Computational Geometry 44 (2011) 191–205

Contents lists available at ScienceDirect

Computational Geometry: Theory and Applications

www.elsevier.com/locate/comgeo

A kinetic triangulation scheme for moving points in the plane ☆

Haim Kaplan [a], Natan Rubin [a,*], Micha Sharir [a,b]

[a] The Blavatnik School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel
[b] Courant Institute of Mathematical Sciences, New York University, New York, NY 10012, USA

A R T I C L E   I N F O

Article history:
Received 9 May 2010
Accepted 3 November 2010
Available online 5 November 2010
Communicated by M. de Berg

Keywords:
Triangulation
Kinetic data structure
Pseudo-triangulation
Treaps

A B S T R A C T

We present a simple randomized scheme for triangulating a set P of n points in the plane, and construct a kinetic data structure which maintains the triangulation as the points of P move continuously along piecewise algebraic trajectories of constant description complexity. Our triangulation scheme experiences an expected number of $O(n^2\beta_{s+2}(n)\log^2 n)$ discrete changes, and handles them in a manner that satisfies all the standard requirements from a kinetic data structure: compactness, efficiency, locality and responsiveness. Here s is the maximum number of times at which any specific triple of points of P can become collinear, $\beta_{s+2}(q) = \lambda_{s+2}(q)/q$, and $\lambda_{s+2}(q)$ is the maximum length of Davenport–Schinzel sequences of order $s+2$ on q symbols. Thus, compared to the previous solution of Agarwal, Wang and Yu (2006) [4], we achieve a (slightly) improved bound on the number of discrete changes in the triangulation. In addition, we believe that our scheme is conceptually simpler, and easier to implement and analyze.

1. Introduction

Let $P(t) = \{p_1(t), \dots, p_n(t)\}$ be a set of n moving points in the plane. We assume that the motions of the points are simple, in the sense that the trajectory of each point is a connected piecewise-algebraic curve of *constant description complexity*, meaning that it can be described as a Boolean combination of a constant number of polynomial equalities and inequalities of constant maximum degree.

Our goal is to devise a reasonably simple scheme for triangulating $P(t)$ at any fixed time t, and to maintain the triangulation as the points move. That is, we wish to partition the convex hull $\mathcal{CH}(P)$ of P into pairwise openly disjoint triangles whose vertices are the points of P, so that the interior of each triangle is empty—it does not contain any point of P. The scheme has to be *kinetic*, so that we can keep track of the discrete combinatorial changes that the triangulation undergoes as the points move, and update the triangulation so that it continues to conform to the underlying scheme. (That is, at any given time t the maintained triangulation coincides with the one that would result from applying the static scheme to $P(t)$.)

The study of triangulations plays a central role in computational geometry because triangulations have numerous applications in such areas as computer graphics, physical simulation, collision detection, and geographic information systems [8, 13]. See [18] for additional information. With the advancement in technology, many applications, for instance, video games,

☆ Work by Haim Kaplan and Natan Rubin was partially supported by Grant 975/06 from the Israel Science Fund. Work by Micha Sharir and Natan Rubin was partially supported by Grants 155/05 and 338/09 from the Israel Science Fund. Work by Haim Kaplan was also supported by Grant 2006-204 from the U.S.–Israel Binational Science Foundation. Work by Micha Sharir was also supported by NSF Grant CCF-08-30272, by Grant 2006-194 from the U.S.–Israel Binational Science Foundation, and by the Hermann Minkowski–MINERVA Center for Geometry at Tel Aviv University. A preliminary version of this paper appeared in *Proc. 26th Annual Symposium on Computational Geometry*, 2010, pp. 137–146.

* Corresponding author.
E-mail addresses: haimk@post.tau.ac.il (H. Kaplan), rubinnat@post.tau.ac.il (N. Rubin), michas@post.tau.ac.il (M. Sharir).

0925-7721/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.comgeo.2010.11.001

virtual reality, dynamic simulations, and robotics, call for maintaining a triangulation as the data points move. See [18] for additional information. For example, the arbitrary Eulerian–Lagrangian method [12] provides a way to integrate the motion of fluids and solids within a moving finite-element mesh.

