk by its axis-aligned bounding box—see Fig. 4. The bounding box of a threshold disk D_i associated with a $B_i \in \mathcal{B}_L$ is called a *threshold box* and is denoted by $\mathrm{TB}(B_i)$. The following lemma states that the threshold boxes retain the crucial property of the threshold disks, namely, that each point $q \in T$ is contained in at most a constant number of threshold boxes. It follows fairly easily from Lemma 2.

Lemma 4 *Each point $q \in T$ is contained in at most a constant number of threshold boxes.*

Proof Instead of considering the threshold boxes directly, we consider the disks defined by the circumcircles $D(\mathrm{TB}(B_j))$ of each threshold box $\mathrm{TB}(B_j)$ with $B_j \in \mathcal{B}_L$. We have $\mathrm{radius}(D(\mathrm{TB}(B_j))) = \sqrt{2} \cdot \mathrm{radius}(D_j)$ for all $B_j \in \mathcal{B}_L$. Let $\mathrm{TB}(B_i)$ be the smallest box containing q. Lemma 2 with $c = \sqrt{2}$ implies that the number of circumcircle disks that are at least as large as $D(\mathrm{TB}(B_i))$ and that intersect $D(\mathrm{TB}(B_i))$ is constant. Hence the number of threshold boxes that are not smaller than $\mathrm{TB}(B_i)$ and intersect $\mathrm{TB}(B_i)$ is constant, and so is the number of threshold boxes containing q. □

Kinetic Maintenance Recall that to detect collisions between $\mathcal{B}_S$ and $\mathcal{B}_L$, for each ball $B_j \in \mathcal{B}_S$ we determine which threshold boxes of balls in $\mathcal{B}_L$ contain the tangency point t_j. Note that according to Lemma 4, t_j is contained in a constant number of threshold boxes. For each $B_j \in \mathcal{B}_S$ we maintain the set of threshold boxes that contain t_j and certificates that guarantee disjointness of B_j and the balls from $\mathcal{B}_L$ whose threshold boxes contain t_j.

To maintain our structure we only need to detect when a tangency point t_j enters or leaves a threshold box. To do so, we maintain two sorted lists: one storing the x-coordinates of the tangency points of $\mathcal{B}_S$ and the minimum and maximum x-coordinates of the threshold boxes associated with the balls in $\mathcal{B}_L$, the other storing the y-coordinates of the tangency points of $\mathcal{B}_S$ and the minimum and maximum y-coordinates of the threshold boxes. If the objects follow constant-degree algebraic trajectories, the number of events processed by our structure—that is, the number of swaps in these sorted lists—is quadratic in the size of $\mathcal{B}_S$ and $\mathcal{B}_L$. Moreover, each such event can be processed in $O(\log n)$ time: $O(1)$ time to swap the points, and $O(\log n)$ time to update the event queue.

Lemma 5 *Let $\mathcal{B}_S$ and $\mathcal{B}_L$ be two disjoint sets of balls that roll on a plane where the balls in $\mathcal{B}_L$ are at least as large as the balls in $\mathcal{B}_S$. There is a KDS for collision detection between the balls of $\mathcal{B}_S$ and those of $\mathcal{B}_L$ that uses $O(|\mathcal{B}_S|+|\mathcal{B}_L|)$ space, and that processes $O((|\mathcal{B}_S|+|\mathcal{B}_L|)^2)$ events if the balls follow constant-degree algebraic trajectories. Each event can be handled in $O(\log n)$ time.*

Remark Recall that we have a collision-detection KDS as in Lemma 5 for every node of a collision tree, as described at the beginning of this section. Each such collision-detection KDS generates events, but we do not maintain these events in separate event queues. Instead we maintain a global event queue and we insert the failure time of each certificate into the global event queue. It is easy to see that at any time there are $O(n \log n)$ certificates in the global event queue. Hence, the asymptotic complexity of inserting (deleting) certificates into (from) the global event queue remains $O(\log n)$.

