53 MODELING MOTION

Leonidas J. Guibas and Marcel Roeloffzen

INTRODUCTION

Motion is ubiquitous in the physical world, yet its study is much less developed than that of another common physical modality, namely shape. While we have several standardized mathematical shape descriptions, and even entire disciplines devoted to that area—such as *Computer-Aided Geometric Design* (CAGD)—the state of formal motion descriptions is still in flux. This in part because motion descriptions span many levels of detail; they also tend to be intimately coupled to an underlying physical process generating the motion (dynamics). Thus, there is a wide variety of work on algorithm descriptions of motion and their associated complexity measure. This chapter aims to show how an algorithmic study of motion is intimately tied to discrete and computational geometry. We first discuss some earlier work and various motion models. In Section 53.1 we then go into more detail on models that consider so-called *incremental motion*. We then devote the bulk of this chapter to discussing the framework of *Kinetic Data Structures* (Section 53.2) [Gui98, BGH99].

MOTION IN COMPUTATIONAL GEOMETRY

Dynamic computational geometry refers to the study of combinatorial changes in a geometric structure, as its defining objects undergo prescribed motions. For example, we may have n points moving linearly with constant velocities in $\mathbb{R}^2$, and may want to know the time intervals during which a particular point appears on their convex hull, the steady-state form of the hull (after all changes have occurred), or get an upper bound on how many times the convex hull changes during this motion. Such problems were introduced and studied in [Ata85].

A number of other authors have dealt with geometric problems arising from motion, such as collision detection (Chapter 39) or minimum separation determination [GJS96, ST95b, ST95a]. For instance, [ST95a] shows how to check in subquadratic time whether two collections of simple geometric objects (spheres, triangles) collide with each other under specified polynomial motions.

MOTION MODELS

An issue in the above research is that object motion(s) are assumed to be known in advance, sometimes in explicit form. A common assumption is that the coordinates of each point are bounded-degree polynomial functions of time and that these functions are known in advance. In essence, the proposed methods reduce questions about moving objects to other questions about derived static objects.

While most evolving physical systems follow known physical laws, it is also frequently the case that discrete events occur (such as collisions) that alter the

Preliminary version (August 10, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

motion law of one or more of the objects. Thus motion may be predictable in the short term, but becomes less so further into the future. Because of such discrete events, algorithms for modeling motion must be able to adapt in a dynamic way to motion model modifications. Furthermore, the occurrence of these events must be either predicted or detected, incurring further computational costs. Nevertheless, any truly useful model of motion must accommodate this *on-line* aspect of the temporal dimension, differentiating it from spatial dimensions, where all information is typically given at once.

Here, we distinguish two general motion models. The first considers unknown movement, where we don't have direct knowledge of motions. Instead, we rely on external updates or explicitly requesting new locations of the objects. However, most of the algorithms in this setting, as discussed in Section 53.1, make some assumptions on the motions, such as a bounded movement speed.

The second model is more similar to that of previous work and assumes that trajectories are known at least on the short-term. More precisely, the model assumes that point trajectories are known exactly and any changes in this trajectory are explicitly reported. This model is used by data structures in the KDS-framework discussed in Section 53.1. In most cases the motions are assumed to be polynomial (i.e., coordinates defined by bounded degree polynomials) or pseudo-algebraic, though this is mainly needed to prove efficiency, not correctness of the data structures. (The definition of pseudo-algebraic motion is given in Section 53.2 as it depends on the data-structure, however, polynomial movement is generally also pseudo-algebraic.)

53.1 INCREMENTAL MOTION

In real-world settings, the motion of objects may be imperfectly known and better information may only be obtainable at considerable expense. There have been several studies that incorporate this concept into their motion models. Most of these models are based on updating motions or locations of objects at discrete times. These updates may be obtained either by *pushing* or *pulling*, where in the former case an outside source or component provides new information and in the latter case the data structure must query for new location information when needed.

One way to deal with this more unpredictable motion is to separate the continuous tracking of motion from the more discrete maintenance of the data structure. Several works consider an observer component whose task is to observe motion and provide updated location or motion predictions to a builder component. The builder is then responsible for maintaining the data structure itself. The observer always has access to the exact location of the objects and the goal is to minimize communication between the two components. Communication can happen through *pushing* or *pulling*.

In case of pushing the observer is tasked with ensuring that the builder always has a sufficiently accurate location [YZ12]. Efficiency can then be measured in terms of competitive ratio, that is, by comparing the number of updates sent by the observer with those by an optimal update schedule that knows the full motions, but must adhere to the same accuracy bounds.

Information can also be pulled from the observer by the builder. In this case the builder queries the observer for the location of specific objects when more precise

Preliminary version (August 10, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

information is needed to ensure a correct data structure [Kah91]. This is convenient when there are strict bounds on the movement rates or directions of objects that enable to builder to compute its own guarantees on the objects locations (e.g., an object must be within a certain region). A compromise between the pulling and pushing mechanism can be found by allowing the builder to provide triggers to the observer [MNP+04, CMP09]. A trigger is a simple assumption on the location of the objects, for example that it remains within a certain region or follow a given trajectory. When such an assumption is no longer valid the observer can then inform the builder, which can then update its internal structures and provide new triggers. Triggers are similar to *certificates* that form the basis of the kinetic data structures discussed in Section 53.2.

In practice, the observer and builder may not be part of the same system and communication is not instantaneous. That is, we cannot neglect the fact that time passes and objects move while a location update is send from the observer to the builder. This can be modeled by maintaining regions in which each object resides, as time passes these regions grow. A location update then leads to one region shrinking to a point while every other region grows. When considering objects with a bounded movement rate, but unknown motions, the basic problem of minimizing the number of regions that overlap—that is, the maximum number of regions containing the same point—is hard, even when we aim to minimize ply only at a given time in the future. However, a strategy that approximates this number exists under certain conditions [EKLS16].

A much simpler motion model is that of *stepwise incremental motion*, where objects cannot be continuously observed. Instead, their locations are updated at discrete points in time [BRS12a, BRS12b, BRS13]. At each point the data structure must be repaired. When considering basic constraints on movement and distribution of the objects, this allows for updating strategies that are faster than rebuilding the whole structure from scratch with each location update.

53.2 KINETIC DATA STRUCTURES

Suppose we are interested in tracking high-level attributes of a geometric system of objects in motion such as, for example, the convex hull of a set on n points moving in $\mathbb{R}^2$. Note that as the points move continuously, their convex hull will be a continuously evolving convex polygon. At certain discrete moments, however, the combinatorial structure of the convex hull will change (that is, the circular sequence of a subset of the points that appear on the hull will change). In between such moments, tracking the hull is straightforward: its geometry is determined by the positions of the sequence of points forming the hull. How can we know when the combinatorial structure of the hull changes? The idea is that we can focus on certain elementary geometric relations among the n points, a set of cached assertions we call *certificates*, which altogether certify the correctness of the current combinatorial structure of the hull. Furthermore, we can hope to choose these relations in such a way so that when one of them fails because of point motion, both the hull and its set of certifying relations can be updated locally and incrementally, so that the whole process can continue.

