31 MOTION

Leonidas J. Guibas

31.1 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, until fairly recently, proper abstractions were lacking and there was only limited work on algorithmic descriptions of motion and their associated complexity measures. This chapter is aimed to show how an algorithmic study of motion is intimately tied to discrete and computational geometry. After a quick survey of earlier work (Sections 31.2 and 31.3), we devote the bulk of this chapter to discussing the framework of *Kinetic Data Structures* (Section 31.4) [Gui98, BGH99]. We also briefly discuss methods for querying moving objects (Section 31.5).

31.2 MOTION IN COMPUTATIONAL GEOMETRY

Dynamic computational geometry refers to a study of combinatorial changes in a geometric structure, as its defining objects undergo prescribed motions. For instance, we may have n points moving linearly with constant velocities in $\mathcal{R}^2$ 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 or minimum separation determination [GJS96, ST95, ST96]. For instance, [ST96] shows how to check in subquadratic time whether two collections of simple geometric objects (spheres, triangles) collide with each other under specified polynomial motions.

31.3 MOTION MODELS

An issue in the above works is that object motion(s) are always assumed to be known in advance, sometimes in explicit form (e.g., points moving as polynomial

0-8493-8524-5/97/$0.00+$.50

functions of time). Indeed, the proposed methods reduce questions about moving objects to other questions about derived static objects.

Now while most evolving physical systems follow known physical laws, it also frequently the case that discrete events occur (such as collisions, for instance) that alter the 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.

In real-world settings the motion of objects may be imperfectly known and better information may only be obtainable at considerable expense. The model of *data in motion* of [Kah91] assumes that upper bounds on the rate of change are known and focuses on being selective in using sensing to obtain additional information about the objects, in order to answer a series of queries.

31.4 KINETIC DATA STRUCTURES

Suppose we are interested in tracking high-level attributes of a geometric system of objects in motion as, for example, the convex hull of a set on n points moving in $\mathcal{R}^2$. 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*, 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.

GLOSSARY

Kinetic data structure: A kinetic data structure (KDS) for a geometric attribute of interest 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 of them fails.

Certificate: A certificate is one of the elementary geometric relations 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.

The inner loop of a KDS consists of repeated certificate failures and certification repairs.

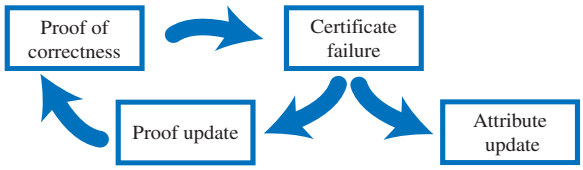


FIGURE 31.4.1
The inner loop of a kinetic data structure.

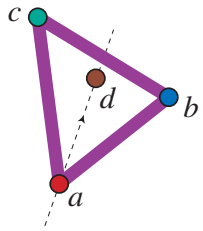
We remark that in the KDS framework objects are allowed to change their motions at will, with appropriate notification to the data structure. When this happens all certificates involving the object whose motion has changed must re-evaluate their failure times.

CONVEX HULL EXAMPLE

Suppose we have four points a, b, c, and d in $\mathcal{R}^2$ 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 counterclockwise oriented. Figure 31.4.2 shows a configuration of the four points and four CCW relations that hold among them. It turns out that these four relations are sufficient to prove that the convex of the four points is the triangle abc. Indeed the points can move an form different configurations, but as long as the four certificates shown remain valid, the convex hull must be abc.

Now suppose that points a, b, and c are stationary and only point d is moving as shown in Figure 31.4.3. 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 convex hull has now changed to $abdc$.

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.



Proof of correctness:

- CCW(a, b, c)
- CCW(d, b, c)
- CCW(d, c, a)
- CCW(d, a, b)

FIGURE 31.4.2
Determining the convex hull of the points.