Remark In the above KDS, we maintain two sorted lists for every node of the collision tree. Thus an event may happen in $O(\log n)$ nodes on a path from the root to a leaf simultaneously. This forces the KDS to insert (delete) $O(\log n)$ certificates into (from) the event queue which takes $O(\log^2 n)$ time. Another possibility is to maintain two global sorted lists based on x- and y-coordinates, instead of having two sorted lists for each node. This way an event can create or delete a constant number of certificates, which implies the response time is $O(\log n)$. Since every ball is associated with $O(\log n)$ threshold boxes, the size of the global sorted list is $O(n \log n)$, which means the number of events is $O(n^2 \log^2 n)$. Hence, the decrease in response time comes at the cost of an increase in the number of events to be processed.

3 Free-Flying Fat Objects in 3-Space

We now turn our attention to collision detection for a set $\mathcal{K}$ of n free-flying objects in 3-space. We will show how to obtain a compact and responsive KDS when $\mathcal{K}$ consists of convex, constant-complexity fat objects. Note that we do not require the objects to be polyhedral.

We will use the following definition of fatness [16]. An object K is called *ρ-fat*, for some $\rho \geq 1$, if there are two concentric balls $B^-(K)$ and $B^+(K)$ such that $B^-(K) \subset K \subset B^+(K)$ and

$$\text{radius}(B^+(K))/\,\text{radius}(B^-(K)) \leq \rho.$$

Since we are dealing with convex objects, this definition is equivalent up to constant factors to other definitions of fatness that have been used [9]. We call radius$(B^-(K))$ and radius$(B^+(K))$ the *inner radius* and *outer radius* of K, respectively, and we call the common center of $B^-(K)$ and $B^+(K)$ the *center* of K. We say that an object K is *larger* than another object K' if the inner radius of K is larger than the inner radius of K'.

Unfortunately the approach of the previous section does not work for free-flying objects, not even if we are dealing with balls. The problem is that the radius of the threshold ball of a ball B_i will now be $r_i + r_{\min}$ instead of $2\sqrt{r_i \cdot r_{\min}}$ and this invalidates the proof of Lemma 2 for $c > 1$ and thus invalidates Lemma 4.

3.1 Similarly Sized Objects

We first consider the case where the objects have similar sizes. More precisely, let σ be the *scale factor* of the scene, that is, the ratio between the sizes of the largest and the smallest inner ball:

$$\sigma = \frac{\max_{K \in \mathcal{K}} \text{radius}(B^-(K))}{\min_{K \in \mathcal{K}} \text{radius}(B^-(K))}.$$

It follows from the results of Zhou and Suri [23] that the number of pairs of intersecting bounding boxes of the objects in $\mathcal{K}$ is at most $O(\rho\sqrt{\rho^3\sigma^3}n) = O(\rho^2\sigma\sqrt{\rho\sigma}n)$. (A similar but slightly weaker result also follows directly from results in Van der Stappen's thesis [22].) Hence, if σ is a constant, we can simply maintain the set of pairs of intersecting bounding boxes, and for each such pair add a certificate to test for disjointness of the corresponding objects.

To maintain the pairs of intersecting bounding boxes, we maintain three sorted lists: one on the minimum and maximum x-coordinates of the boxes, one on the minimum and maximum y-coordinates of the boxes, and one on the minimum and maximum z-coordinates of the boxes. Whenever there is a swap in one of these lists, two boxes may intersect or become apart. If two boxes intersect, we add a certificate for the corresponding objects. If they become apart, we remove the corresponding certificate. This leads to the following theorem.

Theorem 6 *For any set $\mathcal{K}$ of n convex, constant-complexity ρ-fat objects with scale factor σ, there is a KDS for collision detection that uses $O(\rho^2\sigma\sqrt{\rho\sigma}n)$ storage and processes $O(n^2)$ events in the worst case, assuming that the objects follow constant-degree algebraic trajectories. Each event can be handled in $O(\log n)$ time.*

3.2 Arbitrarily Sized Objects

When the sizes of the objects vary greatly, then there can be a quadratic number of intersecting bounding boxes even when the objects are fat. Hence, a more sophisticated approach is needed. Our global strategy for this case is as follows. We place a number of so-called *guarding points*—or *guards*, for short—around each object $K \in \mathcal{K}$. The guards for K are defined in a local reference frame for K, so they follow the motion of K. More precisely, they follow the motion of a fixed reference point of K.