Preliminary version (August 10, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

GLOSSARY

Kinetic data structure: A kinetic data structure (KDS) for a geometric attribute is a collection of simple geometric relations that certifies the combinatorial structure of the attribute, as well as a set of rules for repairing the attribute and its certifying relations when one relation fails.

Certificate: A certificate is an elementary geometric relation used in a KDS.

Event: An event is the failure of a KDS certificate during motion. Events are classified as *external* when the combinatorial structure of the attribute changes, and *internal*, when the structure of the attribute remains the same, but its certification needs to change.

Event queue: In a KDS, all certificates are placed in an event queue, according to their earliest failure time.

Pseudo-algebraic motion: The class of allowed motions is usually specified as the class of pseudo-algebraic motions, in which each KDS certificate can flip between true and false at most a bounded number of times. Note that this class is specific for each KDS, but points whose coordinates are bounded-degree polynomials of time generally satisfy this condition.

The inner loop of a KDS consists of repeated certificate failures and certification repairs, as depicted in Figure 53.2.1.

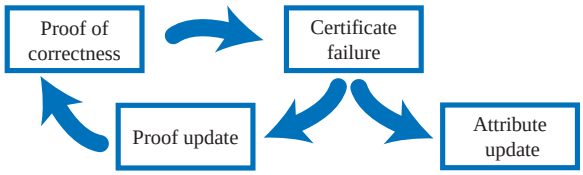


FIGURE 53.2.1
The inner loop of a kinetic data structure.

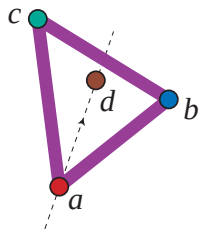
We remark that in the KDS framework, objects are allowed to change their motions at will, with appropriate notification to the data structure. When this happens all certificates involving the object whose motion has changed must re-evaluate their failure times.

CONVEX HULL EXAMPLE

Suppose we have four points a, b, c, and d in $\mathbb{R}^2$, and wish to track their convex hull. For the convex hull problem, the most important geometric relation is the CCW predicate: $\text{CCW}(a, b, c)$ asserts that the triangle abc is oriented counterclockwise. Figure 53.2.2 shows a configuration of four points and four CCW relations that hold among them. It turns out that these four relations are sufficient to prove that the convex hull of the four points is the triangle abc. Indeed the points can move

Preliminary version (August 10, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

and form different configurations, but as long as the four certificates shown remain valid, the convex hull must be *abc*.



Proof of correctness:

- $\text{CCW}(a, b, c)$
- $\text{CCW}(d, b, c)$
- $\text{CCW}(d, c, a)$
- $\text{CCW}(d, a, b)$

FIGURE 53.2.2
Determining the convex hull of the points.

Now suppose that points a, b, and c are stationary and only point d is moving, as shown in Figure 53.2.3. At some time t_1 the certificate $\text{CCW}(d, b, c)$ will fail, and at a later time t_2 $\text{CCW}(d, a, b)$ will also fail. Note that the certificate $\text{CCW}(d, c, a)$ will never fail in the configuration shown even though d is moving. So the certificates $\text{CCW}(d, b, c)$ and $\text{CCW}(d, a, b)$ schedule events that go into the event queue. At time t_1, $\text{CCW}(d, b, c)$ ceases to be true and its negation, $\text{CCW}(c, b, d)$, becomes true. In this simple case the three old certificates, plus the new certificate $\text{CCW}(c, b, d)$, allow us to conclude that the convex hull has now changed to *abdc*.

Old proof	New proof
$\text{CCW}(a, b, c)$	$\text{CCW}(a, b, c)$
$\text{CCW}(d, b, c)$	$\text{CCW}(c, b, d)$
$\text{CCW}(d, c, a)$	$\text{CCW}(d, c, a)$
$\text{CCW}(d, a, b)$	$\text{CCW}(d, a, b)$

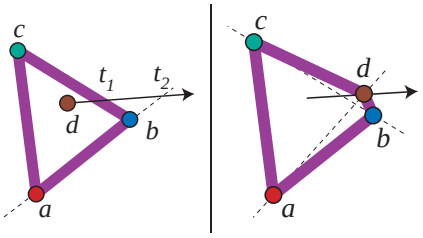


FIGURE 53.2.3
Updating the convex hull of the points.

If the certificate set is chosen judiciously, the KDS repair can be a local, incremental process—a small number of certificates may leave the cache, a small number may be added, and the new attribute certification will be closely related to the old one. A good KDS exploits the continuity or coherence of motion and change in the world to maintain certifications that themselves change only incrementally and locally as assertions in the cache fail.

Preliminary version (August 10, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

PERFORMANCE MEASURES FOR KDS

Because a KDS is not intended to facilitate a terminating computation but rather an on-going process, we need to use somewhat different measures to assess its complexity. In classical data structures there is usually a tradeoff between operations that interrogate a set of data and operations that update the data. We commonly seek a compromise by building indices that make queries fast, but such that updates to the set of indexed data are not that costly as well. Similarly in the KDS setting, we must at the same time have access to information that facilitates or trivializes the computation of the attribute of interest, yet we want information that is relatively stable and not so costly to maintain. Thus, in the same way that classical data structures need to balance the efficiency of access to the data with the ease of its update, kinetic data structures must tread a delicate path between "knowing too little" and "knowing too much" about the world. A good KDS will select a certificate set that is at once economical and stable, but also allows a quick repair of itself and the attribute computation when one of its certificates fails. To measure these concerns, four quality criteria are introduced below. To measure efficiency, we need to define a class of motions that the KDS accepts. Generally this is the class of *pseudo-algebraic* motions. That is, the class of all motions where each certificate can flip between true and false at most a bounded number of times. Although this definition depends on the data-structure, the class of polynomial motions (motions defined by bounded degree polynomial functions) are also pseudo-algebraic in mostly all KDSs.

GLOSSARY

Responsiveness: A KDS is *responsive* if the cost, when a certificate fails, of repairing the certificate set and updating the attribute computation is small. By "small" we mean polylogarithmic in the problem size—in general we consider small quantities that are polylogarithmic or $O(n^\epsilon)$ in the problem size.

Efficiency: A KDS is *efficient* if the number of certificate failures (total number of events) it needs to process is comparable to the number of required changes in the combinatorial attribute description (external events), over some class of allowed motions. Technically, we require that the ratio of total events to external events is small.

Compactness: A KDS is *compact* if the size of the certificate set it needs is close to linear in the degrees of freedom of the moving system.

Locality: A KDS is *local* if no object participates in too many certificates; this condition makes it easier to re-estimate certificate failure times when an object changes its motion law. (The existence of local KDSs is an intriguing theoretical question for several geometric attribute functions.)

CONVEX HULL, REVISITED

We now briefly describe a KDS for maintaining the convex hull of n points moving around in the plane [BGH99].

