Kinetic convex hulls and Delaunay triangulations in the black-box model

Kinetic Convex Hulls and Delaunay Triangulations in the Black-Box Model

Mark de Berg
mdberg@win.tue.nl

Marcel Roeloffzen*
mroeloff@win.tue.nl

Bettina Speckmann†
speckman@win.tue.nl

Department of Computer Science, TU Eindhoven
PO Box 513, 5600 MB Eindhoven, the Netherlands

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 and the Delaunay triangulation 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. For the Delaunay triangulation our main contribution is an analysis of the standard edge-flipping approach; we show that the number of flips is $O(k^2\Delta_k^2)$ at each time step.

Categories and Subject Descriptors: F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems—*Geometrical problems and computations*

General Terms: Algorithms, Theory

Keywords: Kinetic Data Structures, Black-Box Model, Delaunay Triangulation, Convex Hull

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

†Bettina Speckmann was supported by the Netherlands' Organisation for Scientific Research (NWO) under project no. 639.022.707.

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. Instead of sampling the object locations at regular time intervals, KDSs follow an event-driven approach. 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. A KDS computes for each certificate the nearest time in the future when it will fail and puts all these failure times into an event queue. Whenever there is an event—that is, a certificate failure—the KDS is updated. Note that the fact that we know which certificate has failed when we handle an event gives us valuable information to update the attribute efficiently. See one of the surveys by Guibas [12, 13, 14] 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 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. Our goal is to study the kinetic maintenance of two fundamental geometric structures—convex hulls and Delaunay triangulations—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 framework is not always valid. The need for a hybrid model, which combines ideas from KDSs 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.* [11] 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.* [16] 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. The MP monitors the points to see whether they move "as expected", and notifies the IM when this is no longer the case or some other important event happens. The IM then recomputes the structure, possibly querying the MP for the location of certain points, and gives the MP new motion estimates. 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 [15] on certain kinetic 1-dimensional problems. See also the more recent works by Cho *et al.* [8] and by Yi and Zhang [19].

The following papers show how to repair a triangulation after the vertices have moved. Shewchuk [18] considers d-dimensional Delaunay triangulations. He introduces star splaying, which estimates the neighborhood of each vertex and then resolves all inconsistencies between neighborhoods until the new Delaunay triangulation is found. The worst-case expected running time is $O(n^{\lceil d/2 \rceil +1} + n^2 \log n)$, but the algorithm runs in linear time when the degree of each vertex is $O(1)$. Agarwal *et al.* [2] repair an (arbitrary) planar triangulation by finding "inverted" triangles and then finding regions that can be locally re-triangulated. After $O(n)$ time to find all inverted triangles, they use $O(k^2 \log k)$ time to find and re-triangulate the local regions, where k is the total complexity of these regions. (The analysis by Agarwal *et al.* is more refined and depends on additional parameters that indicate how entangled the triangulation is.)

Experimental work has also been done on kinetic Delaunay triangulations. De Castro *et al.* [5] describe how to easily determine a tolerance region for each point, such that as long as the point remains within its tolerance region we do not have to check its certificates. They then give experimental results showing that for fairly stable Delaunay triangulations this filtering method is faster than traditional KDSs. Russel argues in his thesis [17] that in practise a naive traditional KDS for a Delaunay triangulation is never faster than rebuilding and presents a filtering approach that is faster than rebuilding when the number of certificate failures is small.

The theoretical results discussed above typically express the running time in terms of the number of changes to the structure at hand, without further analyzing this number. This is not surprising, since without assumptions on the maximum displacements of the points one cannot say much about the number of changes. This "abstract" analysis is nice 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 simply 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—under certain assumptions on the trajectories—provably more efficient than recomputing the structure from scratch.

Our results. We study black-box KDSs for the convex hull and the Delaunay triangulation of set P of n points moving in the plane. As mentioned, we 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. Next we discuss our assumptions in detail, and state our results.

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

$$\text{mindist}_k(P) := \min_{p \in P} \text{dist}(p, \text{NN}_k(p, P)).$$

Our basic assumption is that $d_{\max}$, the maximum displacement of any point during one time step, satisfies the *Displacement Assumption*, that $d_{\max} \leqslant \text{mindist}_k(P)$, for some small k. Note that $\text{mindist}_k(P)$ may change as the points move. Thus a more precise statement of the Displacement Assumption is that $d_{\max}$ is bounded by the minimum value of $\text{mindist}_k(P)$ over all time steps—see Section 2. We believe that in many practical applications, the sampling rate will be such that the Displacement Assumption is satisfied.

To describe the distribution of P we use the concept of k-*spread*, as introduced by Erickson [10] and defined as follows. Let $\text{diam}(P)$ denote the diameter of P. Then the k-*spread* of P, denoted by $\Delta_k(P)$, is defined as

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

Note that the 1-spread of P is simply the standard spread. The 1-spread is not very suitable for moving points, however, as it blows up as soon as two points get very close to each other. The k-spread is more robust since it allows up to k objects to get very close to each other without causing a blow-up in the k-spread. Our analyses will be in terms of Δ_k, the maximum k-spread over all time steps, where k is such that the motions satisfy the Displacement Assumption.

For the convex-hull problem, we present an algorithm that updates the convex hull at each time step in $O(k\Delta_k \log^2 n)$ amortized time. We also present a variant of the algorithm whose running time does not depend on the k-spread: for any set of moving points satisfying the Displacement Assumption, it updates the convex hull in $O(n \log k)$ time. Lastly we show how to generalise our convex hull algorithm to higher dimensions.

For the Delaunay triangulation we consider two straightforward algorithms. The first one moves the points one at a time from their old to their new locations, meanwhile updating the Delaunay triangulation using edge flips. The second algorithm deletes each point from the triangulation and reinserts it at its new location; the triangle into which the new location lies is found by walking in the triangulation. Our main contribution lies in the analysis of these simple approaches under the Displacement Assumption and in terms of Δ_k. For example, we show that the simple flipping algorithm performs only $O(k^2 \Delta_k^2)$ flips.

2. PRELIMINARIES

In this section we introduce some notation, and we discuss a few 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 back-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 or the Delaunay triangulation 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—indeed, we will show how to obtain sublinear update time for the 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. More precisely, we assume the sampling rate satisfies the following assumption.

Displacement Assumption: There is a maximum displacement $d_{\max}$ such that

- $d_{\max} \leqslant \min_{t_i} \operatorname{mindist}_k(P(t_i))$, and
- $\operatorname{dist}(p(t_i), p(t_{i+1})) \leqslant d_{\max}$ for each $p \in P$ and any time step t_i.