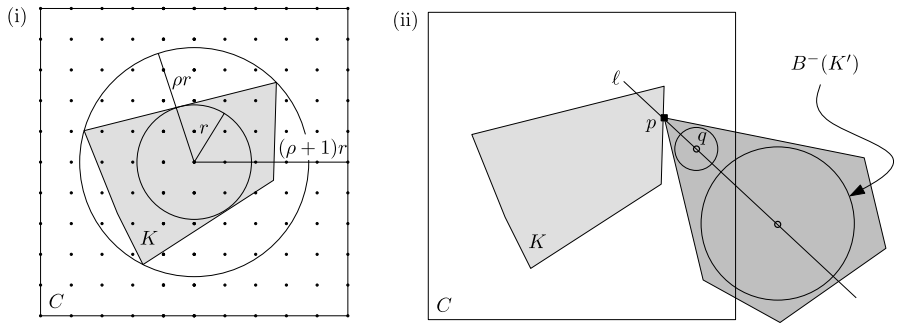

Fig. 5 Illustrations for the proof of Lemma 7

We choose the guards in such a way that when two objects collide, the larger object must contain at least one guard from the smaller object. This reduces the collision-detection problem to maintaining for each guard which object contains it. The next lemma states that we can always find a small guarding set because the objects are fat.

Lemma 7 *For any ρ-fat object K, there is a set $G(K)$ of $O(\rho^6)$ guarding points such that any ρ-fat object K' that collides with K and is at least as large as K contains a point from $G(K)$.*

Proof Let $r := \text{radius}(B^-(K))$. Let C be the cube whose center coincides with the center of K, and whose side length is $2(\rho+1)r$. Draw a regular grid in C whose cells have side length $2r/(\sqrt{3}(\rho+1))$—see Fig. 5(i) for a (2-dimensional) illustration. The grid points together form the set $G(K)$. Clearly $|G(K)| = O(\rho^6)$. It remains to argue that $G(K)$ is a guarding set.

Let K' be an object colliding with K and at least as large as K, and let p be a point where K and K' touch. Because K' is at least as large as K, the ball $B^-(K') \subset K'$ has radius r' at least r. Consider the line ℓ through p and $\text{center}(B^-(K'))$. Let d be the distance between p and $\text{center}(B^-(K'))$, and let q be the point in between p and $\text{center}(B^-(K'))$ at distance $\frac{r/(\rho+1)}{r'} \cdot d$ from p. Finally, let $B(q)$ be the ball centered at q and with radius $r/(\rho+1)$—see Fig. 5(ii). Observe that $B(q)$ can be obtained by scaling $B^-(K')$ with respect to p by a factor of $\frac{r/(\rho+1)}{r'}$. Since K' is convex, $p \in K'$, and $B^-(K') \subset K'$, this implies $B(q) \subset K'$. We claim that $B(q) \subset C$. Since $B(q)$ has radius $r/(\rho+1)$, this means it must contain at least one point of $G(K)$, which will prove the lemma.

It remains to prove the claim that $B(q) \subset C$. To this end note that d is at most $\rho \cdot r'$, because K' is ρ-fat. This implies that the distance of p to any point in $B(q)$ is at most

$$\frac{r/(\rho+1)}{r'} \cdot d + \frac{r}{\rho+1} \leq r.$$

On the other hand, the distance of p to the boundary of C is at least r. Hence, $B(q) \subset C$, as claimed. □