In $\mathbb{R}^2$, the Delaunay triangulation $DT(P)$ of P produces well-shaped triangles, and it is very easy to maintain kinetically, so it is a good candidate for such a triangulation scheme. The problem, however, is that the best known upper bound on the number of discrete changes in $DT(P(t))$, as a function of time t, is only nearly cubic in n (the bound is cubic if the points move with constant velocities); see [2,15,16,22]. While it is strongly believed that the maximum possible number of discrete changes that $DT(P(t))$ can experience is only nearly quadratic in n, this is one of the hardest open problems in computational and combinatorial geometry (as recognized, e.g., in [11]). Until this conjecture is established, one seeks alternative triangulation schemes with a provable *nearly-quadratic* upper bound on the number of discrete changes. (This is nearly best possible, since the convex hull itself can change $\Omega(n^2)$ times during a simple motion of the points of P; see [22].) Moreover, the scheme should be sufficiently simple to define, to implement, and (as a secondary aesthetic virtue) to analyze. Finally, the scheme should satisfy the four basic properties of kinetic data structures [7] detailed below.

Agarwal, Wang and Yu [4] have recently presented such a randomized triangulation scheme which experiences $O(n^2 2^{\sqrt{\log n \log\log n}})$ discrete changes. Their scheme, however, is fairly complicated, and its analysis is also rather involved. It uses a hierarchy of subsets $\emptyset = R_0 \subseteq R_1 \subseteq \cdots \subseteq R_w = P$, where each set R_{i-1}, for $1 \leqslant i \leqslant w$, is a random sample of roughly $|R_i|^{1-1/i} \log n$ points of R_i. The algorithm maintains an entire hierarchy of triangulations $\emptyset = \mathcal{T}_0 \subseteq \mathcal{T}_1 \subseteq \cdots \subseteq \mathcal{T}_w$, where each $\mathcal{T}_i$ is a triangulation of R_i, so $\mathcal{T}_w$ is the desired triangulation of P; it is a refinement of $\mathcal{T}_{i-1}$ which is obtained by a suitable variant of the *fan triangulation*, introduced in [1].

1.1. Kinetic data structures

The *kinetic data structure* (KDS) framework, introduced by Basch, Guibas and Hershberger [7], proposes an algorithmic approach, together with several quality criteria, for maintaining certain geometric configurations determined by a set of objects, each moving along a trajectory whose graph, as a function of time, is a piecewise-algebraic curve (in space–time) of constant description complexity. Several interesting algorithms have been designed, using this framework, over the past decade, including algorithms for maintaining the convex hull of a set of (moving) points in the plane [7], the closest pair and all nearest neighbors in any dimension [3,7], and many other configurations. See [17] for a comprehensive, albeit old, survey, and [3] for a list of more recent results and references.

Typically, a KDS operates by maintaining a set of *certificates*. As long as they are all valid, the structure being maintained is guaranteed to be valid too. Each certificate has a (first future) failure time, and we store these critical times in an event priority queue. When a certificate fails, we repair the KDS, update, if needed, the geometric structure that we maintain, generate new certificates and insert their failure times into the queue.

Generally, a good KDS is expected to possess the following four properties: (i) *Compactness*, meaning that the storage that it requires is larger by only a polylogarithmic factor than the space required for the structure being maintained. (ii) *Efficiency*, meaning that the number of events that it processes (i.e., failure times of the certificates) is larger only by a polylogarithmic factor[1] than the maximum possible number of discrete changes in the structure being maintained. (iii) *Responsiveness*, meaning that repairing the KDS at a certificate failure event takes only polylogarithmic time. (iv) *Locality*, meaning that each input object is stored at only a polylogarithmic number of places in the KDS, so that an unexpected change in the motion of a single object can be processed efficiently. See [5,7] for more details.

Therefore, a good KDS for kinetic triangulation in $\mathbb{R}^2$ should have only nearly linear storage, process only a nearly-quadratic number of events, each in polylogarithmic time, and each moving point should be stored at only a polylogarithmic number of places in the KDS.

1.2. Our result

In Section 2, we present a simple triangulation scheme for a set P of n moving points in the plane. For the sake of efficient kinetization we make the scheme randomized, and assume a (natural) model in which the flight plans of the moving points are independent of the randomization used by the algorithm. The basic idea of the (static) triangulation is quite simple (some details are glossed over in this informal overview): We sort the points of P by their x-coordinates, split P at a (random) point p into a left portion P_L and a right portion P_R, compute recursively the upper convex hulls of $P_L \cup \{p\}$ and of $P_R \cup \{p\}$, and merge them into the upper convex hull of the whole set P.

This process results in a *pseudo-triangulation* of the portion of the convex hull of P lying above the x-monotone polygonal chain $\mathcal{C}(P)$ connecting the points of P in their x-order. Each pseudo-triangle is x-monotone, and consists of an upper *base* and of left and right lower concave chains, meeting at its bottom *apex*. See Fig. 1 for an illustration. A symmetric process is

[1] In [7], a KDS is considered to be efficient if the ratio between the number of processed events to the maximum possible number of changes in the maintained structure (i.e., the triangulation) is bounded by an arbitrary small power of the number of input objects. In our definition of efficient KDS, we only allow a degradation factor which is a polylogarithmic function of the number of input objects. We impose similar more stringest restrictions on the other performance parameters of the structure.