The k-spread of a point set. Recall that $\Delta_k(P)$, the k-spread of P, is defined as

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

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* $\operatorname{diam}(R) < \operatorname{diam}(P)/\Delta_k(P)$. *Then R contains at most k points from P.*

PROOF. Assume for a contradiction that there are $k+1$ points inside R. Let $p \in P \cap R$. Then

$$\operatorname{dist}(p, \textsc{nn}_k(p, P)) \leqslant \operatorname{diam}(R) < \frac{\operatorname{diam}(P)}{\Delta_k(P)}.$$

Hence,

$$\Delta_k(P) = \frac{\operatorname{diam}(P)}{\operatorname{mindist}_k(P)} > \frac{\operatorname{diam}(P)}{\operatorname{diam}(P)/\Delta_k(P)} = \Delta_k(P),$$

a contradiction. □

COROLLARY 1. *Let B be a minimum bounding square of P, and consider a partitioning of B into a regular grid with $\Delta_k(P) \times \Delta_k(P)$ cells. Then each grid cell contains $O(k)$ points.*

Remark. A statement similar to the converse of Corollary 1 also holds: if a partitioning of the minimum bounding square B into a regular grid with $\Delta \times \Delta$ cells has at most k points per cell, then $\Delta_{k'}(P) = \Delta$ for some $k' = O(k)$. The best case is when a $\sqrt{n} \times \sqrt{n}$ grid has at most k points per cell, so that $\Delta_k(P) = O(\sqrt{n})$. One may think that then the problems become easy, but this is in fact not the case; for

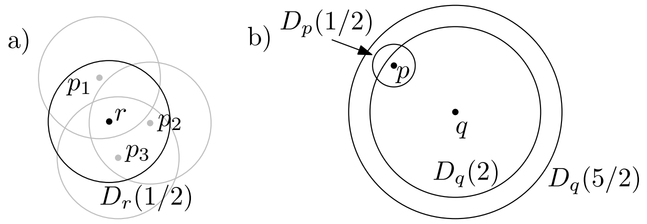

Figure 1: a) r can only be contained in up to k discs $D_p(1/2)$ for $p \in P$. b) $D_q(5/2)$ contains all discs $D_p(1/2)$ for which $p \in D_q(2)$.

example, one can show that maintaining the points from P in x-order still takes $\Omega(n \log n)$ in the black-box model, even for point sets with $\Delta_k(P) = O(\sqrt{n})$.

Remark. For small k, the k-spread can be arbitrarily large. For $k = n-1$, on the other hand, we have $\Delta_k(P) \leqslant 2$. Obviously, the k-spread decreases monotonically (though not necessarily strictly monotonically) as k increases. A natural question is whether anything more precise can be said about how $\Delta_k(P)$ changes as k varies. It is easy to see that we cannot say much about the change in k-spread when k is decreased: $\Delta_{k-1}(P)$ cannot be bounded in terms of $\Delta_k(P)$, since $\operatorname{mindist}_{k-1}(P)$ can be arbitrarily much smaller than $\operatorname{mindist}_k(P)$. When k is increased, on the other hand, then at some point the k-spread will go down.

LEMMA 2. *For a pointset P where the k-spread of P is $\Delta_k(P)$ it holds that $\Delta_{k'}(P) \leqslant \Delta_k(P)/2$ for $k' = 25k$.*

PROOF. Let $D_r(\alpha)$ denote the open disk centered at $r \in \mathbb{R}^2$ with radius $\alpha \cdot \operatorname{mindist}_k$ and let $P_r(\alpha) = P \cap D_r(\alpha)$. From Lemma 1 it follows that a disk $D_r(1/2)$ contains no more then k points of P. We claim that there is no point $r \in \mathbb{R}^2$ such that $r \in D_p(1/2)$ for more than k points $p \in P$. Assume that a point r exists that is in the disk $D_p(1/2)$ for $k+1$ different points $p \in P$, then $D_r(1/2)$ contains $k+1$ points, which contradicts the spread assumption—see Figure 1a. It follows that the intersection depth of the disks $D_p(1/2)$ for all points $p \in P$ is at most k.

Let $q \in P$, then the disks $D_p(1/2)$ for $p \in P_q(2)$ are contained in $D_q(5/2)$ as illustrated in Figure 1b. Because the stabbing depth of the disks $D_p(1/2)$ is at most k the sum of the areas of these disks is at most $k\,|D_q(5/2)|$, where $|D_q(5/2)|$ is the area of $D_q(5/2)$. This gives:

$$|P_q(2)| \leqslant \frac{k\ |D_q(5/2)|}{|D_p(1/2)|} = 25k$$

It follows that $\operatorname{mindist}_{k'} \geqslant 2\operatorname{mindist}_k$ and $\Delta_{k'} \leqslant \Delta_k/2$ for $k' = 25$. □

Finally, observe that Corollary 1 implies that $\Delta_k(P) = \Omega(\sqrt{n/k})$.

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 3. *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 3 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 3, compute a *time stamp* $t(p)$ for p, which is the number of time steps until p can be a vertex of the convex hull. Thus p can be ignored until the time stamp expires after $t(p)$ time steps. 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—here we may use $\mathcal{CH}(t_{i-1})$—and compute new time stamps for the points in $Q(t_i)$.

To implement this algorithm we use an array A. $A[t_i]$ points to a list that contains the points whose time stamps expire at time t_i. We actually work with an array $A[0..n-1]$ with n entries, and let time advance through the array in a cyclic manner (using without loss of generality that $t_i = i$). We bound the time stamps to be at most n steps, and we use an approximation of $\text{dist}(p(t), \partial\mathcal{CH}(t))$ 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 foreach $p \in Q(t)$ do
4     Compute a lower bound $d_p^*$ on $\text{dist}(p(t), \partial\mathcal{CH}(t))$.
5     $t(p) \leftarrow \min(1 + \lfloor \frac{d_p^*}{2d_{\max}} \rfloor, n)$
6     Add $p$ to $A[(t + t(p)) \mod n]$.
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—see Figure 2. 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 $q_{\text{above}}, q_{\text{below}}, q_{\text{left}}$, and q_{right}. Then we set $d_p^* = q_{\min}/\sqrt{2}$ (note that $d_p^* \leqslant \text{dist}(p, \partial\mathcal{CH}(t)) \leqslant \sqrt{2}\, d_p^*$). We get the following result.

LEMMA 4. *At each time step t,* UPDATECH *updates the convex hull in $O(|Q(t)| \log n)$ time.*

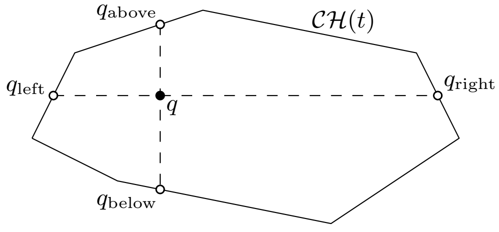

Figure 2: Points straight above, below, left and right of q are used to approximate the minimum distance from q to $\partial\mathcal{CH}(t)$.

Below we describe a more efficient version of the algorithm for large k-spread, but first we analyze the number of time stamps expiring in each time step.

Analysis of 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 prove a bound on the number of convex-hull vertices.

LEMMA 5. *The number of vertices of the convex hull $\mathcal{CH}(P)$ of a point set P is $O(k\Delta_k)$.*

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

In the worst case all time stamps expire in a single time step. However, using an amortization argument we show that on average only $O(k\Delta_k \log n)$ points expire in each time step.