The key goal in designing a KDS is to produce a ***repairable certification*** of

Preliminary version (August 10, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

the geometric object we want to track. In the convex hull case it turns out that it is a bit more intuitive to look at the dual problem, that of maintaining the upper (and lower) envelope of a set of moving lines in the plane, instead of the convex hull of the primal points. For simplicity we focus only on the upper envelope part from now on; the lower envelope case is entirely symmetric. Using a standard divide-and-conquer approach, we partition our lines into two groups of size roughly $n/2$ each, and assume that recursive invocations of the algorithm maintain the upper envelopes of these groups. For convenience, call the groups red and blue.

In order to produce the upper envelope of all the lines, we have to merge the upper envelopes of the red and blue groups and also certify this merge, so we can detect when it ceases to be valid as the lines move; see Figure 53.2.4.

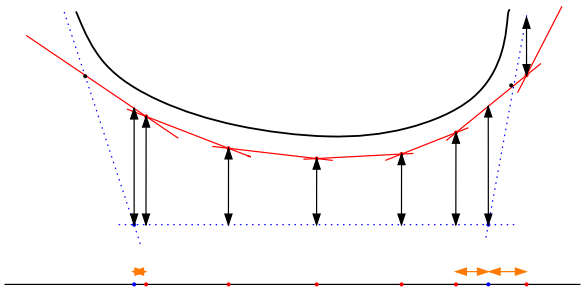


FIGURE 53.2.4
Merging the red and blue upper envelopes. In this example, the red envelope (solid line) is above the blue (dotted line), except at the extreme left and right areas. Vertical double-ended arrows represent y-certificates and horizontal double-ended arrows represent x-certificates, as described below.

Conceptually, we can approach this problem by sweeping the envelopes with a vertical line from left to right. We advance to the next red (blue) vertex and determine if it is above or below the corresponding blue (red) edge, and so on. In this process we determine when red is above blue or vice versa, as well as when the two envelopes cross. By stitching together all the upper pieces, whether red or blue, we get a representation of the upper envelope of all the lines.

The certificates used in certifying the above merge are of three flavors:

- x-certificates ($<_x$) are used to certify x-ordering among the red and blue vertices; these involve four original lines.
- y-certificates ($<_y$) are used to certify that a vertex is above or below an edge of the opposite color; these involve three original lines and are exactly the duals of the CCW certificates discussed earlier.
- s-certificates ($<_s$) are slope comparisons between pairs of original lines; though these did not arise in our sweep description above, they are needed to make the KDS local [BGH99].

Figure 53.2.5 shows examples of how these types of certificates can be used to specify x-ordering constraints and to establish intersection or nonintersection of the envelopes. A total of $O(n)$ such certificates suffice to verify the correctness of the upper envelope merge.

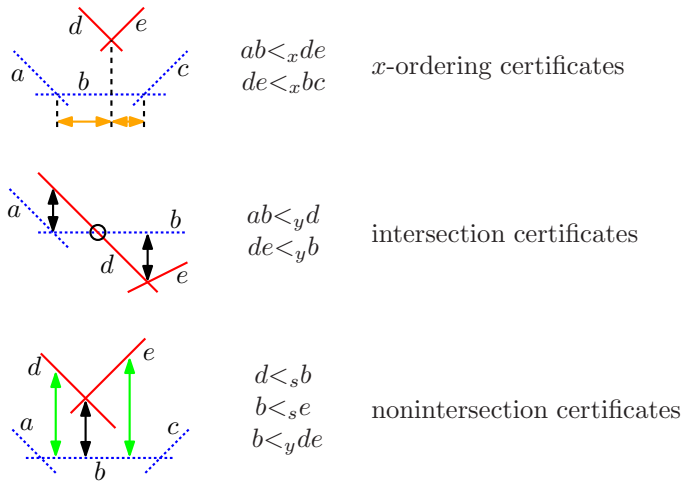


FIGURE 53.2.5
Using the different types of certificates to certify the red-blue envelope merge.

Whenever the motion of the lines causes one of these certificates to fail, a local, constant-time process suffices to update the envelope and repair the certification. Figure 53.2.6 shows an example where a y-certificate fails, allowing the blue envelope to poke up above the red.

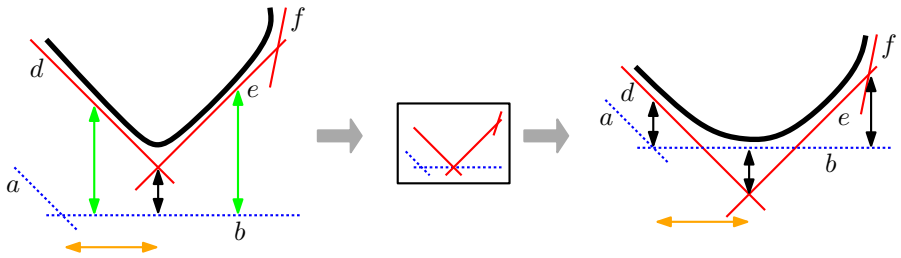


FIGURE 53.2.6
Envelope repair after a certificate failure. In the event shown, lines b, d, and e become concurrent, producing a red-blue envelope intersection.

It is straightforward to prove that this kinetic upper envelope algorithm is responsive, local, and compact, using the logarithmic depth of the hierarchical structure of the certification. In order to bound the number of events processed, however, we must assume that the line motions are polynomial or at least pseudo-algebraic. A proof of efficiency can be developed by extruding the moving lines into space-time surfaces. Using certain well-known theorems about the complexity of upper envelopes of surfaces [Sha94] and the overlays of such envelopes [ASS96] (cf. Chapter 50), it can be shown that in the worst case the number of events processed by this algorithm is near quadratic ($O(n^{2+\epsilon})$, for any constant $\epsilon > 0$, where the constant of proportionality depends on ϵ). Since the convex hull of even linearly moving points can change $\Omega(n^2)$ times [AGHV01], the efficiency result follows. No comparable structure is known for the convex hull of points in dimensions $d \geq 3$.

Preliminary version (August 10, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

EXTENT PROBLEMS

A number of problems for which kinetic data structures were developed are aimed at different measures of how "spread out" a moving set of points in $\mathbb{R}^2$ is—one example is the convex hull, whose maintenance was discussed in the previous subsection. Other measures of interest include the diameter, width, and smallest area or perimeter bounding rectangle for a moving set S of n points. All these problems can be solved using the kinetic convex hull algorithm above; the efficiency of the algorithms is $O(n^{2+\epsilon})$, for any $\epsilon > 0$. There are also corresponding $\Omega(n^2)$ lower bounds for the number of combinatorial changes in these measures. Surprisingly, the best upper bound known for maintaining the smallest enclosing disk containing S is still near-cubic. Extensions of these results to dimensions higher than two are also lacking.

These costs can be dramatically reduced if we consider approximate extent measures. If we are content with $(1 + \epsilon)$-approximations to the measures, then an approximate smallest axis-aligned rectangle, diameter, and smallest enclosing disk can be maintained with a number of events that is a function ϵ only and not of n [AHP01]. For example, the bound of the number of approximate diameter updates in $\mathbb{R}^2$ under linear motion of the points is $O(1/\epsilon)$.