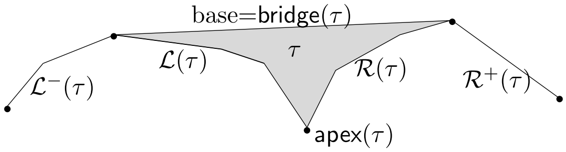

Fig. 1. A single pseudo-triangle τ in our pseudo-triangulation of $\mathcal{CH}^+(P)$. In addition to its funnel (drawn shaded), τ has two tails $\mathcal{L}^-(\tau)$, $\mathcal{R}^+(\tau)$.

applied to the portion of the hull below $C(P)$, by recursively computing lower convex hulls of the respective subsets of P. In particular, we obtain a hierarchical representation of $\mathcal{CH}(P)$, similar to the one of Overmars and van Leeuwen [20]; see also [5]. See [1,6,23] for additional applications of hierarchical pseudo-triangulations to kinetic problems.

To obtain a proper triangulation of (the convex hull of) P, we partition each pseudo-triangle τ into triangles. We accomplish this in the following randomized incremental manner. We process the vertices of τ (other than its apex and its leftmost and rightmost vertices) in order, according to the random ranks that they received during the first splitting phase, and draw from each processed vertex v a chord, within the current sub-pseudo-triangle τ' of τ containing v. This chord splits τ' into two sub-pseudo-triangles, and the process continues recursively until we obtain a triangulation of τ. We apply this process to each of the pseudo-triangles, to obtain the full triangulation of $\mathcal{CH}(P)$.

In Section 3, we prove that the expected number of events that can arise during the motion is $O(n^2\beta_{s+2}(n)\log n)$ (with s and β as defined in the abstract), and that the expected number of discrete (also called topological) changes caused in our triangulation by each such event is bounded by $O(\log n)$.

In Section 4, we show how to maintain this triangulation, as the points of P move, using a kinetic data structure that satisfies the criteria of [7], as listed above. There are several kinds of critical events we need to watch for, in which pairs of points are swapped in the x-order or triples of points become collinear. We process each event of the former type in $O(\log^2 n)$ expected time, and each event of the latter type in $O(\log n)$ expected time, for a total of $O(n^2\beta_{s+2}(n)\log^2 n)$ (expected) processing time. Our implementation encodes the pseudo-triangulation as a *treap* on P [21].

The upper bounds that we obtain on the number of discrete events, and on their overall processing time, are slightly better than those of the scheme of Agarwal, Wang and Yu [4], and we believe that our scheme is simpler (and more "explicit") than that of [4].

2. The static triangulation

In this section we describe a simple scheme for constructing a static triangulation $\mathcal{T}(P)$ of $\mathcal{CH}(P)$. We fix a random permutation π of the points of P. For each $p \in P$ we denote its rank in π as $\mathsf{priority}(p)$ (where these values range from 1 to n). Let $C(P)$ denote, as above, the x-monotone polygonal chain that connects the points of P in their x-order, assuming that no two points of P have the same x-coordinate. (In degenerate cases, which will arise at discrete instances during the motion of the points of P, $C(P)$ connects the points in the lexicographical order of their coordinates.) Since the two points of P with extreme x-coordinates are vertices of $\mathcal{CH}(P)$, $C(P)$ partitions $\mathcal{CH}(P)$ into two components, $\mathcal{CH}^+(P)$ and $\mathcal{CH}^-(P)$, lying respectively above and below $C(P)$. With no loss of generality, we only describe a triangulation $\mathcal{T}^+(P)$ of $\mathcal{CH}^+(P)$, and obtain the triangulation $\mathcal{T}^-(P)$ of $\mathcal{CH}^-(P)$ in a fully symmetric fashion. The overall triangulation $\mathcal{T}(P)$ is the union of $\mathcal{T}^+(P)$ and $\mathcal{T}^-(P)$.

2.1. A static pseudo-triangulation of $\mathcal{CH}^+(P)$

We first construct a *pseudo-triangulation* of $\mathcal{CH}^+(P)$ and then refine it into a triangulation by partitioning each pseudo-triangle into triangles.