LEMMA 6. *The amortized number of time stamps expiring in each time step is $O(k\Delta_k \log n)$.*

PROOF. 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. To prove the lemma we need to devise a scheme such that (i) a point always has at least 1 euro in its account when its time stamp expires, and (ii) the total amount of money handed out at each time step is $O(k\Delta_k \log \Delta_k)$. Our scheme is that at time step t_i each point p receives

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

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

To prove (i), 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

$$t(p) = 1 + \left\lfloor \frac{d_p^*(t_j)}{2d_{\max}} \right\rfloor \geqslant 1 + \left\lfloor \frac{d_p(t_j)}{2\sqrt{2} \cdot d_{\max}} \right\rfloor \geqslant \frac{d_p(t_j)}{2\sqrt{2} \cdot d_{\max}}.$$

The amount of money received by p from t_j up to t_{i-1} is thus at least

$$t(p) \cdot \frac{8\sqrt{2} \cdot d_{\max}}{\max_{t_j \leqslant t \leqslant t_{i-1}} d_p(t)} \geqslant \frac{4d_p(t_j)}{\max_{t_j \leqslant t \leqslant t_{i-1}} d_p(t)}$$

The distance between p and $\mathcal{CH}(P)$ increases (or decreases) by at most $2d_{\max}$ at each time step, so during at any time $t_j \leqslant t \leqslant t_{i-1}$ the distance from p to $\partial\mathcal{CH}(P)$ is at most

$$\begin{aligned} d_p(t_j) + t(p) \cdot 2d_{\max} &\leqslant d_p(t_j) + \left(1 + \frac{d_p(t_j)}{2d_{\max}}\right) \cdot 2d_{\max} \\ &= 2 \cdot d_p(t_j) + 2d_{\max} \leqslant 4 \cdot d_p(t_j), \end{aligned}$$

where in the last step we assumed that $d_p(t_j) \geqslant d_{\max}$ (since otherwise p already receives at least 1 euro at time t_j). Hence the total amount of money received by p is at least

$$\frac{4d_p(t_j)}{\max_{t_j \leqslant t \leqslant t_{i-1}} d_p(t)} \geqslant \frac{4d_p(t_j)}{4 \cdot d_p(t_j)} = 1$$

To prove (ii), 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 $G_1, \ldots, G_\ell$. 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 $\ell = 8\sqrt{2}n$. All points from G_j lie in an annulus of diameter $O(\Delta_k \cdot \text{mindist}(P(t_i)))$ where the distance between the inner and outer boundary of the annulus is $\Theta(d_{\max})$. Using a simple packing argument and Lemma 1 it follows that such an annulus contains $O(k\Delta_k)$ points. Hence, the total amount we have to pay to all points in a single group G_j is

$$O(k\Delta_k) \cdot \min\left(1, \frac{8\sqrt{2} \cdot d_{\max}}{(j-1) \cdot d_{\max}}\right) = O\left(\frac{k\Delta_k}{j}\right) \text{ euros.}$$

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

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

□

Lemma 4 and 6 imply the following theorem.

Theorem 1. *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 $O(k\Delta_k \log^2 n)$ time amortized per time step, where Δ_k is the maximum k-spread of P at any time.*

A linear-time algorithm without spread assumption. Next we show how the convex hull of a point set with a high k-spread can still be maintained in near-linear time per step, under the Displacement Assumption. Consider a partitioning of the plane into vertical columns of width $d_{\max}$. We maintain a left-to-right ordered list $\mathcal{L}_{\text{col}}$ of the columns that contain at least one point from P. For each column C we also maintain the set $P(C)$ of points inside that column. By the Displacement Assumption each point can either stay in its column or move to a neighboring column in a single time step. Hence we can update $\mathcal{L}_{\text{col}}$ and and the sets $P(C)$ in $O(1)$ time per point at each time step. We also maintain a bottom-to-top sorted list $\mathcal{L}_{\text{row}}$ of the rows with height $d_{\max}$ that contain at least one point from P.

After we have updated $\mathcal{L}_{\text{col}}$ and the sets $P(C)$ at time t each set $P(C)$ contains those points that are in C at time t. Now suppose we want to compute $\mathcal{CH}(t)$ from $\mathcal{CH}(t-1)$. We divide $\mathcal{CH}(t-1)$ into four sections using points $p_{\text{nw}}, p_{\text{ne}}, p_{\text{sw}}, p_{\text{se}}$. The points p_{nw} and p_{ne} are vertices on the upper boundary of $\mathcal{CH}(t-1)$ that have tangent lines with a slope of 1 and -1 respectively. The points p_{sw} and p_{se} are on the lower boundary of $\mathcal{CH}(t-1)$ and they have tangent lines with a slope of -1 and 1 respectively (see Figure 3). We focus on the northern section $\mathcal{CH}_{\text{n}}(t-1)$ of the convex hull between p_{nw} and p_{ne}.

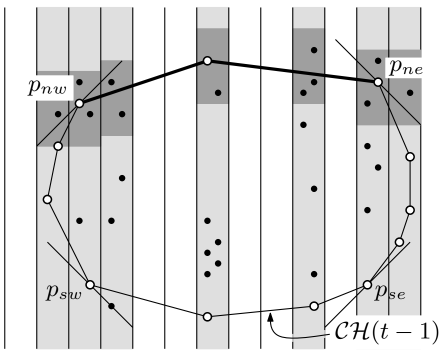

Figure 3: Potential vertices of the northern section of $\mathcal{CH}(t)$ must be in the dark gray regions.

We want to find all points of $P(t)$ that are within distance $d_{\max}$ of the northern section of the convex hull. We inspect the columns of $\mathcal{L}_{\text{col}}$ from left to right. For each column C we see if it is intersected by $\mathcal{CH}_{\text{n}}(t-1)$. If this is the case we find the lowest and highest y-coordinate of this intersection, denoted by $y_\ell(C)$ and $y_h(C)$ respectively. Now for each point $p(t) \in P(C)$ we test if its y-coordinate $y(p(t))$ is greater than $y_\ell(C) - 2d_{\max}$. We define $P^*(C) \subseteq P(C)$ as the set of points which satisfy this criterion. The slope of edges of $\mathcal{CH}_{\text{n}}(t-1)$ is between 1 and -1, which guarantees that all points of $P(C)$ that are within distance $d_{\max}$ of $\mathcal{CH}_{\text{n}}(t-1)$ are in $P^*(C)$. Note that using $y_l(C) - d_{\max}$ as bound on the y-coordinate is not sufficient, since a point $s(t) \in C$ may have a shorter distance to the part of $\mathcal{CH}_{\text{n}}(t-1)$ in a neighboring column (see Figure 4). Additionally we inspect the columns C_{nw} and C_{ne} which are to the left of the column containing p_{nw} and right of the column containing p_{ne} as shown in Figure 3. For C_{nw} we find the set $P^*(C_{\text{nw}})$ of points which have a y-coordinate greater than $y(p_{\text{nw}}) - 2d_{\max}$.