PROXIMITY PROBLEMS

The fundamental proximity structures in computational geometry are the Voronoi diagram and the Delaunay triangulation (Chapter 27). The edges of the Delaunay triangulation contain the closest pair of points, the closest neighbor to each point, as well as a wealth of other proximity information among the points. From the kinetic point of view, these are nice structures, because they admit completely local certifications. Delaunay's 1934 theorem [Del34] states that if a local empty sphere condition is valid for each $(d-1)$-simplex in a triangulation of points in $\mathbb{R}^d$, then that triangulation must be Delaunay. This makes it simple to maintain a Delaunay triangulation under point motion: an update is necessary only when one of these empty sphere conditions fails. Furthermore, whenever that happens, a local retiling of space (of which the classic "edge-flip" in $\mathbb{R}^2$ is a special case; cf. Section 29.3) easily restores Delaunayhood. Thus the KDS for Delaunay (and Voronoi) that follows from this theorem is both responsive and efficient—in fact, each KDS event is an external event in which the structure changes. Tight bounds on the number of events were a long-standing open problem with only a quadratic lower bound and near-cubic upper bound being known [AGMR98]. Here, near-cubic (or near-quadratic) indicates at most a factor n^ϵ difference with cubic (or quadratic) for an arbitrarily small constant $\epsilon > 0$. Progress towards closing this gap has been made in several different ways. Firstly, by using a randomized triangulation an expected near-quadratic number of events occur [KRS11]. Secondly, the Voronoi diagram for a convex polygonal distance measure—a distance measure where the unit ball is a convex polygon—can also be maintained with near-quadratic events [AKRS15]. Thirdly, better bounds on the number of events in the Euclidean setting can be proven when motions are more restricted. When the points move along straight-line trajectories with constant speed, then only near-quadratic events occur [Rub15]. Lastly, one could consider not maintaining all edges. Specifically the *stable* edges

Preliminary version (August 10, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

of a Delaunay triangulation can be maintained with a near-quadratic number of events [AGG+15]. Here a Delaunay edge is stable if the angle from either of its endpoints to the endpoints of the corresponding Voronoi edge is large enough. Intuitively, stable edges are those that are not close to flipping as their Voronoi edges are far from collapsing.

A set of easily checked local conditions that implies a global property has also been used in kinetizing other proximity structures. For instance, in the *power diagram* [Aur87] of a set of disjoint balls, the two closest balls must be neighbors [GZ98]—and this diagram can be kinetized by a similar approach. Voronoi diagrams of more general objects, such as convex polytopes, have also been investigated. For example, in $\mathbb{R}^2$ [GSZ00] shows how to maintain a compact Voronoi-like diagram among moving disjoint convex polygons; again, a set of local conditions is derived which implies the global correctness of this diagram. As the polygons move, the structure of this diagram allows one to know the nearest pair of polygons at all times.

In many applications we do not need the full Delaunay triangulation and instead maintaining the nearest neighbor for each vertex or the *minimum spanning tree* may be sufficient. Instead of maintaining the Delaunay graph we can instead maintain the *Theta-* or *Yao*-graph. These are defined as follows. For each point p, the plane is divided into cones, then for each nonempty cone we create an edge from p to the nearest point in that cone, where nearest for the Theta-graph is nearest in the direction of the cone, whereas for the Yao graph it is the nearest in the Euclidean metric. The Theta-graph based on six cones can be used to maintain the global nearest neighbor pair and the graph encounters $\Theta(n^2)$ events as the number of events is bounded by the number of changes in the sorted orders along the directions of the cones, which are fixed [BGZ97]. Using variations of the Yao graph it is also possible to maintain the minimum spanning tree and nearest neighbors for all vertices, and, although the analysis is more complicated, this also yields a near-quadratic number of events [RAK+15].

TRIANGULATIONS AND TILINGS

Many areas in scientific computation and physical modeling require the maintenance of a triangulation (or more generally a simplicial complex) that approximates a manifold undergoing deformation. The problem of maintaining the Delaunay triangulation of moving points in the plane mentioned above is a special case. More generally, local re-triangulations are necessitated by collapsing triangles, and sometimes required in order to avoid undesirably "thin" triangles. In certain cases the number of nodes (points) may also have to change in order to stay sufficiently faithful to the underlying physical process; see, for example, [CDES01]. Because in general a triangulation meeting certain criteria is not unique or canonical, it becomes more difficult to assess the efficiency of kinetic algorithms for solving such problems. The lower-bounds in [ABB+00] indicate that one cannot hope for a subquadratic bound on the number of events in the worst case for the maintenance of *any* triangulation, even if a linear number of additional Steiner points is allowed.

When considering deterministic algorithms the best-known result for maintaining a triangulation uses $O(n^{7/3})$ events [ABG+02]. However, when allowing randomized triangulation it was shown that indeed $O(n^{2+\epsilon})$ events is possible with high probability. The first near-quadratic triangulation uses a hierarchical scheme

Preliminary version (August 10, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

where first a subset of points is triangulated, splitting the pointset into subsets that are then further triangulated [AWY06]. A more recent result slightly improves the bounds on number of events, but is also simpler [KRS11]. In fact, the approach is closer to the much earlier deterministic result; in both cases, first a pseudotriangulation is created, which is then triangulated further.

COLLISION DETECTION

Collision detection is the problem of determining whether there are insections between objects of a given input set. The more interesting version of this problem is of course when the objects move and collisions have to be detected as soon as they occur, see Chapter 39 for more background on collision detection. Kinetic methods are naturally applicable to the problem of collision detection between moving geometric objects. Typically collisions occur at irregular intervals, so that fixed-time stepping methods have difficulty selecting an appropriate sampling rate to fit both the numerical requirements of the integrator as well as those of collision detection. A kinetic method based on the discrete events that are the failures of relevant geometric conditions can avoid the pitfalls of both oversampling and undersampling the system. For two moving convex polygons in the plane, a kinetic algorithm where the number of events is a function of the relative separation of the two polygons is given in [EGSZ99]. The algorithm is based on constructing certain outer hierarchies on the two polygons. Analogous methods for 3D polytopes were presented in [GXZ01], together with implementation data. Such methods however do not easily extend to situations with many objects, hence a different approach is needed.

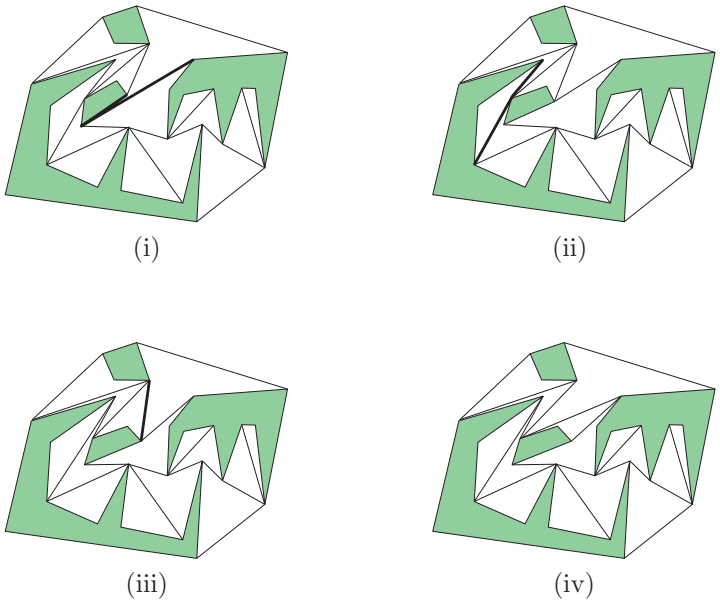


FIGURE 53.2.7
Snapshots of the mixed pseudotriangulation of [ABG+02]. As the center trapezoid-like polygon moves to the right, the edges corresponding to the next about-to-fail certificate are highlighted.