Old proof	New proof
CCW(a, b, c)	CCW(a, b, c)
CCW(d, b, c)	CCW(c, b, d)
CCW(d, c, a)	CCW(d, c, a)
CCW(d, a, b)	CCW(d, a, b)

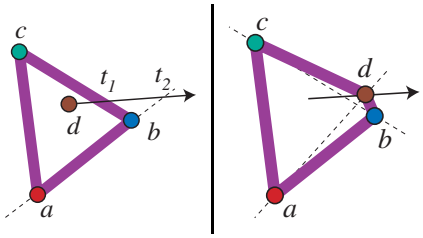


FIGURE 31.4.3
Updating the convex hull of the points.

PERFORMANCE MEASURES FOR KDS

Since a KDS is not meant to facilitate a terminating computation but an ongoing process, we need to use somewhat different measures to assess its complexity. In classical data structures there is usually a trade-off 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.

GLOSSARY

responsiveness: A KDS is called *responsive* if the cost, when a certificate fails,

of repairing the certificate set and updating the attribute computation is small[1].

efficiency: A KDS is called *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[2].

compactness: A KDS is called *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 called *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[3].

A description of a KDS for the general case of maintaining the convex hull of n moving points in $\mathcal{R}^2$ can be found in [BGH99], as well as the proof that it is responsive, efficient, local, and compact. No comparable structure is known for the convex hull of points in dimensions $d \geq 3$.

EXTENT PROBLEMS

A number of the original problems for which kinetic data structures were developed are aimed at different measures of how 'spread out' a moving set of points in $\mathcal{R}^2$ 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; 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 known upper bound 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 orthogonal 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 instance, the bound of the number of approximate diameter updates in $\mathcal{R}^2$ under linear motion is $O(1/\epsilon)$.

PROXIMITY PROBLEMS

The fundamental proximity structures in computational geometry are the Voro-

[1]Say polylogarithmic in the problem size — in general we consider small quantities that are polylogarithmic or $O(n^\epsilon)$ in the problem size.

[2]Technically, we require that the ratio of total events to external events is small. 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

[3]The existence of local KDSs is an intriguing theoretical question for several geometric attribute functions.

noi Diagram and the Delaunay triangulation. 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 very nice structures, because they admit of 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 $\mathcal{R}^d$, then that triangulation must be Delaunay. This makes it very 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 R^2 is a special case) 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. Though no redundant events happen, an exact upper bound for the total number of such events in the worst-case is still elusive even in R^2, where the best known upper bound is nearly cubic, while the best lower bound only quadratic [AGMR98].

This principle of a set of easily checked local conditions that implies a global property has been used in kinetizing other proximity structures as well. 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 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 the exact L_2 distance between objects is not needed and more relaxed notions of proximity suffice. Polyhedral metrics (such as L_1 or L_∞, for instance) are widely used and the normal unit ball in L_2 can be approximated arbitrarily closely by polyhedral approximants. It is more surprising, however, that if we partition the space around each point into a set of polyhedral cones and maintain a number of directional nearest neighbors to each point in each cone, then we can still capture the globally closest pair of points in the L_2 metric. By directional neighbors here we mean that we measure distance only along a given direction in that cone. This geometric fact follows from a packing argument and is exploited in [BGZ97] to give a different method for maintaining the closest pair of points in $\mathcal{R}^d$. The advantage of this method is that the kinetic events are changes of the sorted order of the points along a set of directions fixed *a priori*, and therefore the total number of events is provably quadratic.

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]. Since 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-bound results in [ABdB+99] indicate that one cannot hope for a subquadratic bound on the number of events in the worst case in the maintenance an *any* triangulation, even if a linear number of additional Steiner points is allowed.

There is big gap between the desired quadratic upper bound and the current state of art. Even for maintaining an arbitrary triangulation of a set of n points moving linearly in the plane, the best-known algorithm processes $O(n^{7/3})$ events [ABG+00] in the worst case. The algorithm actually maintains a pseudo-triangulation of the convex hull of the point set and then a triangulation of each pseudo-triangle. Although there are only $O(n^2)$ events in the pseudo-triangulation, some of the events change too many triangles because of high-degree vertices. Unless additional Steiner points are allowed, there are point configurations for which high-degree vertices are inevitable and therefore some of the events will be expensive. A more clever, global argument is needed to prove a near-quadratic upper bound on the total number of events in the above algorithm. Methods that choose to add additional points, on the other hand, have the burden of defining appropriate trajectories for these Steiner points as well. Finally, today no triangulation that guarantees certain quality on the shapes of triangles as well as a subcubic bound on the number of retiling events is known.