Each pseudo-triangle τ that we construct consists of a *left tail*, a middle *funnel*, and a *right tail* (any of these substructures may be empty; the tails were not mentioned in the overview in the introduction). The funnel is an x-monotone simple polygon, whose boundary consists of an upper *base*, which is the segment connecting its leftmost and rightmost vertices, and of left and right lower concave *chains*, which are denoted respectively as $\mathcal{L}(\tau)$ and $\mathcal{R}(\tau)$. The point at which $\mathcal{L}(\tau)$ and $\mathcal{R}(\tau)$ meet is called the *apex* of τ and denoted by $\mathsf{apex}(\tau)$. The left chain $\mathcal{L}(\tau)$ extends from the left endpoint of the base to $\mathsf{apex}(\tau)$, and the right chain extends from $\mathsf{apex}(\tau)$ to the right endpoint of the base; see Fig. 1. In addition, τ may have a left tail[2] $\mathcal{L}^-(\tau)$ and a right tail $\mathcal{R}^+(\tau)$, so that $\mathcal{L}^-(\tau)$ is an x-monotone polygonal chain which extends from the left vertex of the funnel to the left, till the *left endpoint* $\mathsf{left}(\tau)$ of τ, so that $\mathcal{L}^-(\tau)\cup\mathcal{L}(\tau)$ is a concave chain, and symmetrically for $\mathcal{R}^+(\tau)$, that extends to the right till the *right endpoint* $\mathsf{right}(\tau)$ of τ. Moreover, the line containing the base of τ is an upper common tangent of $\mathcal{L}^-(\tau)\cup\mathcal{L}(\tau)$ and $\mathcal{R}(\tau)\cup\mathcal{R}^+(\tau)$. Again, see Fig. 1. We construct the pseudo-triangulation of $\mathcal{CH}^+(P)$ recursively. At each step of the recursion we have some subset $Q \subseteq P$ of points that are consecutive in the x-order of P, and we construct a pseudo-triangulation $\mathcal{PT}^+(Q)$ of $\mathcal{CH}^+(Q)$. At the topmost level of the recursion we have $Q = P$. The construction of $\mathcal{PT}^+(Q)$ proceeds as follows; see Fig. 2. Let $\mathsf{left}(Q)$ (resp., $\mathsf{right}(Q)$) denote the point of Q with the

[2] These tailed pseudo-triangles are a special case of so-called *geodesic triangles* introduced in [9].

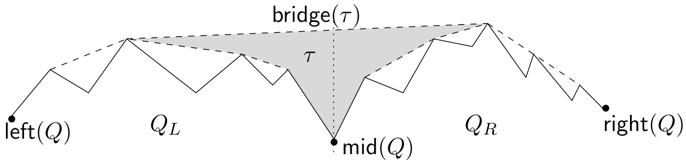

Fig. 2. The recursive pseudo-triangulation of $\mathcal{CH}^+(Q)$. We add to $\mathcal{PT}^+(Q)$ the pseudo-triangle τ (whose funnel is drawn shaded), with endpoints $\mathrm{left}(\tau) = \mathrm{left}(Q)$, $\mathrm{right}(\tau) = \mathrm{right}(Q)$, and $\mathrm{apex}(\tau) = \mathrm{mid}(Q)$, and then recursively construct $\mathcal{PT}^+(Q_L), \mathcal{PT}^+(Q_R)$.

minimal (resp., maximal) x-coordinate, and let $\mathrm{mid}(Q)$ be the point p of $Q \setminus \{\mathrm{left}(Q), \mathrm{right}(Q)\}$ with the minimum value of $\mathrm{priority}(p)$. Set $Q_L = \{p \in Q \mid x(p) \leqslant x(\mathrm{mid}(Q))\}$, $Q_R = \{p \in Q \mid x(p) \geqslant x(\mathrm{mid}(Q))\}$ (so $\mathrm{mid}(Q)$ belongs to both sets). We add to $\mathcal{PT}^+(Q)$ the following pseudo-triangle τ. The base of τ is the portion of the upper common tangent to $\mathcal{CH}^+(Q_L)$ and $\mathcal{CH}^+(Q_R)$ between the points of tangency. We call this base the *bridge* of τ and denote it by $\mathrm{bridge}(\tau)$. The left (resp., right) chain $\mathcal{L}(\tau)$ (resp., $\mathcal{R}(\tau)$) is the portion of the upper hull of Q_L (resp., Q_R) below $\mathrm{bridge}(\tau)$. We take $\mathcal{L}^-(\tau)$ to be the portion of the upper hull of Q_L to the left of $\mathcal{L}(\tau)$, and define $\mathcal{R}^+(\tau)$ symmetrically as the portion of the upper hull of Q_R to the right of $\mathcal{R}(\tau)$. The points $\mathrm{left}(Q)$ and $\mathrm{right}(Q)$ become the respective *endpoints* $\mathrm{left}(\tau)$, $\mathrm{right}(\tau)$ of τ. We also have $\mathrm{apex}(\tau) = \mathrm{mid}(Q)$ which belongs, by definition, to both chains. (The funnel of τ may be empty, if $\mathrm{mid}(Q)$ is a vertex of the upper hull of Q. In this case we can think of the funnel of τ as the singleton $\mathrm{apex}(\tau) = \mathrm{mid}(Q)$, and τ consists of the two tails $\mathcal{L}^-(\tau), \mathcal{R}^+(\tau)$, meeting at $\mathrm{mid}(Q)$, and forming together a common concave chain. Similarly, a pseudo-triangle may have an empty left tail and/or empty right tail.)