One option is to tile the free space around objects. Such a tiling can serve as a proof of nonintersection of the objects. If such a tiling can be efficiently maintained under object motion, then it can be the basis of a kinetic algorithm for collision detection. Several papers have developed techniques along these lines, including the case of two moving simple polygons in the plane [BEG+04], or multiple moving polygons [ABG+02, KSS02]. These developments all exploit deformable pseudotriangulations of the free space—tilings which undergo fewer combinatorial changes than, for example, triangulations. An example from [ABG+02] is shown in Figure 53.2.7. The figure shows how the pseudotriangulation adjusts by local retiling to the motion of the inner quadrilateral. An advantage of all these methods is that the number of certificates needed is close to the size of the min-link separating subdivision of the objects, and thus sensitive to how intertwined the objects are.

Deformable and nonpolygonal objects are more challenging to handle. Static methods such as bounding volume hierarchies [GLM96] deal with more complex objects by using two phases. In the *broad phase* a selection of pairs of objects is found that may be colliding. In the *narrow phase* these pairs of objects are explicitly tested for intersection. Efficiency of this method depends heavily on the number of pairs produced in the broad phase, so also broad-phase algorithms have been studied within the KDS framework. In a kinetic setting the pairs produced by the broad phase can be used as certificates of nonintersection. Based on this idea efficient KDSs have been produced for deformable [AGN+04] and constant-complexity convex objects in 3D [ABPS09]. In both results the number of potentially intersecting pairs, and hence, the number of certificates is close to linear. For unit balls of similar size a kinetic data structure has also been developed based on dividing space into cells and maintaining which cells the spheres intersect [KGS98]. This method was also used in a more general setting where motions are not known exactly and experimentally shown to work well under reasonable motions [KGS05].

CONNECTIVITY AND CLUSTERING

Closely related to proximity problems is the issue of maintaining structures encoding connectivity among moving geometric objects. Connectivity problems arise frequently in *ad hoc* mobile communication and sensor networks, where the viability of links may depend on proximity or direct line-of-sight visibility among the stations desiring to communicate. With some assumptions, the communication range of each station can be modeled by a geometric region, so that two stations can establish a link if and only if their respective regions overlap. There has been work on kinetically maintaining the connected components of the union of a set of moving geometric regions for the case of rectangles [HS99] and unit disks [GHSZ01].

Clustering mobile nodes is an essential step in many algorithms for establishing communication hierarchies, or otherwise structuring *ad hoc* networks. Nodes in close proximity can communicate directly, using simpler protocols; correspondingly, well-separated clusters can reuse scarce resources, such as the same frequency or time-division multiplexing communication scheme, without interference. Maintaining clusters of mobile nodes requires a tradeoff between the tightness, or optimality of the clustering, and its stability under motion. A basic clustering scheme is to cover the points with a minimum number of unit boxes. A randomized scheme that maintains such a clustering was given by Gao *et al.* [GGH+03]. This scheme maintains a number of boxes that is within a (large) constant factor of the minimum

Preliminary version (August 10, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

number needed. Later, a deterministic solution was found that works in higher dimensions with an approximation ratio of 3^d in d dimensions [Her05]. Both schemes have the nice property that the clustering is smooth, that is, each event affects only a small number of boxes.

A next step in establishing a connected network between mobile nodes is to create a connectivity graph that can be used for routing. Graphs that work well for this purpose have a bounded *spanning ratio*. That is, for any two nodes, the ratio between their Euclidean distance and their shortest distance along the graph is bounded. Here, the distance along the graph is the sum of Euclidean lengths of all edges along a path between the two nodes. Graph for which this spanning ratio is bounded are generally referred to as *spanners*. One of the earliest kinetic spanners was based on the randomized clustering scheme from [GGH+03] and had a constant spanning ratio. Using a different approach Gao *et al.* [GGN06] proposed a spanner with a spanning ratio of $(1+\epsilon)$ for any constant $\epsilon > 0$. This spanner also allows logarithmic time insertion and deletion of points. However, the number of events and query time depend on the spread of the input set. That is, the ratio between the shortest distance and longest distance between any pair of points. The dependency on the spread was removed by Abam *et al.* who present two spanners with spanning ratio $(1+\epsilon)$. The first works for any constant number of dimensions [AB11], but is somewhat complex, whereas the other works only in two dimensions, but is much simpler [ABG10].

VISIBILITY

The problem of maintaining the visible parts of the environment when an observer is moving is one of the classic questions in computer graphics and has motivated significant developments, such as binary space partition trees, the hardware depth buffer, etc. The difficulty of the question increases significantly when the environment itself includes moving objects; whatever visibility structures accelerate occlusion culling for the moving observer must now themselves be maintained under object motion.

Binary space partitions (BSP) are hierarchical partitions of space into convex tiles obtained by performing planar cuts (Section 33.8.2). Tiles are refined by further cuts until the interior of each tile is free of objects or contains geometry of limited complexity. Once a BSP tree is available, a correct visibility ordering for all geometry fragments in the tiles can be easily determined and incrementally maintained as the observer moves. A kinetic algorithm for visibility can be devised by maintaining a BSP tree as the objects move. The key insight is to certify the correctness of the BSP tree through certain combinatorial conditions, whose failure triggers localized tree rearrangements—most of the classical BSP construction algorithms do not have this property. In $\mathbb{R}^2$, a randomized algorithm for maintaining a BSP of moving disjoint line segments is given in [AGMV00]. The algorithm processes $O(n^2)$ events, the expected cost per tree update is $O(\log n)$, and the expected tree size is $O(n \log n)$. The maintenance cost increases to $O(n\lambda_{s+2}(n) \log^2 n)$ [AEG98] for disjoint moving triangles in $\mathbb{R}^3$ (s is a constant depending on the triangle motion). Both of these algorithms are based on variants of vertical decompositions (many of the cuts are parallel to a given direction). It turns out that in practice these generate "sliver-like" BSP tiles that lead to robustness issues [Com99].

As the pioneering work on the visibility complex has shown [PV96], another

structure that is well suited to visibility queries in $\mathbb{R}^2$ is an appropriate pseudo-triangulation. Given a moving observer and convex moving obstacles, a full radial decomposition of the free space around the observer is quite expensive to maintain. One can build pseudotriangulations of the free space that become more and more like the radial decomposition as we get closer to the observer. Thus one can have a structure that compactly encodes the changing visibility polygon around the observer, while being quite stable in regions of the free space well occluded from the observer [HH02].

