Kinetic Convex Hulls in the Black-Box Model*

Mark de Berg† Marcel Roeloffzen† Bettina Speckmann†

Abstract

Over the past decade, the kinetic-data-structures framework has become the standard in computational geometry for dealing with moving objects. A fundamental assumption underlying the framework is that the motions of the objects are known in advance. This assumption severely limits the applicability of KDSs. We study KDSs in the *black-box model*, which is a hybrid of the KDS model and the traditional time-slicing approach. In this more practical model we receive the position of each object at regular time steps and we have an upper bound on $d_{\max}$, the maximum displacement of any point in one time step.

We study the maintenance of the convex hull of a planar point set P in the black-box model, under the following assumption on $d_{\max}$: there is some constant k such that for any point $p \in P$ the disk of radius $d_{\max}$ contains at most k points. We analyze our algorithms in terms of Δ_k, the so-called *k-spread* of P. We show how to update the convex hull at each time step in $O(k\Delta_k \log^2 n)$ amortized time.

1 Introduction

Motivation. Algorithms dealing with objects in motion traditionally discretize time and recompute the structure of interest at every time step from scratch. This can be wasteful, especially if the time steps are small: then the objects will have moved only slightly, and the structure may not have changed at all. Ideally an object gets attention if and only if its new location triggers an actual change in the structure. *Kinetic data structures (KDSs)*, introduced by Basch *et al.* [3], try to do exactly that: they maintain not only the structure itself, but also additional information that helps to find out when and where the structure will undergo a "real" (combinatorial) change. They maintain a collection of simple geometric tests—these are called *certificates*—with the property that as long as these certificates remain valid, the structure of interest does not change combinatorially. Whenever there is an event—that is, a certificate failure—the KDS is updated. See one of the surveys by Guibas [7, 8, 9] for more information and results on KDSs.

A basic assumption in the KDS framework is that the object trajectories are known. This is necessary to be able to compute the failure times of the certificates, which is essential for the event-driven approach taken in the KDS framework. This assumption severely limits the applicability of the framework. When tracking moving objects, for instance, one gets the object locations only at (probably regular) time steps in an online manner—no detailed knowledge of future trajectories is available. The same is true for physical simulations, where successive locations are computed by a numerical integrator. The goal of our paper is to study the kinetic maintenance of a fundamental geometric structure—the convex hull—in a less restrictive setting: instead of assuming knowledge of the trajectories, we assume only that we know upper bounds on the speeds of the objects and that we get their positions at regular time steps.

Related work. We are not the first to observe that the basic assumption in the KDS model is not always valid. Indeed, the need for a hybrid model, which combines ideas from the KDS model with a traditional time-slicing approach, was already noted in the survey by Agarwal *et al.* [1]. Since then there have been several papers in this direction, as discussed next.

Gao *et al.* [6] study spanners for sets of n moving points in a model where one does not know the trajectories in advance but receives only the positions at each time step. They call this the *blackbox replacement model*—we simply call it the *black-box model*—and show how to update the spanner at each time step in $O(n + k \log \alpha)$ time. Here α is the *spread* of the point set, and k is the number of changes to the hierarchical structure defining their spanner.

Mount *et al.* [11] also study the maintenance of geometric structures in a setting where the trajectories are unknown. They separate the concerns of tracking the points and updating the geometric structure into two modules: the motion processor (MP) is responsible for tracking the points, and the incremental motion algorithm (IM) is responsible for maintaining the geometric structure. Mount *et al.* describe a protocol trying to minimize the interaction between the modules, and they prove that under certain conditions their protocol has good competitive ratio. Their approach goes back to the work of Kahan [10] on certain

*M. Roeloffzen was supported by the Netherlands' Organisation for Scientific Research (NWO) under project no. 600.065.120. B. Speckmann was supported by the Netherlands' Organisation for Scientific Research (NWO) under project no. 639.022.707.

†Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands. {mdberg,mroeloff,speckman}@win.tue.nl

kinetic 1-dimensional problems. See also the more recent work by Cho *et al.* [4]. There is also some work on repairing a triangulation after the vertices have moved. Agarwal *et al.* [2] repair arbitrary planar triangulations, whereas Shewchuk [12] looks at $d-$dimensional Delaunay triangulations.