Every point $p(t) \in P(t)$ for which $\text{dist}(p(t), \mathcal{CH}_{\text{n}}(t-1)) \leqslant d_{\max}$ is in the set $P^*(C)$ for some column C. The points $P^*(C)$ all have a y-coordinate of at least $y_\ell(C) - 2d_{\max}$. Also no point in $P(C)$ can have a y-coordinate of more than $y_h + 2d_{\max} \leqslant y_\ell(C) + 3d_{\max}$ since a point $p(t)$ can be no more then $d_{\max}$ away from $\mathcal{CH}(t-1)$. For every column C the points of $P^*(C)$ are contained in a $d_{\max} \times 5d_{\max}$ rectangle and thus contain $O(k)$ points. It follows that we can sort them in $O(k \log k)$ time per column and sort the points in $\bigcup_{C \in \mathcal{L}_{\text{col}}} P^*(C)$ in $O(n \log k)$ time. It then takes $O(n)$ time to compute the convex hull of $\bigcup_{C \in \mathcal{L}_{\text{col}}} P^*(C)$.

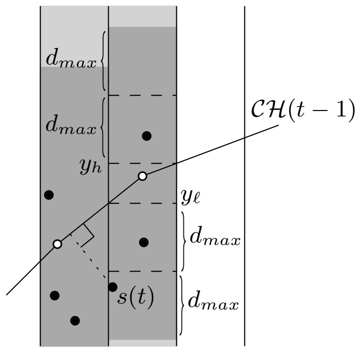

Figure 4: Potential vertices are between $y_l(C) - 2d_{\max}$ and $y_h + 2d_{\max}$.

In a similar fashion we compute the convex hulls for the points that are within distance $d_{\max}$ of the eastern, southern or western part of $\mathcal{CH}(t-1)$. We then merge the four convex hulls in $O(n)$ time to obtain $\mathcal{CH}(t)$.

Theorem 2. *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 $O(n \log k)$ time per time step.*

3.1 Convex Hull in Higher Dimensions

The algorithm we described to update the convex hull after each time step under the spread assumption generalizes to higher dimensions. We follow the steps of Algorithm UpdateCH. The array A again holds pointers to lists of points that expire at a certain time step. Computing the convex hull of $Q(t)$ in line 2 can be done using the output sensitive algorithm by Chan [6]. To compute new time stamps we again find a lower bound on the distance to the boundary of the convex hull. We do this by shooting axis-aligned rays in all $2d$ possible directions using Chans algorithm [6] for ray-shooting queries. The minimum q_{min} of the distances obtained from the ray-shooting queries bounds the minimum distance $d_{p(t)}$ from a point $p(t)$ to $\partial\mathcal{CH}(t-1)$ in d dimensions:

$$q_{min}/\sqrt{d} \leqslant d_{p(t)} \leqslant q_{min}.$$

For the number of expired points in a single time step we use a similar amortization scheme where each point $p(t)$ gets

$$\min\left(1, \max\left(\frac{1}{n}, \frac{8\sqrt{d}\ d_{\max}}{d_{p(t)}}\right)\right) \text{ euros.}$$

Combining this we get the following theorem.

Theorem 3. *Under the Displacement Assumption, the convex hull of a set P of n points moving in $\mathbb{R}^d$ can be maintained in the black-box model in*

$$O((k\Delta_k^{d-1}\log\Delta_k)^{\lfloor d/2\rfloor}\log^{O(1)}k\Delta_k)$$

time amortized per step, where Δ_k is the maximum k-spread of P at any time.

When the spread is optimal—$\Delta_k = O(n^{1/d})$ and $k = O(1)$—this is slightly faster than rebuilding from scratch in three dimensions; it even runs in sublinear time. In higher dimensions Chans output-sensitive algorithm [6] is slightly faster. The reason for this is that when for $d \geqslant 4$ the complexity of the convex hull is $\Omega(n)$, hence we do not benefit from using time stamps to inspect only a small number of points.

4. DELAUNAY TRIANGULATION

We denote the Delaunay triangulation of a point set P by $\mathcal{DT}(P)$, and we use $\mathcal{DT}(t)$ as a shorthand for $\mathcal{DT}(P(t))$. We describe two algorithms to recompute $\mathcal{DT}(t)$ given $\mathcal{DT}(t-1)$. Both algorithms use the same global approach: they update the position of each point in turn (and change the Delaunay triangulation accordingly). Let $P = \{p_1, \ldots, p_n\}$ and define $Q_i = \{p_1(t), \ldots, p_i(t), p_{i+1}(t-1), \ldots p_n(t-1)\}$. Thus $Q_0 = P(t-1)$ and $Q_n = P(t)$. Updating the position of p_i now means computing $\mathcal{DT}(Q_i)$ from $\mathcal{DT}(Q_{i-1})$. This can be done in two ways.

- MoveAndFlip: The point $p_i(t-1)$ is moved along a straight line to its new position $p_i(t)$ while maintaining the Delaunay triangulation. For each point we follow the traditional KDS approach. We compute all in-circle certificates that involve p_i and sort them by their failure times—the failure time of a certificate is the position along the line $p_i(t-1)p_i(t)$ where the certificate becomes false. Each time a certificate fails we flip an edge of the Delaunay triangulation and update the collection of certificates.
- InsertAndDelete: We first walk in $\mathcal{DT}(Q_{i-1})$ from $p_i(t-1)$ to $p_i(t)$ to determine the triangle τ of $\mathcal{DT}(Q_{i-1})$ containing $p_i(t)$. Then we insert $p_i(t)$ into $\mathcal{DT}(Q_{i-1})$ in the usual way, namely by adding edges from $p_i(t)$ to the vertices of τ and then perform edge flips until we have the Delaunay triangulation of $Q_{i-1} \cup \{p_i(t)\}$ [4]. Finally, we delete $p_i(t-1)$ using the algorithm by Devillers [9] or we delete $p_i(t-1)$ with all its edges and then use the algorithm by Chin *et al.* [7] to repair the Delaunay triangulation.

Let $D(p, r)$ denote the disk of radius r centered at point p. Note that $D(p_i(t-1), d_{\max})$ is exactly the region reachable from $p_i(t-1)$ in one time step. Now consider two points $p_j, p_\ell \in P$. We say that $p_j p_\ell$ is a *potential edge* at time t if there is a placement of each point $p_i \in P$ somewhere in its disk $D(p_i(t-1), d_{\max})$ such that $p_j p_\ell$ is an edge in the Delaunay triangulation of the resulting point set. We then call p_j and p_ℓ *potential neighbors*. Theorem 6 below states the number of potential edges is $O(k^2\Delta_k^2)$. With this result in hand, we can analyze our update algorithms.