We then recursively pseudo-triangulate each of $\mathcal{CH}^+(Q_L)$, $\mathcal{CH}^+(Q_R)$. The recursion terminates when $|Q| \leqslant 3$ (by construction, $|Q| \geqslant 2$). If $|Q| = 3$ then we output a single pseudo-triangle τ, which is either a triangle, when the midpoint lies below the segment connecting the endpoints, or, in the opposite case, consists of the two segments $\mathcal{L}^-(\tau) = \mathrm{left}(\tau)\mathrm{apex}(\tau)$ and $\mathcal{R}^+(\tau) = \mathrm{apex}(\tau)\mathrm{right}(\tau)$. If $|Q| = 2$, no pseudo-triangle is output. In this case $\mathcal{CH}^+(Q)$ is a single edge of the chain $\mathcal{C}(P)$.

Consider a pseudo-triangle τ such that $\mathrm{left}(\tau)$ is not the leftmost point of P and $\mathrm{right}(\tau)$ is not the rightmost point of P. Then one can show that the triple $(\mathrm{left}(\tau), \mathrm{right}(\tau), \mathrm{apex}(\tau))$ have the smallest priorities among all points whose x-coordinates are between $x(\mathrm{left}(\tau))$ and $x(\mathrm{right}(\tau))$, inclusive (see Lemma 2.1 below). To make this true for all pseudo-triangles, we augment the initial point set P with two dummy points $p_{-\infty}$ and p_∞, and assign to them priorities -1 and 0, respectively. The points $p_{-\infty}$, p_∞ are symbolically placed at infinity to the left and to the right of P, and far below, so that the upper hull of the augmented point set is obtained from the upper hull of P by adding two vertical downward-directed rays at the leftmost and rightmost points of P.[3] Hence, any triangulation of $\mathcal{CH}^+(P)$ is also a triangulation of $\mathcal{CH}^+(P \cup \{p_{-\infty}, p_\infty\})$, and vice versa. In the rest of the paper we denote by P the augmented point set.

The following lemma gives an operational definition of $\mathcal{PT}^+(P)$, which will be used in the sequel.

Lemma 2.1. *Let a, b, and c be three points in P, such that $x(a) < x(b) < x(c)$. Then $\mathcal{PT}^+(P)$ contains a pseudo-triangle τ having endpoints $\mathrm{left}(\tau) = a$, $\mathrm{right}(\tau) = c$, and $\mathrm{apex}(\tau) = b$, if and only if*

$$\max\bigl\{\mathrm{priority}(a), \mathrm{priority}(c)\bigr\} < \mathrm{priority}(b) \leqslant \mathrm{priority}(p), \tag{1}$$

for all points $p \in P$ with $x(a) < x(p) < x(c)$.

Proof. To prove the "only if" part we proceed by induction on our recursive construction. Recall that at each recursive step we process some subset $Q \subseteq P$ whose points are consecutive in the x-order of P, and add to $\mathcal{PT}^+(P)$ a pseudo-triangle τ with $\mathrm{left}(\tau) = \mathrm{left}(Q)$, $\mathrm{right}(\tau) = \mathrm{right}(Q)$, and $\mathrm{apex}(\tau) = \mathrm{mid}(Q)$. To establish the asserted condition (1) for τ, it is sufficient to observe that each point p, such that $x(\mathrm{left}(Q)) < x(p) < x(\mathrm{right}(Q))$, satisfies $\mathrm{priority}(p) > \max\{\mathrm{priority}(\mathrm{left}(Q)), \mathrm{priority}(\mathrm{right}(Q))\}$. Indeed, the desired property holds initially for P by our choice of the artificial points $p_{-\infty}$ and p_∞ and their priorities. Assuming that this holds when we process some subset Q, and using the fact that $\mathrm{mid}(Q)$ is the point with smallest priority in the range $x(\mathrm{left}(Q)) < x(p) < x(\mathrm{right}(Q))$, the claim also holds for Q_L and Q_R.