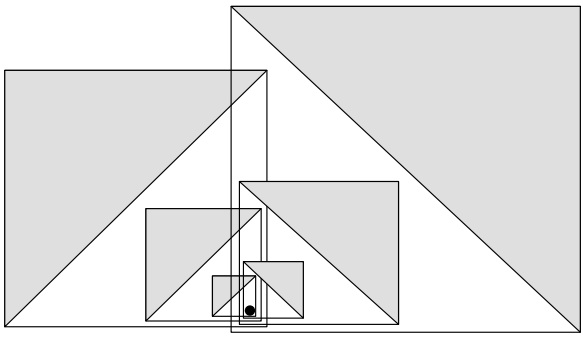

Fig. 6 A guard can be contained in many bounding boxes

Our KDS for collision detection thus works as follows. For each object $K \in \mathcal{K}$ we compute a set $G(K)$ of guards according to Lemma 7. Our goal is now to maintain for each $g \in G(K)$ the object $K(g)$ containing g (if such an object exists). Let $\text{Cand}(K) := \{K(g) : g \in G(K)\}$; the set $\text{Cand}(K)$ contains the candidates with which we check for collisions. More precisely, for each object $K(g) \in \text{Cand}(K)$, our KDS has a certificate testing for the disjointness of K and $K(g)$.

Unfortunately, it seems difficult to maintain the set $\text{Cand}(K)$ directly. This would require us to detect when an object K' starts to contain a guard g, which is difficult to do efficiently. Hence, we replace the objects by their bounding boxes. Because the bounding boxes are axis-aligned, it will be easier to check whether any of them starts (or stops) to contain a guard of some other object. This introduces a new problem, however; a guard can be contained in many bounding boxes—see Fig. 6. Clearly, we cannot afford to maintain for each guard g all the bounding boxes that contain it. Next we describe how to deal with this problem.

Consider a guard g. As noted earlier, there can be many disjoint objects whose bounding boxes contain g. When this happens, however, the objects must become larger and larger, as shown in Fig. 6, with the larger objects being "behind" the smaller ones. Thus the objects that are closest to g in a some direction are the candidates for containing g. Hence, the idea is to maintain for g not all objects whose bounding boxes contain g, but only the closest objects around g.

To make this idea work, we first partition the space around g into cones, as follows. Let U be the unit cube, centered at the origin. Draw a grid on each face of U, such that the grid cells have edge length $1/(2\sqrt{6}\rho)$. Triangulate each grid cell. We have now partitioned the surface of U into $O(\rho^2)$ triangles. Each triangle defines, together with the origin, an (infinite) cone γ by taking the union of all rays emanating from the origin and passing through the triangle. Since the grid cells have edge length $1/(2\sqrt{6}\rho)$ their diagonals have length $1/(4\sqrt{3}\rho)$, which implies the following.

Lemma 8 *Let ℓ_1 and ℓ_2 be two rays originating from the apex of a cone γ and being inside the cone. Then the angle between ℓ_1 and ℓ_2 is at most* $\arctan(1/(2\sqrt{3}\rho))$.

The set of cones for a guard g is obtained by translating these cones such that their apices—the origin in the construction—coincide with g. We denote this set by $\Gamma(g)$.

Note that the motion of each cone is purely translational: even when an object K rotates, its guards just follow the path of the reference point of K and so the cones for

that guard only translate. This means that any cone will always be a translated copy of one of the "standard" cones defined for U. From now on, whenever we speak of a cone we refer to a cone constructed for a guard, as described above.

Since it seems to be difficult to efficiently maintain the closest object to g, the apex of a cone γ, we maintain the object whose center's orthogonal projection onto a specific side of γ is the closest one to g. More precisely, for each cone we defined for U we choose one of its edges as its *representative edge*. This also gives us a representative edge for each cone constructed for any guard g. From now on, whenever we are discussing a cone γ with apex g and we are talking about the object closest to g, we refer to the object whose center's orthogonal projection onto γ's representative edge is closest to g.

The next lemma implies that we can indeed restrict our attention to the closest object to g among those objects whose bounding boxes contain g. For an object K, let $\mathrm{bb}(K)$ denote its (axis-aligned) bounding box.

Lemma 9 *Let $\mathcal{K}(\gamma)$ be the set of all objects K whose center lies in a cone γ and such that $\mathrm{bb}(K)$ contains the apex g of the cone. Suppose that one of these objects, $K(g)$, contains g. Then $K(g)$ must be the closest object to g in $\mathcal{K}(\gamma)$. Moreover, suppose the objects in $\mathcal{K}(\gamma)$ move in such a way that their centers remain inside γ. Then the order of the orthogonal projections of their centers onto the representative edge of γ remains unchanged.*

Proof Let r and c be the inner radius and the center of an object $K \in \mathcal{K}(\gamma)$, respectively. Consider a plane passing through c and being orthogonal to the representative edge of γ. The intersection of γ and this plane is a triangle xyz—see Fig. 7. We claim (and will prove later) that $d(c,x), d(c,y), d(c,z) \leq r$. Because K is convex, this implies that the triangle xyz is inside the object K. Hence, the objects whose centers are inside the cone cannot exchange order during the motion as long as their centers remain inside the cone.