Theorem 4. *Computing $\mathcal{DT}(t)$ from $\mathcal{DT}(t-1)$ using* MoveAndFlip *requires $O(k^2\Delta_k^2)$ flips in total and takes $O(k^2\Delta_k^2\log n)$ time.*

Proof. Consider the movement of p_i from $p_i(t-1)$ to $p_i(t)$. A flip occurs when p_i becomes co-circular with three other points, say a, b, c, and $\text{circ}(a, b, c)$, the circle defined by a, b, c, is empty. Next we observe that each point a, b, c must form a potential edge with p_i: at the time of the flip p_i is on $\text{circ}(a, b, c)$, which contains no other points, hence ap_i, bp_i and cp_i are edges of the Delaunay triangulation at that time. Let $\text{E}(p_i)$ denote the set of potential edges with p_i as an endpoint, and let $N(p_i) = \{q \in Q_i \mid qp_i \in \text{E}(p_i)\}$ be the set of potential neighbors of p_i. The number of empty circles defined by $N(p_i)$ is equal to the number of Delaunay triangles in $\mathcal{DT}(N(p_i))$, which is $O(|N(p_i)|) = O(|\text{E}(p_i)|)$. Using Theorem 6 we conclude that the total number of flips for moving all points $p_i \in P$ is

$$\sum_{p_i \in P} O(|\text{E}(p_i)|) = O(k^2\Delta_k^2).$$

At each flip we can update the Delaunay triangulation in $O(1)$ time. We must also update the event queue storing the certificate failure times, which takes $O(\log n)$ time since we have to delete and insert only $O(1)$ certificates into the queue. The running time thus becomes $O(k^2\Delta_k^2\log n)$. □

The $O(\log n)$ factor in the running time for the MoveAndFlip approach stems from the event queue on the certificates. The InsertAndDelete approach avoids this factor.

Theorem 5. *Computing $\mathcal{DT}(t)$ from $\mathcal{DT}(t-1)$ using* InsertAndDelete *takes $O(k^2\Delta_k^2)$ time.*

Proof. Consider the update of the position of p_i. Note that whenever we cross an edge $p_j p_\ell$ during the walk with p_i, then $p_j p_\ell$ would be part of a flip in the MoveAndFlip strategy. This implies that the number of edges crossed is at most linear in the number of potential edges. Moreover, inserting $p_i(t)$ takes time linear in the degree of $p_i(t)$ in $\mathcal{DT}(Q_{i-1} \cup \{p_i(t)\})$ [4] and deletion of $p_i(t-1)$ takes linear time in the degree of $p_i(t-1)$ [7]. Overall, updating the position of p_i takes linear time in the number of potential edges involving p_i, so the total time is $O(k^2\Delta_k^2)$ by Theorem 6. The algorithm by Chin *et al.* [7] is somewhat difficult and

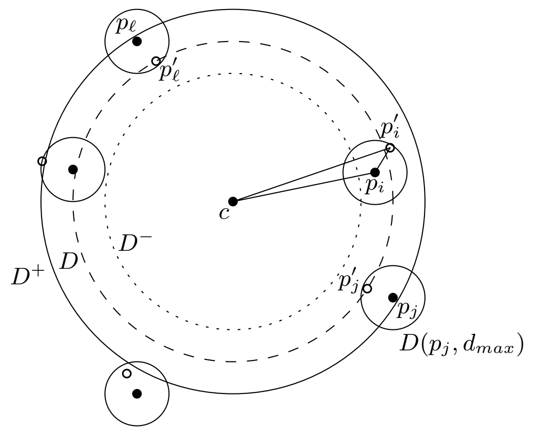

Figure 5: $p_j p_\ell$ is a potential edge only if a disk D^- exists that has p_j and p_ℓ within distance $2d_{\max}$ and does not contain any other points of P.

might be slow when deleting points with a small degree. In that case it may be more efficient to use the asymptotically slower, but simpler algorithm by Devillers [9]. ☐

Analysis of the number of potential edges. Potential edges are defined on the point set $P(t-1)$ at a single time step, so we drop the time parameter and use $P = \{p_1 \dots p_n\}$ to denote the set of points we are dealing with. We also define $\text{mindist}_k := \text{mindist}_k(P)$ and $\Delta_k := \Delta_k(P)$. (The latter is a slight abuse of notation as in fact Δ_k was defined as an upper bound on the k-spread at any time.) We first give a necessary condition for two points p_j and p_ℓ to form a potential edge. We call a disk *empty* if its interior does not contain any point from P. The following lemma follows from the empty-disk property of Delaunay triangulations and the fact that points move by at most $d_{\max}$ in a single time step.

LEMMA 7. *If $p_j p_\ell$ is a potential edge then there is an empty disk D^- such that p_j and p_ℓ are within distance $2d_{\max}$ of D^-.*

PROOF. Assume that $p_j p_\ell$ is a potential edge. Then there is a set of points $P' = \{p'_1 \dots p'_n\}$, where $p'_i \in D(p_i, d_{\max})$ for all i, such that $p'_j p'_\ell$ is an edge in $\mathcal{DT}(P')$. Let $D = D(c, r)$ denote an empty disk with p'_j and p'_ℓ on its boundary. For now assume $r > d_{\max}$, and let $D^- = D(c, r - d_{\max})$; see Figure 5. For any point p'_i, we know that $|cp'_i| \geqslant r$ and $|p_i p'_i| \leqslant d_{\max}$. It follows from the triangle inequality that $|cp_i| \geqslant r - d_{\max}$ and that p_i cannot be inside D^-.

For p'_j it holds that $|cp'_j| = r$ and $|p_j p'_j| \leqslant d_{\max}$. By the triangle inequality we get that $|cp_j| \leqslant r + d_{\max}$. The same holds for p_ℓ, which proves that $p_j p_\ell$ can be a potential edge only if there is a disk D^- that does not contain any other points of P but has p_j and p_ℓ within distance $2d_{\max}$ of its boundary. It remains to consider the case $r \leqslant d_{\max}$. Then $|p_j p_\ell| \leqslant 4d_{\max}$. The overlap region of $D(p_j, 2d_{\max})$ and $D(p_\ell, 2d_{\max})$ is non-empty. If the overlap has positive area then we can place a (possibly very small) disk D inside $D(p_j, 2d_{\max}) \cap D(p_\ell, 2d_{\max})$ that does not contain any points from P and, hence, satisfies the conditions of the lemma. If the overlap is a single point q—note that this point could happen to be a point in P—then we can place a small empty disk touching q and satisfying the conditions of the lemma. ☐

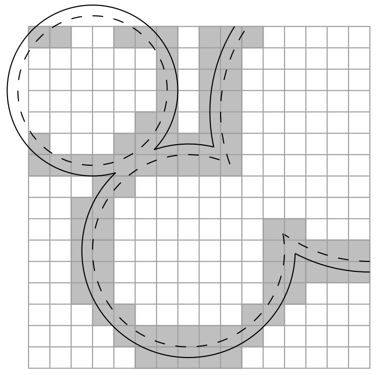

Figure 6: Points in $U^+ \backslash U$ must be in cells intersected by ∂U^+ or ∂U.

Let E_{pot} denote the set of potential edges defined by P. For each potential edge $p_j p_\ell$ we pick a disk as in Lemma 7, which we call its *witness disk*. We split the set of witness disks into three subsets based on the size of the disks:

- $\mathcal{D}_{\text{S}}$: the *small* witness disks, which have radius at most $16 \cdot \text{mindist}_k$,
- $\mathcal{D}_{\text{M}}$: the *medium-size* witness disks, which have radius between $16 \cdot \text{mindist}_k$ and $\text{diam}(P)$,
- $\mathcal{D}_{\text{L}}$: the *large* witness disks, which have radius larger than $\text{diam}(P)$.

The number of potential edges contributed by disks in $\mathcal{D}_{\text{S}}$ is easy to bound: if a potential edge pq has a small witness disk, then q must lie within $O(\text{mindist}_k)$ distance of p, and so by Lemma 1 there are only $O(k)$ such points for any point p.