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. 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 3-D polytopes were presented in [GXZ01], together with implementation data.

A tiling of the free space around objects can serve as a proof of non-intersection 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+99], or multiple moving polygons [ABG+00, KSS00]. These developments all exploit deformable pseudotriangulations of the free space — tilings which undergo fewer combinatorial changes than, for example, triangulations. Some maintain canonical pseudotriangulations, while others are based on letting a pseudotriangulation evolve according to the history of the motion. An advantage of such methods is that the number of certificates needed is close to size of the min-link separating subdivision of the objects, and thus sensitive to how intertwined the objects are.

Deformable objects are more challenging to handle. Classical methods, such as bounding volume hierarchies [GLM96], become expensive, as the fixed object hierarchies have to be rebuilt frequently. A possibility is to let the hierarchies

themselves deform continuously, with the bounding volumes defined implicitly in terms of object features. Such an approached was developed for flexible linear objects (such as rope or macromolecules), using combinatorially defined sphere hierarchies in [GNRZ02]. The pseudotriangulation-based methods above can also be adapted to deal with object deformation.

CONNECTIVITY AND CLUSTERING

Closely related to proximity problems is the issue of maintaining structures encoding connectivity among moving geometric objects. Connectivity problems arise frequently is *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 of the union of a set of moving geometric regions for the case of rectangles [HS99] and unit disks [GHSZ00].

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 the same frequency or time-division multiplexing communication scheme, without interference. Maintaining clusters of mobile nodes requires a trade-off between the tightness, or optimality of the clustering and its stability under motion. In [GGH$^+$01b] a randomized clustering scheme is discussed based on a repeated leader-election algorithm that produces a number of clusters within a constant factor of the optimum, and in which the number of cluster changes is also asymptotically optimal. This scheme was used in [GGH$^+$01a] to maintain a routing graph on mobile nodes that is always sparse and in which communication paths exist that are nearly as good as those in the full communication graph.

Another fundamental kinetic question is the maintenance of a minimum spanning tree (MST) among n mobile points in the plane, closely related to earlier work on parametric spanning trees [FBSE96] in a graph whose edge weights are functions of a parameter λ (λ is time in the kinetic setting). Since the MST is determined by the sorted order of the edge weights in the graph, a simple algorithm can be obtained by simply maintaining the sorted list of weights and some auxiliary data structures (such an algorithm is quadratic in the graph size, or $O(n^4)$ in our case). This was improved when the weights are linear functions of time to nearly $O(n^{11/6})$ (subquadratic) for planar graphs or other minor-closed families [AEGH98]. When the weights are the Euclidean distances between moving points, only approximation algorithms are known and the best event bounds are nearly cubic [BGZ97]. For many other optimization problems on geometric graphs, such as shortest paths for example, the corresponding kinetic questions are wide open.

VISIBILITY

The problem of maintaining the visible parts of the environment when an ob-

server 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. 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. They 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 $\mathcal{R}^2$, 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 $\mathcal{R}^3$ (s is a constant depending on the triangle motion). Both of these algorithms are based on variants on 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 some robustness issues [Com99].

As the pioneering work on the visibility complex has shown [PV96], another structure that is well suited to visibility queries in $\mathcal{R}^2$ is an appropriate pseudotriangulation. 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 [OH02].

OPEN PROBLEMS

As mentioned above, we still lack efficient kinetic data structures for many fundamental geometric problems, including minimum spanning circle and triangulation maintenance in $\mathcal{R}^2$, and convex hull in $\mathcal{R}^3$. 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 is repaired after each certificate failure. In many realistic scenarios, however, it is impossible to predict exactly when certificates will fail because explicit motion descriptions may not be available. In such settings we may need to sample the system and thus we must be prepared to deal with multiple (but hopefully few) certificate failures at each time step. A general area of research that this suggests is the study of how to efficiently update common geometric structures, such as convex hulls, Voronoi and Delaunay diagrams, arrangements,

etc., after 'small motions' of the defining geometric objects.