FACILITY LOCATION

A common networking strategy is to use hierarchies, where a small set of the nodes is selected as hubs that serve as communication gateways for the nodes around them. We can then establish a network between these hubs either inductively, by sampling superhubs, or through another paradigm. These networks have the advantage that routing is generally very simple as each node just needs to communicate to its nearest hub and the hubs can be connected with much simpler protocols, because there are only few. Selecting hubs from a set of nodes can be seen as the classic facility location problem, originally studied in the context of a facility (store, distribution centers, post office) serving a number of clients (customers, stores, addresses). Here the facilities are the network hubs and the clients the other nodes. Many variations of this problem exist, which consider the maintenance cost of operating a facility, the distance of clients to the facility, and the number of clients a facility can serve.

With the increased number of mobile networks it becomes interesting to study this problem in a mobile setting, where clients and facilities move. Clients may then be promoted to servers and servers demoted to clients. A recently studied variant considers a set of moving points that may be used as client or as facility moving (as before) along pseudo-algebraic trajectories. Each point has a maintenance cost when it is used as a facility and a demand when it is a client [DGL10]. The quality measure here is the total maintenance cost of all facilities plus the distances of each client to its nearest facility multiplied by their demand. In this setting a constant factor approximation of the optimal solution can be maintained with a quadratic number of events and a logarithmic number of status changes (changing a client to a facility of vice versa) per event. Note that the number of events and number of changes per event also depend on the ratio between the largest and smallest maintenance cost as well as the ratio between the largest and smallest demand of any point.

Unfortunately these status changes may be quite expensive in practice as they necessitate changes in network structure, which often results in loss of packages. An open problem is whether an efficient strategy exists that takes the cost of a status change into account explicitly. Such an algorithm would have to be analyzed in a competitive way.

QUERYING MOVING OBJECTS

Continuous tracking of a geometric attribute may be more than is needed for some applications. There may be time intervals during which the value of the attribute is of no interest; in other scenarios we may be just interested to know the attribute

Preliminary version (August 10, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

value at certain discrete query times. For example, given n moving points in $\mathbb{R}^2$, we may want to pose queries asking for all points inside a rectangle R at time t, for various values of R and t, or for an interval of time Δt, etc. A number of other classical range-searching structures, such as k-d-trees and R-trees been investigated for moving objects [PAHP02, ABS09].

Another interesting observation is that maintenance of a kinetic data structure may be made more efficient if we allow queries to spend more time. Such trade-offs were investigated for kinetic dictionaries [Ber03, AAE03] and for sorting and convex hulls [AB07].

OPEN PROBLEMS

As mentioned above, we still lack efficient kinetic data structures for many fundamental geometric questions. Here is a short list of such open problems:

1. Find an efficient (and responsive, local, and compact) KDS for maintaining the convex hull of points moving in dimensions $d \geq 3$.
2. Find an efficient KDS for maintaining the smallest enclosing disk in $d \geq 2$. For $d = 2$, a goal would be an $O(n^{2+\epsilon})$ algorithm.
3. Find a KDS to maintain the MST of moving points under the Euclidean metric achieving subquadratic bounds.
4. Maintain mobile facilities with a guaranteed small number of facility changes while maintaining low cost in terms of facility maintenance and travel distance of clients.

Beyond specific problems, there are also several important structural issues that require further research in the KDS framework. These include:

Recovery after multiple certificate failures. We have assumed up to now that the KDS assertion cache can detect certificate failures exactly. In real-world applications this may be impossible due to inexact knowledge of the trajectories or finite precision computations. Multiple certificates may fail at the same time or may be treated in the wrong order. Although for some structures work has been done to create robust KDSs this is still ongoing work and a key step in making KDSs more applicable in practice [AABY11].

There is also a related subtlety in the way that a KDS assertion cache can certify the value, or a computation yielding the value, of the attribute of interest. Suppose our goal is to certify that a set of moving points in the plane, in a given circular order, always forms a convex polygon. A plausible certificate set for convexity is that all interior angles of the polygon are convex. See Figure 53.2.8. In the normal KDS setting where we can always predict accurately the next certificate failure, it turns out that the above certificate set is sufficient, *as long as at the beginning of the motion the polygon was convex*. One can draw, however, nonconvex self-intersecting polygons all of whose interior angles are convex, as shown in the same figure. The point here is that a standard KDS can offer a *historical* proof of the convexity of the polygon by relying on the fact that the certificate set is valid *and* that the polygon was convex during the prior history of the motion. Indeed the counterexample shown cannot arise under continuous motion without one of the angle certificates failing first. On the other hand, if an oracle can move the points

Preliminary version (August 10, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

when "we are not looking," we can wake up and find all the angle certificates to be valid, yet our polygon need not be convex. Thus in this oracle setting, since we cannot be sure that no certificates failed during the time step, we must insist on *absolute* proofs—certificate sets that in any state of the world fully validate the attribute computation or value.

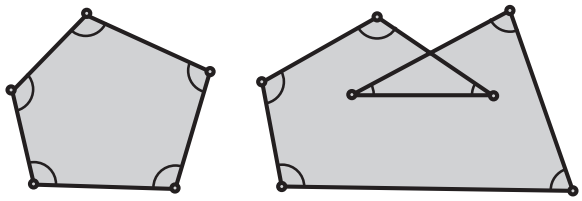


FIGURE 53.2.8
Certifying the convexity of a polygon.

Hierarchical motion descriptions. 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 superposition 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.

What both of these examples suggest is that there can be economies in motion description, if the motion of objects in the environment can be described as a superposition of terms, some of which can be shared among several objects. Such hierarchical motion descriptions can simplify certificate evaluations, as certificates are often local assertions concerning nearby objects, and nearby objects tend to share motion components. For example, in a simple articulated figure, we may wish to assert $\text{CCW}(A, B, C)$ to indicate that an arm is not fully extended, where $\overline{AB}$ and $\overline{BC}$ are the upper and lower parts of the arm, respectively. Evaluating this certificate is clearly better done in the local coordinate frame of the upper arm than in a world frame—the redundant motions of the legs and torso have already been factored out.

Motion sensitivity. As already mentioned, the motions of objects in the world are often highly correlated and it behooves us to find representations and data structures that exploit such motion coherence. It is also important to find mathematical measures that capture the degree of coherence of a motion and then use this as a parameter to quantify the performance of motion algorithms. If we do not do this, our algorithm design may be aimed at unrealistic worst-case behavior, without capturing solutions that exploit the special structure of the motion data that actually arise in practice—as already discussed in a related setting in [BSVK02]. Thus it is important to develop a class of kinetic *motion-sensitive* algorithms, whose performance can be expressed as a function of how coherent the motions of the underlying objects are.