Now let $K = K(g)$. Then the tetrahedron $gxyz$ is inside $K(g)$ which means the centers of the other objects must lie outside $gxyz$. This implies $K(g)$ is the closest object to g.

It remains to prove the claim that $d(c,x), d(c,y), d(c,z) \leq r$. Assume (the extension of) gy is the representative edge of γ. Since $\mathrm{bb}(K)$ contains g and $\angle gyc$ is a

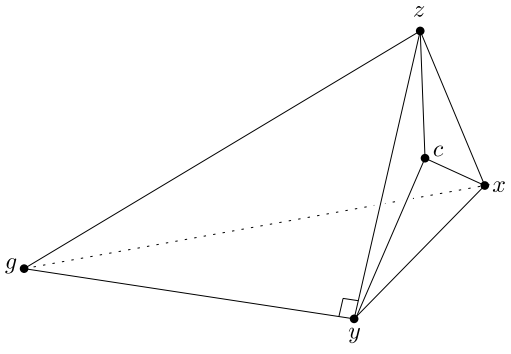

Fig. 7 The intersection of cone γ and the plane orthogonal to the representative edge of γ

right angle, we have

$$d(g, y) \leq d(g, c) \leq \sqrt{3}\rho r.$$

Moreover, we have

$$\begin{aligned} d(c, y) &= d(g, y) \cdot \tan(\angle cgy) \leq d(g, y) \cdot \tan(\arctan(1/(2\sqrt{3}\rho))) \\ &\leq (\sqrt{3}\rho r) \cdot (1/(2\sqrt{3}\rho) \leq r/2. \end{aligned}$$

Similarly, $d(z, y) \leq r/2$ and $d(x, y) \leq r/2$. It follows that

$$d(c, z) \leq d(c, y) + d(y, z) \leq r$$

and

$$d(c, x) \leq d(c, y) + d(y, x) \leq r.$$

□

To summarize, our KDS works as follows. For each object $K \in \mathcal{K}$ we compute a set $G(K)$ of guards according to Lemma 7. For each guard g we construct a collection $\Gamma(g)$ of infinite cones with apex g. For each cone $\gamma \in \Gamma(g)$ we maintain the closest object whose center is inside γ and whose bounding box contains g, and we have a certificate testing for disjointness for this object with the object for which g is a guard. Next we describe a KDS that maintains all this information efficiently.

3.2.1 Details of the KDS

Let $G(\mathcal{K}) := \{G(K) : K \in \mathcal{K}\}$ denote the set of all guards over all objects, let $\Gamma(\mathcal{K}) := \{\Gamma(g) : g \in G(\mathcal{K})\}$ denote the collection of all cones, and let $\text{bb}(\mathcal{K})$ denote the set of bounding boxes of the objects in $\mathcal{K}$.

Detecting Events We wish to maintain for each $\gamma \in \Gamma(g)$ the closest object $\mathcal{K}^*(\gamma)$ to g whose center is inside γ and whose bounding box contains g. By Lemma 9 this object can change only when one of the following two events happens:

Box event: a bounding box starts or stops to contain a guard.
Center event: a center moves into or out of a cone.

To detect box events, we maintain three sorted lists. The first list is sorted on x-coordinate and contains the guards in $G(\mathcal{K})$ as well as the bounding boxes, where each bounding box occurs twice (according to its maximum and minimum x-coordinates). We have similar lists sorted on y- and z-coordinates.

To detect center events, we observe that each cone is a translate of one of the $O(\rho^2)$ cones defined for the unit cube. The cones are generated by six triangulated grids, one on each facet of the unit cube, so the facets of the cones have only $O(\rho)$ distinct orientations. Hence, we can detect center events using $O(\rho)$ sorted lists. Each sorted list corresponds to a possible orientation of a cone facet, and stores the object centers and the cones that have a facet in the given orientation. More precisely, instead of storing the cones themselves, we store the planes containing the facets of the cones. Notice that a plane bounds up to $O(\rho)$ cones—see Fig. 8.