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 form a convex polygon. A plausible certificate set for convexity is that all interior angles of the polygon are convex. See Figure 31.4.4. 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, non-convex self-intersecting polygons all of whose interior angles are convex, as also 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 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.

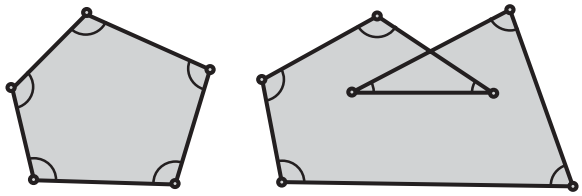


FIGURE 31.4.4
Certifying the convexity of a polygon.

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 $\mathrm{CCW}(A, B, C)$ to indicate that an arm in 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 how coherent motions are and then use this coherence 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 [dBKvdSV97]. Thus it is important to develop a class of kinetic *motion-sensitive* algorithms, whose performance can be expressed a function of how coherent the motions of the underlying objects are.

Non-canonical structures: The complexity measures for KDSs mentioned earlier 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 hard to analyze the efficiency of a kinetic triangulation algorithm. Most of the current approaches for maintaining non-canonical 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 for analyzing such history-dependent structures.

31.5 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 value at certain discrete query times. For example, given n moving points in $\mathcal{R}^2$, 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. Such problems can be handled by a mixture of kinetic and static techniques, including standard range-searching tools such as partition trees and range trees [dBvKOS00]. They typically involve trade-offs between evolving indices kinetically, or prebuilding indices for static snapshots. An especially interesting special case is when we want to be able answer queries about the near future faster than those about the distant future — a natural desideratum in many real-time applications.

A number of other classical range-searching structures, such as *kd*-trees and *R*-trees have recently been investigated for moving objects [AHPP02, AGG02].

31.6 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.

[AG$^+$ar]: 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 XX: To be filled in
Chapter XX: To be filled in
Chapter XX: To be filled in
Chapter XX: To be filled in
Chapter XX: To be filled in

REFERENCES

[ABdB$^+$99] Pankaj K. Agarwal, J. Basch, Mark de Berg, L. J. Guibas, and J. Hershberger. Lower bounds for kinetic planar subdivisions. In *Proc. 15-th Annu. ACM Sympos. Comput. Geom. (SoCG)*, pages 247–254, 1999.

[ABG$^+$00] Pankaj K. Agarwal, J. Basch, L. J. Guibas, J. Hershberger, and L. Zhang. Deformable free space tiling for kinetic collision detection. In *Fourth Workshop on Algorithmic Foundations of Robotics (WAFR)*, pages 83–96, 2000.

[AEG98] Pankaj K. Agarwal, Jeff Erickson, and Leonidas J. Guibas. Kinetic BSPs for intersecting segments and disjoint triangles. In *Proc. 9th ACM-SIAM Sympos. Discrete Algorithms*, pages 107–116, 1998.

[AEGH98] Pankaj K. Agarwal, D. Eppstein, L. J. Guibas, and M. Henzinger. Parametric and kinetic minimum spanning trees. In *Proc. 39th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 596–605, 1998.

[AGG02] P. Agarwal, J. Gao, and L. Guibas. Kinetic medians and *kd*-trees. In *Proc. 10-th Europ. Symp. on Algorithms (ESA)*, pages 5–16, 2002.

[AGMR98] G. Albers, Leonidas J. Guibas, Joseph 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. *Comp. Geometry, Theory and Appl.*, 16:103–127, 2000.

[AG+ar] P.K. Agarwal, L.J. Guibas, et al. Algorithmic issues in modeling motion. *ACM Computing Surveys*, To appear.

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

[AHPP02] P. Agarwal, S. Har-Peled, and M. Procopiuc. Star-tree: An efficent self-adjusting index for moving points. In *Workshop on Algorithms Engineering*, 2002.

[Ata85] M. J. Atallah. Some dynamic computational geometry problems. *Comput. Math. Appl.*, 11(12):1171–1181, 1985.