For the "if" part, we observe that for every choice of $b \in P$ there is exactly one choice of a and c in P so that the triple (a, b, c) satisfies (1), and every point $b \in P$ is an apex of exactly one pseudo-triangle of $\mathcal{PT}^+(P)$ (and the apex of each pseudo-triangle is distinct from each of p_∞ and $p_{-\infty}$). The latter is easy to establish by induction on the increasing order of the priorities of the points. This, combined with the arguments in the "only if" part, completes the proof. □

2.2. The pseudo-triangulation tree

The pseudo-triangulation $\mathcal{PT}^+(P)$ can be represented by a binary tree in which every node v represents a pseudo-triangle $\tau_v \in \mathcal{PT}^+(P)$, and stores the point $p_v = \mathrm{apex}(\tau_v)$. The inorder of the tree is the increasing x-order of the apices

[3] Intuitively, think of $p_{-\infty}, p_\infty$ as the "points at infinity" on the parabola $y = -x^2$.

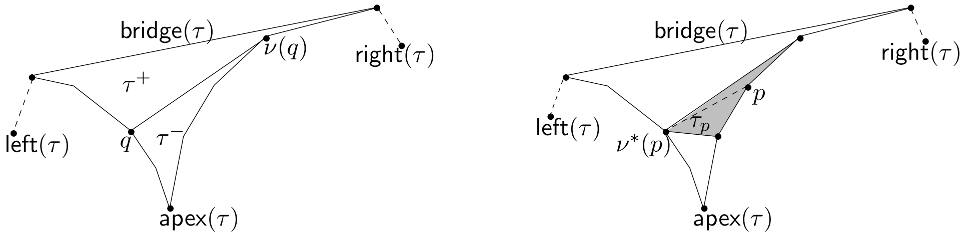

Fig. 3. Left: The first step of triangulating a single pseudo-triangle $\tau \in \mathcal{PT}^+(P)$. Right: During the recursive construction of $\mathcal{T}(\tau)$ every non-corner vertex p of the funnel of τ generates exactly one edge $e_p = p\nu^*(p)$, thus recursively splitting some sub-pseudo-triangle τ_p (drawn shaded). Note that in this figure $\nu^*(p) \neq \nu(p)$, which is the left endpoint of bridge(τ).

(i.e., the points of P). The subtree rooted at v represents the recursive pseudo-triangulation of $\mathcal{CH}^+(P_v \cup \{\mathsf{left}(\tau_v), \mathsf{right}(\tau_v)\})$, where $P_v \subseteq P$ denotes the set of points stored at the nodes of the subtree rooted at v. Note that left(τ_v) and right(τ_v) are not stored at this subtree—they are the next points to the left and to the right of the points of P_v. Abusing the notation slightly, we denote by $\mathcal{PT}^+(P)$ both the pseudo-triangulation $\mathcal{PT}^+(P)$ and the tree representing it.

Remark. Let v be a node in $\mathcal{PT}^+(P)$, so that left$(\tau_v) \neq p_{-\infty}$. Then left$(\tau_v)$ is stored at the lowest ancestor of v whose right subtree contains v. If left$(\tau_v) = p_{-\infty}$ then v belongs to the path from the root of $\mathcal{PT}^+(P)$ to the leftmost leaf. Symmetric properties hold for right(τ_v).

In summary, we have the following lemma, whose proof is immediate from the construction.

Lemma 2.2. *The tree representing $\mathcal{PT}^+(P)$ is a treap on $P \setminus \{p_{-\infty}, p_\infty\}$. That is, $\mathcal{PT}^+(P)$ is a heap with respect to the priorities, and a search tree with respect to the x-coordinates of the points.*

2.3. Triangulating a fixed pseudo-triangle

Let τ be a pseudo-triangle of $\mathcal{PT}^+(P)$. Assume that the funnel of τ is not empty, and is not already a triangle. We say that two vertices p, q of the funnel of τ, where p belongs to $\mathcal{L}(\tau)$ and q belongs to $\mathcal{R}(\tau)$, are *visible* from each other if pq does not intersect $\partial\tau$ (except at its endpoints); in this case pq lies inside the funnel of τ. Denote by $\nu(p)$ the rightmost point on the right chain which is visible from p. Note that either $\nu(p)$ is the rightmost vertex of τ or $p\nu(p)$ is an upper tangent to $\mathcal{R}(\tau)$. A symmetric definition and properties hold for points q on $\mathcal{R}(\tau)$. This definition also applies when p is the leftmost vertex of $\mathcal{L}(\tau)$ and when q is the rightmost vertex of $\mathcal{R}(\tau)$ (the endpoints of bridge(τ)), in which case $\nu(p) = q$ and $\nu(q) = p$. See Fig. 3 (left).