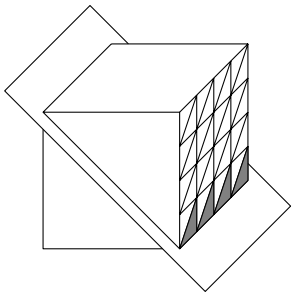

Fig. 8 The cones corresponding to the shaded triangles all share a common plane defining one of their facets

Lemma 10 *The box and center events can be detected with a KDS that uses $O(\rho^7 n)$ storage and that processes $O(\rho^{13} n^2)$ events in total, assuming the objects follow constant-degree algebraic trajectories. At each event processed by this KDS, we spend $O(\log \rho)$ time to test whether the event corresponds to an actual box or center event.*

Proof Recall that we have $O(\rho^6)$ guards per object, and $O(\rho^2)$ cones per guard.

For the box events we have three sorted lists, each storing $O(n)$ boxes and $O(\rho^6 n)$ guards. Hence, their total size is $O(\rho^6 n)$ and the total number of events is $O(\rho^{12} n^2)$. Whenever we have a swap in one of these lists, we just check in $O(1)$ time whether it corresponds to a guard entering or leaving a box.

For the center events we have $O(\rho)$ sorted lists. For each guard, the cones are defined by triangulated grids on the facets of a unit cube centered at the guard. This grid is induced by $O(\rho)$ lines on the facets. Hence, as remarked earlier, the $O(\rho^2)$ cones are generated by $O(\rho)$ planes—one plane for each grid line and one for each diagonal line inducing the triangulation. Since we have $O(\rho^6)$ guards per object, we have in total $O(\rho^7)$ planes per object. Each guard contributes one plane to each list. Hence, we have $O(\rho)$ lists, each containing $O(\rho^6 n)$ planes. This means these lists together use $O(\rho^7 n)$ storage and have $O(\rho^{13} n^2)$ events. Whenever we have an event in one of these lists we check whether it corresponds to a center crossing a plane. If so, we must find out which of the $O(\rho)$ cones bounded by that plane, if any, are involved. Note that there can be two: the center could enter one cone and leave another cone. Finding out the cones involved can easily be done in $O(\log \rho)$ time. □

Handling Events When we have detected a center event, we may have to update the object $\mathcal{K}^*(\gamma)$ of at most two cones. Next we describe how to handle the event involving an object K and some cone γ defined for a guard g.

When $\mathrm{bb}(K)$ starts to contain g, or when the center of K moves into γ, things are easy: If $\mathcal{K}^*(\gamma)$ does not yet exist, K becomes the closest object to g and so we set $\mathcal{K}^*(\gamma) := K$; otherwise, we check whether K is closer to g than the current $\mathcal{K}^*(\gamma)$ and, if so, we set $\mathcal{K}^*(\gamma) := K$.

Handling the case where $\mathrm{bb}(K)$ stops to contain g, or when the center of K moves out of γ, is more difficult. For this we need a supporting data structure that can answer the following query:

Given a cone γ with apex g, report the closest object to g whose center is in γ and whose bounding box contains g.

Recall that the set of cones can be partitioned into $O(\rho^2)$ subsets, where the cones in each subset are translates of some "standard" cone. We construct a data structure for each subset separately. Because the facets of the cones in a subset have only three distinct orientations, we can find all centers inside a query cone in with a three-level range tree. Finding the bounding boxes containing the apex of the query cone can be done with a three-level segment tree, and filtering out the closest object requires a sorted list on the orthogonal projections of the object centers onto the representative edge of the cone. Hence, our total data structure will be have seven levels. Answering a query can be done in $O(\log^6 n)$ time—the query time is not $O(\log^7 n)$ because in the last level we only need to report the closest object—and the amount of storage is $O(n \log^6 n)$. To kinetize the structure, we use the kinetic variants of range trees [3] and segment trees [8] and sorted lists (which are trivial to maintain). The number of events processed to maintain our seven-level structure is $O(n^2)$ and each event can be handled in $O(\log^7 n)$ time.