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

[BEG+99] J. Basch, J. Erickson, L. J. Guibas, J. Hershberger, and L. Zhang. Kinetic collision detection between two simple polygons. In *Proc. 10-th ACM-SIAM Sympos. Discrete Algorithms (SODA)*, pages 327–336, 1999.

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

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

[CDES01] H.L. Cheng, Tamal K. Dey, Herbert Edelsbrunner, and John Sullivan. Dynamic skin triangulation. In *Proc. 12-th SIAM Symposium on Discrete Algorithms (SODA)*, pages 47–56, 2001.

[Com99] J.L.D. Comba. *Kinetic vertical decomposition trees.* PhD thesis, Stanford University, 1999.

[dBKvdSV97] M. de Berg, M. J. Katz, A. F. van der Stappen, and J. Vleugels. Realistic input models for geometric algorithms. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 294–303, 1997.

[dBvKOS00] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications.* Springer-Verlag, Berlin, Germany, 2nd edition, 2000.

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

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

[FBSE96] D. Fernàndez-Baca, G. Slutzki, and D. Eppstein. Using sparsification for parametric minimum spanning tree problems. *Nordic Journal of Computing*, 3:352–366, 1996.

[GGH+01a] J. Gao, L. J. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Geometric spanner for routing in mobile networks. In *Proc. 2n-d ACM Symp. on Ad-Hoc Networking and Computing MobiHoc)*, Oct. 2001.

[GGH+01b] J. Gao, L.J. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Discrete mobile centers. In *Proc. 17-th ACM Symp. on Computational Geometry (SoCG)*, pages 190–198, Jun 2001.

[GHSZ00] Leonidas Guibas, John Hershberger, Subash Suri, and Li Zhang. Kinetic connectivity for unit disks. In *Proc. 16th Annu. ACM Sympos. Comput. Geom.*, pages 331–340, 2000.

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

[GLM96] Stefan Gottschalk, Ming Lin, and Dinesh Manocha. OBB-Tree: A hierarchical structure for rapid interference detection. In *SIGGRAPH 96 Conference Proceedings*, pages 171–180, 1996.

[GNRZ02] L. Guibas, A. Nguyen, D. Russell, and L. Zhang. Collision detection for deforming necklaces. In *Proc. 18-th Annu. ACM Sympos. Comput. Geom. (SoCG)*, pages 33–42, 2002.

[GSZ00] L. Guibas, J. Snoeyink, and L. Zhang. Compact Voronoi diagrams for moving convex polygons. In *Proc. Scand. Workshop on Alg. and Data Structures (SWAT)*, volume 1851 of *Lecture Notes Comput. Sci.*, pages 339–352. Springer-Verlag, 2000.

[Gui98] L. J. Guibas. Kinetic data structures — a state of the art report. In P. K. Agarwal, L. E. Kavraki, and M. Mason, editors, *Proc. Workshop Algorithmic Found. Robot.*, pages 191–209. A. K. Peters, Wellesley, MA, 1998.

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

[GXZ01] L. Guibas, F. Xie, and L. Zhang. Kinetic collision detection: Algorithms and experiments. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 2903–2910, 2001.

[GZ98] L. Guibas and L. Zhang. Euclidean proximity and power diagrams. In *Proc. 10-th Canadian Conf. Computational Geometry*, pages 90–91, 1998.

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

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

[KSS00] David Kirkpatrick, Jack Snoeyink, and Bettina Speckmann. Kinetic collision detection for simple polygons. In *Proc. 16th Annu. ACM Sympos. Comput. Geom.*, pages 322–330, 2000.

[OH02] O.Hall-Holt. *Kinetic visibility.* PhD thesis, Stanford University, 2002.

[PV96] M. Pocchiola and G. Vegter. The visibility complex. *Internat. J. Comput. Geom. Appl.*, 6(3):279–308, 1996.

[ST95] Elmar Schömer and Christian Thiel. Efficient collision detection for moving polyhedra. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages 51–60, 1995.

[ST96] Elmar Schömer and Christian Thiel. Subquadratic algorithms for the general collision detection problem. In *Abstracts 12th European Workshop Comput. Geom.*, pages 95–101. Universität Münster, 1996.