LEMMA 8. *The number of potential edges contributed by the witness disks in $\mathcal{D}_{\text{S}}$ is $O(kn)$.*

To prove bounds for $\mathcal{D}_{\text{L}}$ and $\mathcal{D}_{\text{M}}$ we need the following lemma. For a square σ, let $\text{size}(\sigma)$ denote its edge length and let $\text{union}(\mathcal{D})$ denote the union of a set $\mathcal{D}$ of disks.

LEMMA 9. *Let σ be a square with edge length* $\text{size}(\sigma)$ *and let $P_\sigma = P \cap \sigma$. Let $\mathcal{D}$ be a set of disks with radius at least* $\text{size}(\sigma)/4$ *that do not contain any points of P_σ. Then the number of points in P_σ within distance $2d_{\max}$ of* $\text{union}(\mathcal{D})$ *is* $O(k \cdot \text{size}(\sigma)/\text{mindist}_k)$.

PROOF. Define $\mathcal{D}^+ := \{D(c, r + 2d_{\max}) : D(c, r) \in \mathcal{D}\}$. Set $U := \text{union}(\mathcal{D})$ and $U^+ := \text{union}(\mathcal{D}^+)$. Since the disks in $\mathcal{D}$ are empty, all the points of P_σ that are within distance $2d_{\max}$ of $\text{union}(\mathcal{D})$ lie in $U^+ \setminus U$. We overlay the square σ by a grid whose cells have size $4 \cdot \text{mindist}_k$. Because $d_{\max} \leqslant \text{mindist}_k$, each grid cell intersecting $U^+ \setminus U$ must intersect ∂U^+ or ∂U (or both); see Figure 6. Since the grid cells have size $4 \cdot \text{mindist}_k$ it follows from Lemma 1 that they each contain $O(k)$ points. It remains to prove that the number of grid cells intersected by ∂U^+ or ∂U is $O(\text{size}(\sigma)/\text{mindist}_k)$.

We show how to count the cells intersecting ∂U^+; the cells intersecting ∂U can be counted similarly. We split the boundary of each disk $D^+ \in \mathcal{D}^+$ into a *left arc*, *right arc*, *top arc*, and *bottom arc* at the points where the tangent lines have slope $+1$ and -1. The boundary ∂U^+ consists of parts of these arcs. We count the cells intersecting ∂U^+ separately for each type of arc. If a row is intersected by a left arc α of some disk D^+ then α intersects at most two[1] cells, say C and C', in that row. The $\Theta(\text{size}(\sigma)/\text{mindist}_k)$ cells immediately to the right of C, C' (if these cells exist) are contained in the interior of D^+. These cells cannot intersect another left arc on ∂U^+. Hence, in each row of the grid there are only $O(1)$ cells intersecting a left arc on ∂U^+. The total number of grid cells intersecting a left arc on ∂U^+ is therefore proportional to the number of rows, which is $O(\text{size}(\sigma)/\text{mindist}_k)$. For the right, top, and bottom arcs we can use a similar argument. $\square$

We can now prove that the large witness disks contribute $O(k^2\Delta_k^2)$ potential edges.

Lemma 10. *The number of potential edges contributed by the witness disks in $\mathcal{D}_\text{L}$ is $O(k^2\Delta_k^2)$.*

Proof. Obviously, P is contained inside a $\text{diam}(P) \times \text{diam}(P)$ square. Because the disks in $\mathcal{D}_\text{L}$ have radius at least $\text{diam}(P) = \Delta_k \cdot \text{mindist}_k$ we can apply Lemma 9 to conclude there are only $O(k\Delta_k)$ points within distance $2d_\text{max}$ of $\text{union}(\mathcal{D}_\text{L})$. These points define $O(k^2\Delta_k^2)$ potential edges. $\square$

It remains to bound the number of potential edges contributed by the medium-size disks. For $2 \leqslant i \leqslant \log_4 \Delta_k$, define

$$\mathcal{D}_\text{M}^i := \{D \in \mathcal{D}_\text{M} : 4^i \cdot \text{mindist}_k \leqslant \text{radius}(D) < 4^{i+1} \cdot \text{mindist}_k\}.$$

We bound the number of potential edges contributed by a subset $\mathcal{D}_\text{M}^i$ in terms of the area of $\text{union}(\mathcal{D}_\text{M}^i)$.

Lemma 11. *The number of potential edges contributed by witness disks in $\mathcal{D}_\text{M}^i$ is $O(k^2 \frac{A_i}{\text{mindist}_k^2})$, where A_i is the area of $\text{union}(\mathcal{D}_\text{M}^i)$.*

Proof. We overlay the plane with a grid whose cells have size $4^{i+1} \cdot \text{mindist}_k$. Any two points defining a potential edge with a witness disk in $\mathcal{D}_\text{M}^i$ have distance at most

$$4^{i+1} \cdot \text{mindist}_k + 4d_\text{max} \leqslant 4^{i+2} \cdot \text{mindist}_k$$

from each other, and so the points in any grid cell can form potential edges with points in only $O(1)$ other cells. Lemma 9 implies that in any cell only $O(k4^{i+1})$ points are within distance $2d_\text{max}$ of $\text{union}(\mathcal{D}_\text{M}^i)$. Hence in total the points in any cell C can contribute only $O((k4^{i+1})^2) = O(k^2 4^{2i})$ potential edges with witness disks from $\mathcal{D}_\text{M}^i$. Now we just have to count the number of cells within distance $2d_\text{max}$ of $\text{union}(\mathcal{D}_\text{M}^i)$. Let

$$(\mathcal{D}_\text{M}^i)^+ := \{D(c, r + 2d_\text{max}) : D(c,r) \in \mathcal{D}_\text{M}^i\}$$

and set $U_i^+ := \text{union}((\mathcal{D}_\text{M}^i)^+)$. Note that the area of U_i^+ is $O(A_i)$. We need to count the number of cells intersecting U_i^+. Each cell intersecting U_i^+ either contains at least $1/4$ of a disk with radius at least 4^imindist_k or it is adjacent to such a cell. Hence, the total number of intersected cells is bounded by

$$O(\text{area}(U_i^+)/\text{area of one cell}) = O(A_i/(4^{i+1} \cdot \text{mindist}_k)^2).$$

We already showed that the points in any given cell contribute $O(k^2 4^{2i})$ potential edges in total, so the total number of potential edges is $O(k^2 A_i/\text{mindist}_k^2)$. $\square$

[1]If α has an endpoint in the row, we need to argue a little more carefully. We can show that there are now $O(1)$ cells intersected, rather than two.

Recall that $i \leqslant \log_4 \Delta_k$. Since $\text{diam}(P) = \Delta_k \cdot \text{mindist}_k$, we know that the diameter of $\text{union}(\mathcal{D}_\text{M}^i)$ is at most

$$\Delta_k \cdot \text{mindist}_k + 4^{i+1} \cdot \text{mindist}_k = O(\Delta_k \cdot \text{mindist}_k).$$

Hence, $A_i = O(\Delta_k^2 \cdot \text{mindist}_k^2)$, and so Lemma 11 implies that $\mathcal{D}_\text{M}^i$ contributes $O(k^2\Delta_k^2)$ potential edges. Since the number of subsets $\mathcal{D}_\text{M}^i$ is $O(\log \Delta_k)$, It follows that the total number of potential edges contributed by medium-size disks is $O(k^2\Delta_k^2 \log \Delta_k)$.