The triangulation $\mathcal{T}(\tau)$ of τ is obtained by recursively splitting τ by chords into sub-pseudo-triangles, in the following manner. Choose the minimum priority vertex q of the funnel of τ, other than the leftmost and the rightmost vertices and the apex. Assume, without loss of generality, that q lies on $\mathcal{L}(\tau)$. See Fig. 3 (left). The segment $q\nu(q)$ splits τ into two sub-pseudo-triangles τ^+ and τ^-. The pseudo-triangle τ^+ has q as an apex and the same base as τ. Its left chain is the portion of $\mathcal{L}(\tau)$ from q to the left, and its right chain is the concatenation of $q\nu(q)$ with the portion of $\mathcal{R}(\tau)$ to the right of $\nu(q)$. The pseudo-triangle τ^- has $q\nu(q)$ as its base, the same apex as τ, and its left and right chains are the portions of $\mathcal{L}(\tau)$ and $\mathcal{R}(\tau)$ delimited respectively by q and by $\nu(q)$. A symmetric situation arises when $q \in \mathcal{R}(\tau)$. We add the edge $q\nu(q)$ to $\mathcal{T}(\tau)$, and recursively triangulate each of τ^+ and τ^-. We say that the edge $q\nu(q)$ in $\mathcal{T}(\tau)$ is *generated* by q. In the further recursive steps, we redefine $\nu(p)$, for vertices p of each of these sub-pseudo-triangles, restricting the visibility to within the respective pseudo-triangle. Note that for any pair of vertices p, q that lie on the same chain of τ, the segments $p\nu(p)$ and $q\nu(q)$ do not intersect in their relative interiors. Therefore, if $\nu(p)$ changes after a recursive call then it must change to a vertex of the base of the corresponding sub-pseudo-triangle. See Fig. 3 (right). The recursion bottoms out when the interior of τ is a triangle. Note also that all the chords in $\mathcal{T}(\tau)$ cross the vertical ray above apex(τ), and so they are totally ordered in the vertical direction.

2.4. Properties of $\mathcal{T}(\tau)$

Every vertex p of the funnel of τ, other than the leftmost and the rightmost vertices and the apex, generates exactly one edge e_p during the whole recursive process. (For example, in Fig. 3 (left), the vertex $\nu(q)$ will not generate an edge in τ^-, since it is an endpoint of that funnel, but will still generate an edge within τ^+, or within some recursive sub-pseudo-triangle of τ^+.) We denote by τ_p the sub-pseudo-triangle in which e_p is generated, and by $\nu^*(p)$ the other endpoint of e_p. Note that $\nu^*(p)$ is either the original $\nu(p)$ or an endpoint of the base of τ_p.

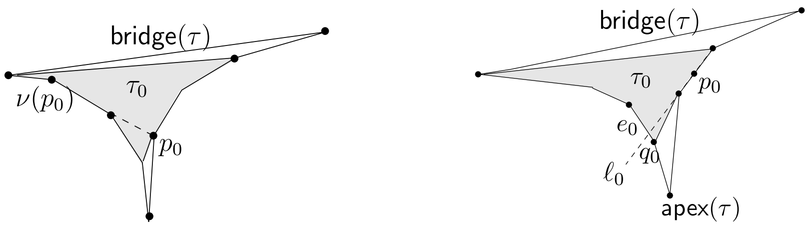

Fig. 4. Left and right: Visibility and envelope events (respectively). The shaded sub-pseudo-triangle τ_0 contains all edges that are inserted into or deleted from $\mathcal{T}(\tau)$ at this event.

3. Number of discrete changes in $\mathcal{T}(P)$

In this section we bound the overall expected number of discrete changes that $\mathcal{T}(P(t))$ experiences as the points of P move along (continuous) pseudo-algebraic trajectories of constant description complexity. The analysis is with respect to a fixed random permutation π of P drawn ahead of the motion, so that the motion is "oblivious" to the choice of π. Thus, even though the x-order of the points may change during the motion, each point retains its initial priority, and the permutation π is still a random permutation of P, with respect to the x-order of these points, at any fixed t.

3.1. Discrete changes in $\mathcal{PT}^+(P)$

For a fixed time instance $t \in \mathbb{R}$, each pseudo-triangle $\tau \in \mathcal{PT}^+(P(t))$ is defined by its endpoints $\mathsf{left}(\tau)$, $\mathsf{right}(\tau)$, and by its apex $\mathsf{apex}(\tau)$. Given such a triple of points, they define a valid pseudo-triangle at time t if and only if they, and the points in-between in the x-order, satisfy the conditions of Lemma 2.1 (at time t). Thus, as long as the x-order of the points does not change, $\mathcal{PT}^+(P(t))$ does not change either. That is, it consists of a fixed set of pseudo-triangles, each defined by a fixed triple of points. However, the geometric structure of a pseudo-triangle may change during such a time interval, and we will bound the number of these changes separately. Changes in (the labelings of the pseudo-triangles of) $\mathcal{PT}^+(P(t))$ occur only at discrete times when the x-order of some pair of points in $P(t)$ changes; we refer to these changes as *x-swap events*.