Lemma 11 *When a center or box event occurs, we can update the closest object $\mathcal{K}^*(\gamma)$ in $O(\log^6 n)$ time, using a supporting KDS that uses $O(\rho^2 n \log^6 n)$ storage. The supporting KDS processes $O(\rho^2 n^2)$ events in the worst case, assuming the objects follow constant-degree algebraic trajectories, and the response time is $O(\log^7 n)$.*

This leads to our main result.

Theorem 12 *For any set $\mathcal{K}$ of n convex, constant-complexity ρ-fat objects, there is a KDS for collision detection that uses $O(\rho^2 n \log^6 n + \rho^7 n)$ storage and that processes $O(\rho^{13} n^2)$ events in the worst case, assuming the objects follow constant-degree algebraic trajectories. Each event can be handled in $O(\log \rho + \log^7 n)$ time.*

Our KDS is compact and responsive, but unfortunately it is not local: a large object K with many small objects around can be involved in many certificates, because it may contain guards for each of the small objects. However, we can show that the locality of our KDS depends on the ratio of the size of the biggest object and the smallest object in $\mathcal{K}$.

Theorem 13 *Each object in the KDS of Theorem 12 is involved in $O(\rho^8 + \rho^3 \sigma^3)$ certificates, where σ is the ratio of the largest inner radius to the smallest inner radius of the objects in $\mathcal{K}$.*

Proof Consider an object K. There are two kinds of certificates in which K is involved:

Order certificates: these certificates arise from the sorted lists which we maintain.
Collision certificates: these certificates certify disjointness for all candidate pairs that include K.

Since the number of sorted lists in the KDS is $O(\rho)$ and there are $O(\rho^6)$ guards for each object, the number of order certificates involving K is $O(\rho^7)$. It remains to count the number of collision certificates. The number of such certificates involving K and a larger object K' is $O(\rho^8)$, because the guarding set of K has $O(\rho^6)$ size and for each guarding point we maintain $O(\rho^2)$ objects—one per cone defined for the guard. Now we have to count the number of objects K' smaller than K such that $\mathrm{bb}(K)$ contains at least one guard of K'. Since the volume of K is less than $O(\rho^3\sigma^3)$ times the volume of K', a simple packing argument shows that the number of such objects K' is $O(\rho^3\sigma^3)$—note that if $\mathrm{bb}(K)$ contain more than one guarding point of K', we just maintain one collision certificate between K and K'. Therefore, the total number of certificates involving K is $O(\rho^8 + \rho^3\sigma^3)$. □

4 Conclusion

We presented the first KDS's for collision detection between multiple convex fat 3D objects that use a near-linear number of certificates and do not require the objects to have similar sizes. We believe that this is an important step forward in the theoretical investigation of KDS's for 3D collision detection. Our KDS for balls rolling on a plane is simple, and may perform well in practice. Our general KDS for free-flying objects of varying sizes, however, is complicated and the dependency on the fatness parameter ρ is large. Thus our result should be seen as a proof that good bounds are possible in theory—whether a simple and practical solution exists that achieves similar worst-case bounds is still open.

As remarked above, our structures are not local: a single object can be involved in a linear number of certificates. Unfortunately, this seems very hard (if not impossible) to avoid if there is a single large object that is closely surrounded by many tiny objects. Thus we do not expect to see a local KDS that can deal with arbitrarily sized objects. (We have shown though that a local KDS is possible for convex fat objects when their sizes are similar.)

Finally, a challenging open problem is to obtain results on non-convex and/or non-fat objects.

Acknowledgements The last author would like to thank David Kirkpatrick for valuable discussions on the presented subject. We also thank an anonymous referee for a suggestion that simplified the KDS described in Sect. 3.