The results above typically express the running time in terms of the number of changes to the structure at hand, without analyzing this number. This is no surprise; without assumptions on the maximum displacements of the points one cannot say much about the number of changes. On the one hand this "abstract" analysis is appealing since it makes the results general, but on the other hand it becomes hard to decide whether it is better to use these kinetic algorithms or to recompute the structure from scratch at each time step. This is the goal of our paper: to develop KDSs in the black-box model that are provably more efficient than recomputing the structure from scratch under certain assumptions on the trajectories.

Our results. We study black-box KDSs for the convex hull of a set P of n points moving in the plane. As already mentioned, we need to make assumptions on the point movements and time steps to obtain provably efficient solutions. In particular, the time steps should be small enough so that there is some coherence between the positions of the points in consecutive time steps—otherwise we cannot do better than recomputing the structure from scratch. Furthermore, we will assume in most of our results that P is fairly evenly distributed at each time step. We discuss these assumptions in more detail in Section 2.

We present an algorithm that updates the convex hull at each time step in $O(k\Delta_k \log^2 n)$ amortized time, where Δ_k is the k-spread of a point set as defined by Erickson [5]—see Section 2. Note that some proofs are omitted due to space limitations but will be available in the full paper.

2 Preliminaries

Here we introduce some notation, and we discuss some basic issues regarding the black-box model and the concept of k-spread. Although some of our results extend to higher dimensions, we will focus here on the case where P is a set of points moving in the plane.

The black-box model. We denote the position of a point p at time t by $p(t)$, and we let $P(t) := \{p(t) : p \in P\}$ denote the point set at time t. In the black-box model, we assume that we receive the positions at regular time steps $t_0, t_1, \ldots$ and the goal is to update the structure of interest—the convex hull in our case—at each time step. The algorithm need not ask for all new positions at each time step; it may ignore some points if the new locations of these points cannot change the structure. Thus a sublinear update time is potentially feasible and indeed, we show how to obtain sublinear update time for convex-hull maintenance, under certain conditions.

As stated in the introduction, we assume the sampling rate is such that the points in P do not move too much in one time step, as compared to their inter-distances. For a point $p \in P$, let $\text{NN}_k(p, P)$ denote the k-th nearest neighbor of p in $P \setminus \{p\}$. Let $\text{dist}(p, q)$ denote the Euclidean distance between two points p and q, and define $\text{mindist}_k(P) := \min_{p\in P} \text{dist}(p, \text{NN}_k(p, P))$. We assume the sampling rate satisfies the following assumption.

> *Displacement Assumption*: There is a maximum displacement $d_{\max}$ such that
>
> - $d_{\max} \leqslant \min_{t_i} \text{mindist}_k(P(t_i))$, and
> - $\text{dist}(p(t_i), p(t_{i+1})) \leqslant d_{\max}$ for each $p \in P$ and any time step t_i.

The k-spread of a point set. The k-spread Δ_k of P, is defined as

$$\Delta_k(P) := \text{diam}(P)/\text{mindist}_k(P).$$

The k-spread of a point set can be used to bound the number of points within a region if the diameter of the region is not too large.

Lemma 1 *Let P be a set of points in $\mathbb{R}^2$, and let R be a region in $\mathbb{R}^2$ such that $\text{diam}(R) < \text{diam}(P)/\Delta_k(P)$. Then R contains at most k points from P.*

Proof. Assume for contradiction that there are $k+1$ points inside R. Let $p \in P \cap R$. Then $\text{dist}(p, \text{NN}_k(p, P)) \leqslant \text{diam}(R)$. Hence,

$$\Delta_k(P) \geqslant \frac{\text{diam}(P)}{\text{diam}(R)} > \frac{\text{diam}(P)}{\text{diam}(P)/\Delta_k(P)} = \Delta_k(P),$$

a contradiction. □

3 Maintaining the convex hull

Let $\mathcal{CH}(P)$ denote the convex hull of a point set P, and let $\partial\mathcal{CH}(P)$ denote the boundary of $\mathcal{CH}(P)$. In this section we give algorithms to maintain $\mathcal{CH}(P(t))$. From now on, we will use $\mathcal{CH}(t)$ as a shorthand for $\mathcal{CH}(P(t))$. Our algorithms rely on the following observation, which follows from the fact that the distance between p and $\partial\mathcal{CH}(P)$ can change by only $2d_{\max}$ in a single time step.