We can get rid of the $O(\log \Delta_k)$ factor by not considering each subset $\mathcal{D}_\text{M}^i$ in isolation, but also considering their interaction. From Lemma 11 we know that the set $\mathcal{D}_\text{M}^i$ can only contribute many edges if $A_i = \text{area}(U_i)$ is large, where $U_i = \text{union}(\mathcal{D}_\text{M}^i)$. In the next lemma we show that A_i can't be large for all i.

Lemma 12. *There is a region $V_i \subseteq U_i$ such that, for all $j \leqslant i-2$, $\text{union}(U_j)$ is disjoint from V_i and this region has area at least $A_i/\gamma = \Theta(A_i)$, for some fixed constant $\gamma > 1$.*

Proof. Consider a disk $D \in \mathcal{D}_\text{M}^i$ with center c and radius r. Set

$$r^- := r - 4^{i-1} \cdot \text{mindist}_k - 2d_\text{max}$$

and define $D^- = D(c, r^-)$. Then D^- must be disjoint from any disk in $\mathcal{D}_\text{M}^j$ for $j \leqslant i-2$. Indeed, any disk of $\mathcal{D}_\text{M}^j$ has radius at most $4^{i-1} \cdot \text{mindist}_k$, so if it were to intersect D^- it would be completely contained in $D(c, r - 2d_\text{max})$. This means that all points are at least distance $2d_\text{max}$ away from it, and so it cannot be a witness disk of a potential edge.

Note that the radius r^- of any inner disk D^- is

$$r^- = r - 4^{i-1} \cdot \text{mindist}_k - 2d_\text{max} \geqslant r - (4^{i-1} + 2) \cdot \text{mindist}_k.$$

We overlay U_i with a grid whose cells have size $(4^{i-2}) \cdot \text{mindist}_k$. Then any inner disk D^- contains at least one grid cell and D itself intersects a constant number of cells (see Figure 7). Hence, we can charge any cell intersecting a disk D of $\mathcal{D}_\text{M}^i$ to a cell that is completely inside some inner disk D^- in such a way that we charge only a constant number of cells to each cell in an inner disk. It follows that

$$A_i = O(\text{area of the union of the inner disks}),$$

which proves the claim. $\square$

With this result we prove that the total number of potential edges contributed by $\mathcal{D}_\text{M}$ is $O(k^2\Delta_k^2)$.

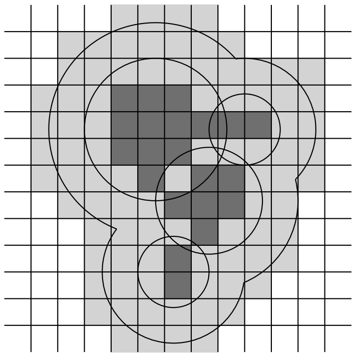

Figure 7: A disk intersects only a constant number of gridcells (light gray) and has at least one grid cell completely contained in its inner area (dark gray).

LEMMA 13. *The number of potential edges contributed by the witness disks in $\mathcal{D}_{\mathrm{M}}$ is $O(k^2\Delta_k^2)$.*

PROOF. We prove the bound for the subsets for which i is even; the proof for the subsets with i odd is similar. Consider the subsets $\mathcal{D}_{\mathrm{M}}^i$, for even i, in order of decreasing i. Let A_i^* be the area that it still available for the disks in $\mathcal{D}_{\mathrm{M}}^i$ after considering all disks from $\mathcal{D}_{\mathrm{M}}^j$ where $j \geqslant i+2$. Thus $A_i \leqslant A_i^*$. Lemma 12 implies that $A_{i-2}^* \leqslant A_i^* - (1/\gamma)A_i$ for some constant $\gamma > 1$. In other words, $A_i \leqslant \gamma(A_i^* - A_{i-2}^*)$. If we define $j_{\max} = \log_4 \Delta_k/2$ then using Lemma 11 we can bound the contribution to the number of potential edges for $\mathcal{D}_{\mathrm{M}}^i$, with i even, as follows.

$$\sum_{j=1}^{j_{\max}} O(k^2 \frac{A_{2j}}{\mathrm{mindist}_k^2}) = O\left(\frac{k^2}{\mathrm{mindist}_k^2}\sum_{j=1}^{j_{\max}}\gamma(A_{2j}^* - A_{2j-2}^*)\right)$$

$$= O\left(\frac{k^2}{\mathrm{mindist}_k^2}\gamma(A_{2j_{\max}}^* - A_0^*)\right)$$

$$= O(k^2\Delta_k^2)$$

The last step follows from the fact that

$$A_{2j_{\max}}^* = O(\mathrm{diam}(P)^2) = O(\Delta_k^2 \cdot \mathrm{mindist}_k^2).$$

□

From Lemma's 8, 10 and 13 we conclude the following.

THEOREM 6. *Let P be a set of n points. Under the Displacement Assumption, the number of potential edges defined by P is $O(\min(k^2\Delta_k^2, n^2))$.*

We can show this bound is tight in the worst case. One could hope that only a smaller number of edges may actually occur during movement, but even this is not the case: Next we show that $\Omega(k^2\Delta_k^2)$ edges can occur when moving the points one at a time.

THEOREM 7. *For large enough n, and $k \geqslant 1$ and $\Delta_k \geqslant \sqrt{8n}$, there is a set P of n points with a k-spread of Δ_k at time $t-1$ such that computing $\mathcal{DT}(P(t))$ from $\mathcal{DT}(P(t-1))$ under the Displacement Assumption takes $\Omega(k^2\Delta_k^2)$ time using the* MOVEANDFLIP *or* INSERTANDDELETE *approach if the points are moved in a bad order.*

PROOF. Consider the following set $P = X \cup Y \cup Z$. The points in Y and Z will generate the desired $\Omega(k^2\Delta_k^2)$ lower-bound, whereas X contains leftover points which are placed on a grid and do not move to ensure the spread assumption

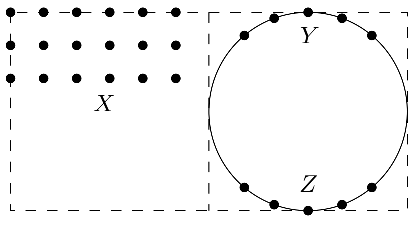

Figure 8: We need $\Omega(k^2\Delta_k^2)$ time to update the Delaunay triangulation when points of Y move in a bad order.