References

1. Agarwal, P.K., Basch, J., Guibas, L.J., Hershberger, J., Zhang, L.: Deformable free space tilings for kinetic collision detection. Int. J. Robot. Res. **21**, 179–197 (2002)
2. Aurenhammer, F., Edelsbrunner, H.: An optimal algorithm for constructing the weighted Voronoi diagram in the plane. Pattern Recognit. **17**(2), 251–257 (1984)
3. Basch, J., Guibas, L., Zhang, L.: Proximity problems on moving points. In: Proc. 13th Symposium on Computational Geometry, pp. 344–351 (1997)
4. Basch, J., Guibas, L., Hershberger, J.: Data structures for mobile data. J. Algorithms **31**, 1–28 (1999)
5. Basch, J., Erickson, J., Guibas, L.J., Hershberger, J., Zhang, L.: Kinetic collision detection for two simple polygons. Comput. Geom.: Theory Appl. **27**(3), 211–235 (2004)

6. de Berg, M.: Linear size binary space partitions for uncluttered scenes. Algorithmica **28**, 353–366 (2000)
7. de Berg, M., van Kreveld, M., Overmars, M., Schwarzkopf, O.: Computational Geometry: Algorithms and Applications, 2nd edn. Springer, Berlin (2000)
8. de Berg, M., Comba, J., Guibas, L.: A segment-tree based kinetic BSP. In: Proc. 17th Symposium on Computational Geometry, pp. 134–140 (2001)
9. de Berg, M., Katz, M., van der Stappen, F., Vleugels, J.: Realistic input models for geometric algorithms. Algorithmica **34**, 81–97 (2002)
10. de Berg, M., David, H., Katz, M., Overmars, M., van der Stappen, F., Vleugels, J.: Guarding scenes against invasive hypercubes. Comput. Geom.: Theory Appl. **26**, 99–117 (2003)
11. Coming, D., Staadt, O.: Kinetic sweep and prune for collision detection. In: Proc. Workshop on Virtual Reality Interactions and Physical Simulations, pp. 81–90 (2005)
12. Erickson, J., Guibas, L., Stolfi, J., Zhang, L.: Separation-sensitive collision detection for convex objects. In: Proc. 10th ACM-SIAM Symposium on Discrete Algorithms, pp. 327–336 (1999)
13. Guibas, L.: Kinetic data structures: a state of the art report. In: Proc. 3rd Workshop on Algorithmic Foundations of Robotics, pp. 191–209 (1998)
14. Guibas, L.: Modeling motion. In: Goodman, J., O'Rourke, J. (eds.) Handbook of Discrete and Computational Geometry, 2nd edn., pp. 1117–1134. CRC Press (2004)
15. Guibas, L., Xie, F., Zhang, L.: Kinetic collision detection: algorithms and experiments. In: Proc. International Conference on Robotics and Automation, pp. 2903–2910 (2001)
16. Katz, M.: 3-D vertical ray shooting and 2-D point enclosure, range searching, and arc shooting amidst convex fat objects. Comput. Geom.: Theory Appl. **8**, 299–316 (1998)
17. Kim, D., Guibas, L., Shin, S.Y.: Fast collision detection among multiple moving spheres. IEEE Trans. Vis. Comput. Graph. **4**(3), 230–242 (1998)
18. Kim, H.K., Guibas, L., Shin, S.Y.: Efficient collision detection among moving spheres with unknown trajectories. Algorithmica **43**, 195–210 (2005)
19. Kirkpatrick, D., Speckmann, B.: Kinetic maintenance of context-sensitive hierarchical representations for disjoint simple polygons. In: Proc. 18th ACM Symposium on Computational Geometry, pp. 179–188 (2002)
20. Kirkpatrick, D., Snoeyink, J., Speckmann, B.: Kinetic collision detection for simple polygons. Int. J. Comput. Geom. Appl. **12**(1&2), 3–27 (2002)
21. Lin, M., Manocha, D.: Collision and proximity queries. In: Goodman, J., O'Rourke, J. (eds.) Handbook of Discrete and Computational Geometry, 2nd edn., pp. 787–807. CRC Press (2004)
22. van der Stappen, A.F.: Motion planning amidst fat obstacles. Ph.D. thesis, Utrecht University, Utrecht, the Netherlands (1994)
23. Zhou, Y., Suri, S.: Analysis of a bounding box heuristic for object intersection. J. ACM **46**(6), 833–857 (1999)