Lemma 2 *Consider a point $p \in P$, and let $d_p(t) := \text{dist}(p(t), \partial\mathcal{CH}(t))$. Then p cannot become a vertex of $\mathcal{CH}(P)$ until at least $\frac{d_p(t)}{2d_{\max}}$ time steps have passed.*

Lemma 2 suggests the following simple scheme to maintain $\mathcal{CH}(P)$. Compute the initial convex hull

$\mathcal{CH}(t_0)$, and compute for each point $p \in P$ its distance to $\partial\mathcal{CH}(t_0)$. Using this distance and Lemma 2 compute a *time stamp* $t(p)$ for each point p, which is the first time step when p could become a convex-hull vertex. Thus p can be ignored until its time stamp *expires*, that is, until time $t(p)$. In a generic time step t_i, we now determine the set $Q(t_i)$ of all points whose time stamps expire, compute their convex hull and compute new time stamps for the points in $Q(t_i)$. We may use $\mathcal{CH}(t_{i-1})$ to compute the new convex hull, but we don't need to here.

To implement this algorithm we use an array A where $A[t_i]$ contains the points whose time stamps expire at time t_i. To restrict the amount of storage we use an array $A[0..n-1]$ with n entries, and we let time advance through the array in a cyclic manner (using without loss of generality that $t_i = i$). Furthermore, we bound the time stamps to be at most n steps, and we use an approximation of $\mathrm{dist}(p(t), \partial\mathcal{CH}(t))$ to speed up the computations. Our approach is made explicit in Algorithm 1. Note that the algorithm needs to know only $d_{\max}$ to work correctly, it does not need to know bounds on the k-spread.

Algorithm 1: UPDATECH

1 $Q(t) \leftarrow$ set of points stored in $A[t]$
2 Compute $\mathcal{CH}(Q(t))$ and set $\mathcal{CH}(t) \leftarrow \mathcal{CH}(Q(t))$.
3 **for** each $p \in Q(t)$ **do**
4 $\quad d_p^* \leftarrow$ lower bound on $\mathrm{dist}(p(t), \partial\mathcal{CH}(t))$.
5 $\quad$ Set p's time stamp:
$\quad t(p) \leftarrow [t + \min(1 + \lfloor \frac{d_p^*}{2d_{\max}} \rfloor, n)] \mod n$.
6 $\quad$ Add p to $A[t(p)]$.
7 $t \leftarrow (t+1) \mod n$

It remains to describe how to compute $\mathcal{CH}(Q(t))$ in Step 2 and how to compute the values d_p^* in Step 5. Computing $\mathcal{CH}(Q(t))$ can be done by an optimal convex-hull algorithm in $O(|Q(t)| \log |Q(t)|)$ time. To compute d_p^* we proceed as follows. Let q_{above} be the point on $\partial\mathcal{CH}(t)$ directly above p, and define q_{below}, q_{left}, and q_{right} similarly. These points can be found in logarithmic time using binary search. Let $q_{\min}$ denote the minimum distance between p and any of the points q_{above}, q_{below}, q_{left}, and q_{right}. Then we set $d_p^* = q_{\min}/\sqrt{2}$ (Note that $d_p^* \leqslant \mathrm{dist}(p, \partial\mathcal{CH}(t)) \leqslant \sqrt{2}\ d_p^*$.) We get the following result.

Lemma 3 *At each time step t,* UPDATECH *updates the convex hull in $O(|Q(t)| \log n)$ time.*

Next we analyse $|Q(t)|$, the number of time stamps that can expire in a single time step.

Analyzing the number of expiring time stamps. We perform our analysis in terms of Δ_k, which is an upper bound on the k-spread of P at any time. We first bound the number of convex hull vertices.