Noncanonical structures. The complexity measures for KDSs mentioned ear-

Preliminary version (August 10, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

lier are more suitable for maintaining *canonical* geometric structures, which are uniquely defined by the position of the data, e.g., convex hull, closest pair, and Delaunay triangulation. In these cases the notion of external events is well defined and is independent of the algorithm used to maintain the structure. On the other hand, as we already discussed, suppose we want to maintain a triangulation of a moving point set. Since the triangulation of a point set is not unique, the external events depend on the triangulation being maintained, and thus depend on the algorithm. This makes it difficult to analyze the efficiency of a kinetic triangulation algorithm. Most of the current approaches for maintaining noncanonical structures artificially impose canonicality and maintain the resulting canonical structure. But this typically increases the number of events. So it is entirely possible that methods in which the current form of the structure may depend on its past history can be more efficient. Unfortunately, we lack mathematical techniques for analyzing such history-dependent structures.

53.3 SOURCES AND RELATED MATERIALS

SURVEYS

Results not given an explicit reference above may be traced in these surveys.

[Gui98]: An early, and by now somewhat dated, survey of KDS work.

[AGE+02]: A report based on an NSF-ARO workshop, addressing several issues on modeling motion from the perspective of a variety of disciplines.

[Gui02]: A "popular-science" type article containing material related to the costs of sensing and communication for tracking motion in the real world.

RELATED CHAPTERS

Chapter 9: Geometry and topology of polygonal linkages
Chapter 26: Convex hull computations
Chapter 27: Voronoi diagrams and Delaunay triangulations
Chapter 29: Triangulations and mesh generation
Chapter 39: Collision and proximity queries

REFERENCES

[AABY11] M.A. Abam, P.K. Agarwal, M. de Berg, and H. Yu. Out-of-order event processing in kinetic data structures. *Algorithmica*, 60:250–273, 2011.

[AAE03] P.K. Agarwal, L. Arge, and J. Erickson. Indexing moving points. *J. Comp. Syst. Sci.*, 66:207–243, 2003.

[AB07] M.A. Abam and M. de Berg. Kinetic sorting and kinetic convex hulls. *Comput. Geom.*, 37:16–26, 2007.

[AB11] M.A. Abam and M. de Berg. Kinetic spanners in $\mathbb{R}^d$. *Discrete Comput. Geom.*,

45:723–736, 2011.

[ABB+00] P.K. Agarwal, J. Basch, M. de Berg, L.J. Guibas, and J. Hershberger. Lower bounds for kinetic planar subdivisions. *Discrete Comput. Geom.*, 24:721–733, 2000.

[ABG+02] P.K. Agarwal, J. Basch, L.J. Guibas, J. Hershberger, and L. Zhang. Deformable free-space tilings for kinetic collision detection. *I. J. Robotic Res.*, 21:179–198, 2002.

[ABG10] M.A. Abam, M. de Berg, and J. Gudmundsson. A simple and efficient kinetic spanner. *Comput. Geom.*, 43:251–256, 2010.

[ABPS09] M.A. Abam, M. de Berg, S.-H. Poon, and B. Speckmann. Kinetic collision detection for convex fat objects. *Algorithmica*, 53:457–473, 2009.

[ABS09] M.A. Abam, M. de Berg, and B. Speckmann. Kinetic kd-trees and longest-side kd-trees. *SIAM J. Comput.*, 39:1219–1232, 2009.

[AEG98] P.K. Agarwal, J. Erickson, and L.J. Guibas. Kinetic binary space partitions for intersecting segments and disjoint triangles (extended abstract). In *Proc. 9th ACM-SIAM Sympos. Discrete Algorithms*, pages 107–116, 1998.

[AGG+15] P.K. Agarwal, J. Gao, L.J. Guibas, H. Kaplan, N. Rubin, and M. Sharir. Stable Delaunay graphs. *Discrete Comput. Geom.*, 54:905–929, 2015.

[AGHV01] P.K. Agarwal, L.J. Guibas, J. Hershberger, and E. Veach. Maintaining the extent of a moving point set. *Discrete Comput. Geom.*, 26:353–374, 2001.

[AGMR98] 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.

[AGMV00] P.K. Agarwal, L.J. Guibas, T.M. Murali, and J.S. Vitter. Cylindrical static and kinetic binary space partitions. *Comput. Geom.*, 16:103–127, 2000.

[AGE+02] P.K. Agarwal, L. Guibas, H. Edelsbrunner, et al. Algorithmic issues in modeling motion. *ACM Computing Surveys*, 34:550-572, 2002.

[AGN+04] P.K. Agarwal, L.J. Guibas, A. Nguyen, D. Russel, and L. Zhang. Collision detection for deforming necklaces. *Comput. Geom.*, 28:137–163, 2004.

[AHP01] P.K. Agarwal and S. Har-Peled. Maintaining approximate extent measures of moving points. In *Proc. 12th ACM-SIAM Sympos. Discrete Algorithms*, pages 148–157, 2001.

[AKRS15] P.K. Agarwal, H. Kaplan, N. Rubin, and M. Sharir. Kinetic Voronoi diagrams and Delaunay triangulations under polygonal distance functions. *Discrete Comput. Geom.*, 54:871–904, 2015.

[ASS96] P.K. Agarwal, O. Schwarzkopf, and M. Sharir. The overlay of lower envelopes and its applications. *Discrete Comput. Geom.*, 15:1–13, 1996.

[Ata85] M.J. Atallah. Some dynamic computational geometry problems. *Comput. Math. Appl.*, 11:1171–1181, 1985.

[Aur87] F. Aurenhammer. Power diagrams: Properties, algorithms and applications. *SIAM J. Comput.*, 16:78–96, 1987.

[AWY06] P.K. Agarwal, Y. Wang, and H. Yu. A two-dimensional kinetic triangulation with near-quadratic topological changes. *Discrete Comput. Geom.*, 36:573–592, 2006.

[BEG+04] J. Basch, J. Erickson, L.J. Guibas, J. Hershberger, and L. Zhang. Kinetic collision detection between two simple polygons. *Comput. Geom.*, 27:211–235, 2004.

[Ber03] M. de Berg. Kinetic dictionaries: How to shoot a moving target. In *Proc. 11th European Sympos. Algorithms*, pages 172–183, vol. 2832 of *LNCS*, Springer, Berlin, 2003.

Preliminary version (August 10, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

[BGH99] J. Basch, L.J. Guibas, and J. Hershberger. Data structures for mobile data. *J. Algorithms*, 31:1–28, 1999.

[BGZ97] J. Basch, L.J. Guibas, and L. Zhang. Proximity problems on moving points. In *Proc. 13th Sympos. Comput. Geom.*, pages 344–351, ACM Press, 1997.

[BRS12a] M. de Berg, M. Roeloffzen, and B. Speckmann. Kinetic compressed quadtrees in the black-box model with applications to collision detection for low-density scenes. In *Proc. 20th European Sympos. Algorithms*, pages 383–394, vol. 7501 of *LNCS*, Springer, Berlin, 2012.

[BRS12b] M. de Berg, M. Roeloffzen, and B. Speckmann. Kinetic convex hulls, Delaunay triangulations and connectivity structures in the black-box model. *J. Comput. Geom.*, 3:222–249, 2012.