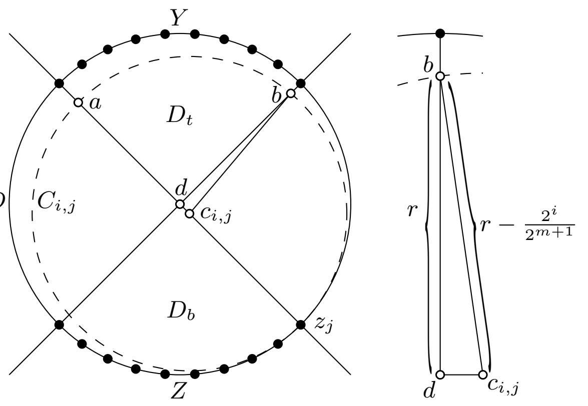

Figure 9: The point of Y and Z are in the top and bottom wedges respectively.

is not violated. Without loss of generality we assume that $\mathrm{mindist}_k = 1$. The points of X are placed in a square of height and width $\Delta_k/(2\sqrt{2})$ and the points of $Y(t-1)$ and $Z(t-1)$ are placed in a similar square (see Figure 8).

We place the points of Y and Z at time $t-1$ along the boundary of a disk $D = D(d, r)$ as follows. We divide D into four sections using lines with a slope of 1 and -1 through the center d of D. Points of $Y(t-1)$ are along the boundary of the top section D_t and points of $Z(t-1)$ along the boundary of the bottom section D_b (see Figure 9). The points of Y will move into D; whenever a point $y \in Y$ has moved it will have an edge with every point of Z in the Delaunay triangulation. Let $y_1 \ldots y_m$ be the points of Y in the order in which we move them and $z_1 \ldots z_m$ the points of Z, which do not move. Let P_i be the point set P between time $t-1$ and t just after y_i has moved:

$$P_i := \{y_1(t)\ldots y_i(t), y_{i+1}(t-1)\ldots y_m(t-1)\} \cup Z \cup X$$

Note that the time parameter for Z and X is omitted as points in these sets do not move.

Each point $y_i(t-1)$ is moved towards d by a distance $2\frac{2^i}{2^{m+1}} = \frac{2^{i+1}}{2^{m+1}} = 2^{-(m-i)}$. Let $C_{i,j}$ be the disk inside D and tangent to D in z_j with radius $r - 2^{-(m-i+1)}$ and let $c_{i,j}$ denote the center of $C_{i,j}$. For every point z_j the point $y_i(t)$ will be contained in $C_{i,j}$.

For $y_i(t)$ and z_j to form an edge in the Delaunay triangulation there has to be a disk that contains $y_i(t)$ and z_j, but no other points of P_i. No point of Z, other than z_j can be inside $C_{i,j}$ as all points of Z are on the boundary of D and $C_{i,j}$ only intersects D in z_j. For each point in $P_i\backslash\{y_i\}$ it holds that the distance to d is at least $r - 2^{-(m-i+1)}$. Now we claim (and will prove later) that every point inside $C_{i,j}$ that is in the top section of D has at most distance $\sqrt{r^2 - r2^{-(m-i)}}$ to d. Since

$$\sqrt{r^2 - r2^{-(m-i)}} \leqslant r - 2^{-(m-i+1)}$$

it follows that no point of $P_i\backslash\{y_i\}$ is contained in $C_{i,j}$. Therefor there is an edge between $y_i(t)$ and z_j in $\mathcal{DT}(P_i)$.

In this manner each point $y_i(t)$ has edges to all points of Q in $\mathcal{DT}(P_i)$. Since both Y and Z contain $\Omega(k\Delta_k)$ points, the total number of edges created is $\Omega(k^2\Delta_k^2)$.

What remains to be proven is that the distance between any point in $C_{i,j} \cap D_t$ and the center d of D is at most $\sqrt{r^2 - r2^{-(m-i)}}$.

The shortest distance from the boundary of $C_{i,j}$ is achieved by the point a on the line through z_j and d and by construction this point is in D_t. If we go clockwise or counterclockwise along the boundary of $C_{i,j}$ the distance to d only becomes larger until we reach z_j. The worst case situation arises for the point b on one of the bounding edges of D_t (see Figure 9). Here the longest distance from the boundary of $C_{i,j}$ to d is

$$\sqrt{(r - 2^{-(m-i+1)})^2 - (2^{-(m-i+1)})^2} = \sqrt{r^2 - r2^{-(m-i)}}.$$

$\square$

5. CONCLUSION

We presented algorithms for maintaining the convex hull and the Delaunay triangulation of a planar point set P in the KDS black-box model. The algorithms are simple and do not require knowledge of the k-spread $\Delta_k(P)$ or k: the convex-hull algorithms needs to know only $d_{\max}$, the maximum displacement of any point in one time step, and the Delaunay-triangulation algorithm needs no knowledge at all. Our main contribution lies in the analysis of these algorithms under the Displacement Assumption and in terms of Δ_k.

For the convex-hull maintenance we spend $O(k\Delta_k \log^2 n)$ amortized time. 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 in the worst case. However, it may be possible to get rid of one logarithmic factor from the time bound by a more clever algorithm. In fact, when we can use the floor function, then we know how to to do this. Unfortunately, the algorithm needs to know $\text{mindist}_k(P(t))$, which is perhaps not 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.

For the Delaunay triangulation we have shown that a simple flipping algorithm needs $O(k^2\Delta_k^2)$ flips. The bound is based on an analysis of the number of potential edges—that is, all edges that can possibly arise in one time step. Our bound on the number of potential edges is tight in the worst case and we show that there is also an $\Omega(k^2\Delta_k^2)$ lower bound on the number of edges that appear in the worst case when moving points one at a time.

It depends on the application how realistic our model is. We expect that most sampling rates are such that the Displacement Assumption is satisfied. A valid question is whether point sets can be expected to have small k-spread. In meshing-type applications, it may be realistic to assume that the k-spread is $O(\sqrt{n})$; our results imply then that the simple flipping approach needs only $O(n)$ flips. In any case, Δ_k seems like a reasonable parameter to measure efficiency. 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 the Δ_k.

6. 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. de Berg, M.van Kreveld, M. Overmars, O. Schwartzkopf. *Computational Geometry: Algorithms and Applications.* Springer-Verlag, Berlin, Germany 3rd edition, 2008.

[5] P.M. Manhães de Castro, J. Tournois, P. Alliez, and O. Devillers. Filtering relocations on a Delaunay triangulation. In *Proc. Sympos. Geometry Processing*, pages 1465–1474, 2009

[6] T. Chan. Output-sensitive results on convex hulls, extreme points, and related problems. *Discr. Comput. Geom.* 16: 369–387 (1996).

[7] F. Chin, J. Snoeyink and C.A. Wang. Finding the medial axis of a simple polygon in linear time. *Discr. Comput. Geom.* 21:405–420 (1999).

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

[9] O. Devillers. On deletion in Delaunay triangulations. In *Proc. 15th ACM Sympos. Comput. Geom.*, pages 181–188, 1999.

[10] J. Erickson. Dense Point Sets Have Sparse Delaunay Triangulations. *Discr. Comput. Geom.* 30:83-115 (2005).

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

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

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

[14] 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.

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

[16] 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.

[17] D. Russel (2007). *Kinetic Datastructures in Practise.* Ph.D. thesis. Stanford University: U.S.A.

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

[19] K. Yi and Q. Zhang. Multi-dimensional online tracking. In *Proc. 20th ACM-SIAM Sympos. Discr. Algo.*, pages 1098–1107, 2009.