Lemma 4 *The number of vertices of the convex hull $\mathcal{CH}(P)$ of a point set P is $O(k\Delta_k(P))$.*

Proof. The length of $\partial\mathcal{CH}(P)$ is $\Theta(\mathrm{diam}(P))$, so we can cut $\partial\mathcal{CH}(P)$ into $\Theta(\mathrm{diam}(P)/\mathrm{mindist}_k(P)) = \Theta(\Delta_k(P))$ pieces with a length less then $\mathrm{mindist}_k(P)$. From Lemma 1 we know that each such piece contains at most k points. It follows that $\partial\mathcal{CH}(P)$ contains $O(k\Delta_k(P))$ vertices. □

Now let us consider the number of expiring time stamps. In the worst case it can happen that all time stamps expire in a single time step. However, using an amortization argument we show that on average only $O(k\Delta_k \log n)$ time stamps expire in each time step.

Lemma 5 *The amortized number of time stamps expiring in each time step is $O(k\Delta_k \log n)$.*

Proof. (Sketch) We prove the lemma using the accounting method: each point has an account into which we put a certain amount of money at each time step, and whenever the time stamp of a point expires it has to pay 1 euro from its account. Our scheme is that at time step t_i each point p receives

$$\min\left(1,\ \max\left(\frac{1}{n}, \frac{8\sqrt{2}\cdot d_{\max}}{d_p(t_i)}\right)\right) \text{ euro},$$

where $d_p(t_i) = \mathrm{dist}(p(t_i), \partial\mathcal{CH}(t_i))$.

To prove each point can pay 1 euro when it expires, consider a point p whose time stamp expires at time t_i, and let $t_j < t_i$ be the previous time step when p's time stamp expired. (If there is no such time step, we can take $j = 0$.) Now define $t(p) := t_i - t_j = i - j$ to be the number of time steps from t_j up to t_{i-1}. If $t(p) = n$ then p certainly has enough money in its account at time t_i, so assume this is not the case. Then we can show that $t(p) \geqslant \frac{d_p(t_j)}{2\sqrt{2}\cdot d_{\max}}$.

The distance between $p(t_m)$ and $\mathcal{CH}(t_m)$ for $t_j \leqslant t_m \leqslant t_{i-1}$ is at most $4d_p(t_j)$. Hence point p gets at least $\frac{8\sqrt{2}d_{\max}}{4\cdot d_p(t_j)}$ euro per time step, which proves each point has at least one euro when it expires.

Next we prove that we only spend $O(k\Delta_k \log n)$ euro per time step. We consider the points p such that $d_p(t_i) \leqslant 8\sqrt{2}n \cdot d_{\max}$; the remaining points get $1/n$ euros each, so in total at most 1 euro. We divide these points into groups $G_1, \ldots, G_\ell$ (see Figure 1). Each group G_j contains the points $p \in P$ such that $(j-1)\cdot d_{\max} \leqslant d_p(t_i) \leqslant j \cdot d_{\max}$, where $\ell = 8\sqrt{2}n$. We can show that each group contains $O(k\Delta_k)$ points, so we pay at most $O(k\Delta_k/j)$ euro per group.

Summing this over all groups we see that the amount we pay at each time step is

$$\sum_{j=1}^{8\sqrt{2}n} O\left(\frac{k\Delta_k}{j}\right) = O(k\Delta_k \log n).$$

□

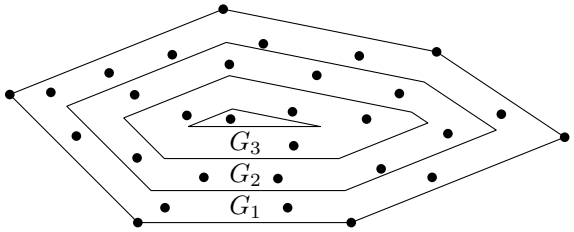

Figure 1: We divide the points into groups based on their distance to $\partial\mathcal{CH}(t_i)$.

From Lemma 3 and 5 we conclude the following:

Theorem 6 *Under the Displacement Assumption, the convex hull of a set P of n points moving in the plane can be maintained in the black-box model in $O(k\Delta_k \log^2 n)$ amortized time per time step, where Δ_k is the maximum k-spread of P at any time.*

4 Conclusion

We presented an algorithm to maintain the convex hull of a planar point set in the KDS black-box model. The algorithm is simple and does not require knowledge of $\Delta_k(P)$ or k: it only needs to know $d_{\max}$, the maximum displacement of any point in one time step.