[BRS13] M. de Berg, M. Roeloffzen, and B. Speckmann. Kinetic 2-centers in the black-box model. In *Proc. 29th Sympos. Comput. Geom.*, pages 145–154, ACM Press, 2013.

[BSVK02] M. de Berg, A.F. van der Stappen, J. Vleugels, and M.J. Katz. Realistic input models for geometric algorithms. *Algorithmica*, 34:81–97, 2002.

[CDES01] H.-L. Cheng, T.K. Dey, H. Edelsbrunner, and J. Sullivan. Dynamic skin triangulation. *Discrete Comput. Geom.*, 25:525–568, 2001.

[CMP09] M. Cho, D.M. Mount, and E. Park. Maintaining nets and net trees under incremental motion. In *Proc. 20th Sympos. Internat. Sympos. Algorithms Computation*, pages 1134–1143, vol. 5878 of *LNCS*, Springer, Berlin, 2009.

[Com99] J. Comba. *Kinetic Vertical Decomposition Trees*. PhD thesis, Stanford University, 1999.

[Del34] B.N. Delaunay. Sur la sphére vide: à la mémoire de Georges Voronoi. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskih i Estestvennyh Nauk*, 7:793–800, 1934.

[DGL10] B. Degener, J. Gehweiler, and C. Lammersen. Kinetic facility location. *Algorithmica*, 57:562–584, 2010.

[EGSZ99] J. Erickson, L.J. Guibas, J. Stolfi, and L. Zhang. Separation-sensitive collision detection for convex objects. In *Proc. 10th ACM-SIAM Sympos. Discrete Algorithms*, pages 102–111, 1999.

[EKLS16] W. Evans, D. Kirkpatrick, M. Löffler, and F. Staals. Minimizing Co-location potential of moving entities. *SIAM J. Comput.*, 45:1870–1893, 2016.

[GGH+03] J. Gao, L. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Discrete mobile centers. *Discrete Comput. Geom.*, 30:45–63, 2003.

[GGN06] J. Gao, L.J. Guibas, and A.T. Nguyen. Deformable spanners and applications. *Comput. Geom.*, 35:2–19, 2006.

[GHSZ01] L.J. Guibas, J. Hershberger, S. Suri, and L. Zhang. Kinetic connectivity for unit disks. *Discrete Comput. Geom.*, 25:591–610, 2001.

[GJS96] P. Gupta, R. Janardan, and M. Smid. Fast algorithms for collision and proximity problems involving moving geometric objects. *Comput. Geom.*, 6:371–391, 1996.

[GLM96] S. Gottschalk, M.C. Lin, and D. Manocha. OBB-Tree: A hierarchical structure for rapid interference detection. In *Proc. 23rd Conf. Comp. Graphics Interactive Techniques*, pages 171–180, ACM Press, 1996.

[GSZ00] L. Guibas, J. Snoeyink, and L. Zhang. Compact Voronoi diagrams for moving convex polygons. In *Proc. Scand. Workshop Algorithms Data Structures*, vol. 1851 of *LNCS*, pages 339–352, Springer, Berlin, 2000.

Preliminary version (August 10, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

[Gui98] L. Guibas. Kinetic data structures: A state of the art report. In *Proc. 3rd Workshop Algorithmic Found. Robot.*, pages 191–209, A.K. Peters, Natick, 1998.

[Gui02] L.J. Guibas. Sensing, tracking, and reasoning with relations. *IEEE Signal Proc. Magazine*, 19:73–85, 2002.

[GXZ01] L.J. Guibas, F. Xie, and L. Zhang. Kinetic collision detection: Algorithms and experiments. In *Proc. IEEE Internat. Conf. Robotics and Automation*, pages 2903–2910, 2001.

[GZ98] L. Guibas and L. Zhang. Euclidean proximity and power diagrams. In *Proc. 10th Canad. Conf. Comput. Geom.*, pages 90–91, Montréal, 1998.

[Her05] J. Hershberger. Smooth kinetic maintenance of clusters. *Comput. Geom.*, 31:3–30, 2005.

[HH02] O. Hall-Holt. *Kinetic Visibility*. PhD thesis, Stanford University, 2002.

[HS99] J. Hershberger and S. Suri. Kinetic connectivity of rectangles. In *Proc. 15th Sympos. Comput. Geom.*, pages 237–246, ACM Press, 1999.

[Kah91] S. Kahan. A model for data in motion. In *23rd ACM Sympos. Theory of Comput.*, pages 267–277, 1991.

[KGS98] D.-J. Kim, L.J. Guibas, and S.-Y. Shin. Fast collision detection among multiple moving spheres. *IEEE Trans. Vis. Comput. Graph.*, 4:230–242, 1998.

[KGS05] H.K. Kim, L.J. Guibas, and S.Y. Shin. Efficient collision detection among moving spheres with unknown trajectories. *Algorithmica*, 43:195–210, 2005.

[KRS11] H. Kaplan, N. Rubin, and M. Sharir. A kinetic triangulation scheme for moving points in the plane. *Comput. Geom.*, 44:191–205, 2011.

[KSS02] D.G. Kirkpatrick, J. Snoeyink, and B. Speckmann. Kinetic collision detection for simple polygons. *Int. J. Comput. Geom. Appl.*, 12:3–27, 2002.

[MNP+04] D.M. Mount, N.S. Netanyahu, C.D. Piatko, R. Silverman, and A.Y. Wu. A computational framework for incremental motion. In *Proc. 20th Sympos. Comput. Geom.*, pages 200–209, ACM Press, 2004.

[PAHP02] C.M. Procopiuc, P.K. Agarwal, and S. Har-Peled. STAR-tree: An efficient self-adjusting index for moving points. In *Proc. 4th Workshop on Algorithms Engineering Experiments*, vol. 2409 of *LNCS*, pages 178–193, Springer, Berlin, 2002.

[PV96] M. Pocchiola and G. Vegter. The visibility complex. *Internat. J. Comput. Geom. Appl.*, 6:279–308, 1996.

[RAK+15] Z. Rahmati, M.A. Abam, V. King, S. Whitesides, and A. Zarei. A simple, faster method for kinetic proximity problems. *Comput. Geom.*, 48:342–359, 2015.

[Rub15] N. Rubin. On kinetic Delaunay triangulations: A near-quadratic bound for unit speed motions. *J. ACM*, 62:25, 2015.

[Sha94] M. Sharir. Almost tight upper bounds for lower envelopes in higher dimensions. *Discrete Comput. Geom.*, 12:327–345, 1994.

[ST95a] E. Schömer and C. Thiel. Subquadratic algorithms for the general collision detection problem. In *Abstracts 12th European Workshop Comput. Geom.*, pages 95–101, Linz, 1995.

[ST95b] E. Schömer and C. Thiel. Efficient collision detection for moving polyhedra. In *Proc. 11th Sympos. Comput. Geom.*, pages 51–60, ACM Press, 1995.

[YZ12] K. Yi and Q. Zhang. Multidimensional online tracking. *ACM Trans. Algorithms*, 8:12, 2012.

Preliminary version (August 10, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.