We spend $O(k\Delta_k \log^2 n)$ amortized time to update the convex hull after each time step. This is optimal up to the logarithmic factors, because the convex hull can undergo $\Omega(k\Delta_k)$ changes in any time step. Moreover, we can show that our bound $O(k\Delta_k \log n)$ on the number of expiring time stamps is tight. However, it may be possible to get rid of one logarithmic factor from the time bound by a more clever algorithm. In fact, if we are allowed to use the floor function, then we know how to to do this. Unfortunately, the resulting algorithm needs to know $\text{mindist}_k(P(t))$, which is perhaps not very realistic. It would be interesting to design an algorithm that needs to know only $d_{\max}$ and achieves $O(k\Delta_k \log n)$ update time. Another interesting open problem is whether it is possible to make the time bound worst-case rather than amortized.

In the full paper we also studied the convex hull problem without a bound on the spread Δ_k. In this case it is still possible to do updates efficiently, namely in $O(n \log k)$ time per time step. Note that when $\Delta_k > n$ all points can appear as vertices on the convex hull giving a lower bound of $\Omega(n)$ on the update time.

Another interesting problem we studied in the KDS black-box model is the Delaunay triangulation. Using the k-spread and displacement assumption we can update the Delaunay triangulation using $O(k^2\Delta_k^2)$ flips in $O(k^2\Delta_k^2 \log n)$ time by moving the points one by one from their old to their new locations. We can reduce the update time to $O(k^2\Delta_k^2)$ by inserting $p(t)$ and removing $p(t-1)$ for each point $p \in P$.

How realistic our model is depends on the application, of course. We expect that in many applications the sampling rate is such that the Displacement Assumption is satisfied. The other question is whether the point set can be expected to have small k-spread. In meshing-type applications, it may be realistic to assume that the k-spread is $O(\sqrt{n})$. In any case, Δ_k seems like a reasonable parameter to measure the efficiency, and we think it will be interesting to study other structures in the KDS black-box model under the Displacement Assumption and to analyze their performance in terms of Δ_k.

References

[1] P.K. Agarwal, L.J. Guibas, H. Edelsbrunner, J. Erickson, M. Isard, S. Har-Peled, J. Hershberger, C. Jensen, L. Kavraki, P. Koehl, M. Lin, D. Manocha, D. Metaxas, B. Mirtich, D. Mount, S. Muthukrishnan, D. Pai, E. Sacks, J. Snoeyink, S. Suri, and O. Wolefson. Algorithmic issues in modelling motion. *ACM Comput. Surv.* 34:550–572 (2002).

[2] P.K. Agarwal, B. Sadri, and H. Yu. Untangling triangulations through local explorations. In *Proc. 24th ACM Sympos. Comput. Geom.*, pages 288–297, 2008.

[3] J. Basch, L.J. Guibas, and J. Hershberger. Data structures for mobile data. In *Proc. 8th ACM-SIAM Sympos. Discr. Algorithms*, pages 747–756, 1997.

[4] M. Cho, D.M. Mount, and E. Park. Maintaining nets and net trees under incremental motion. In *Proc. 20th Int. Sympos. Algo.Comput.*, pages 1134–1143, 2009.

[5] J. Erickson. Dense Point Sets Have Sparse Delaunay Triangulations. In *Discrete and Computational Geometry* 30:83-115 (2005).

[6] J. Gao, L.J. Guibas, A. Nguyen. Deformable spanners and applications. In *Proc. 20th ACM ACM Sympos. Comput. Geom.*, pages 190–199, 2004.

[7] L.J. Guibas. Kinetic data structures—a state-of-the-art report. In *Proc. 3rd Workshop Algorithmic Found. Robot.*, pages 191–209, 1998.

[8] L.J. Guibas. Kinetic data structures. In: D. Mehta and S. Sahni (editors), *Handbook of Data Structures and Applications*, Chapman and Hall/CRC, 2004.

[9] L.J. Guibas. Motion. In: J. Goodman and J. O'Rourke (eds.), *Handbook of Discrete and Computational Geometry (2nd edition)*, pages 1117–1134. CRC Press, 2004.

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

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

[12] R. Shewchuk. Star splaying: an algorithm for repairing Delaunay triangulations and convex hulls. In *Proc. 21st ACM Sympos. Comput. Geom.*, pages 237–246, 2005.