We assume that each pseudo-triangle τ is present in $\mathcal{PT}^+(P(t))$ during a maximal connected time interval $I(\tau)$, which is associated with τ. That is, pseudo-triangles with the same triple $\mathsf{left}(\tau)$, $\mathsf{right}(\tau)$, and $\mathsf{apex}(\tau)$ that appear in $\mathcal{PT}^+(P(t))$ at disjoint time intervals, are considered distinct. We emphasize that all the other features of τ, such as $\mathsf{bridge}(\tau)$, the chains $\mathcal{L}(\tau)$ and $\mathcal{R}(\tau)$, and the triangulation $\mathcal{T}(\tau)$ of its funnel, may undergo discrete changes during the time interval $I(\tau)$. A pseudo-triangle τ is created or destroyed only at a swap event when a point $p \in P$ with $\mathsf{priority}(p) < \mathsf{priority}(\mathsf{apex}(\tau))$ crosses one of the vertical lines through its endpoints $\mathsf{left}(\tau)$ and $\mathsf{right}(\tau)$ (of course, this also subsumes the cases in which $\mathsf{priority}(p)$ is smaller than that of an endpoint of τ), or when the x-order of the points in the triple defining τ changes. In the former case, if $\mathsf{priority}(p) > \max\{\mathsf{priority}(\mathsf{left}(\tau)), \mathsf{priority}(\mathsf{right}(\tau))\}$ then τ is replaced by another pseudo-triangle τ' with the same endpoints $\mathsf{left}(\tau') = \mathsf{left}(\tau)$, $\mathsf{right}(\tau') = \mathsf{right}(\tau)$ but with p as a new apex. If $\mathsf{priority}(p) < \max\{\mathsf{priority}(\mathsf{left}(\tau)), \mathsf{priority}(\mathsf{right}(\tau))\}$ then p replaces the endpoint it was swapped with.

In the remainder of this paper $\mathcal{PT}^+(P(t))$ denotes the pseudo-triangulation (and the tree which represents it) at a fixed moment of time t, whereas $\mathcal{PT}^+(P)$ denotes the set of all pseudo-triangles that ever appear in $\mathcal{PT}^+(P(t))$ for some $t \in \mathbb{R}$. Thus, each pseudo-triangle τ in our kinetic pseudo-triangulation $\mathcal{PT}^+(P)$ is *defined* by at most five points: $\mathsf{left}(\tau)$, $\mathsf{right}(\tau)$, $\mathsf{apex}(\tau)$, and at most two additional points that determine, by swaps with the endpoints of τ, the endpoints of the lifespan $I(\tau)$ of τ in $\mathcal{PT}^+(P(t))$.

3.2. Discrete changes in $\mathcal{T}(\tau)$

Fix a pseudo-triangle $\tau \in \mathcal{PT}^+(P)$. We consider only discrete changes in the funnel of τ and its triangulation $\mathcal{T}(\tau)$, and ignore changes in the tails $\mathcal{L}^-(\tau), \mathcal{R}^+(\tau)$ (unless they also affect the funnel). This is because the changes in the tails will also show up as changes in the funnels of other pseudo-triangles that are created further down the recursion.

For a fixed time instance $t \in I(\tau)$, the combinatorial structure of the triangulation $\mathcal{T}(\tau)$ of τ depends only on the discrete structure of the boundary of the funnel of τ (i.e., the ordered sequences of the points along the chains $\mathcal{L}(\tau)$, $\mathcal{R}(\tau)$, and the base $\mathsf{bridge}(\tau)$) and the visibility points $\nu(p)$ of all the vertices of the funnel of τ, excluding $\mathsf{apex}(\tau)$ (of course, it also depends on π). Therefore, as the points of P move during the time interval $I(\tau)$, $\mathcal{T}(\tau)$ can change combinatorially only at events at which the boundary or visibility structure of τ changes. These events fall into the following three types:

(i) *Envelope events*, which occur at time values when one of the chains $\mathcal{L}(\tau)$, $\mathcal{R}(\tau)$ contains three collinear vertices; see Fig. 4 (right). This happens when a vertex (which is not an endpoint of $\mathsf{bridge}(\tau)$) is added to or removed from one of the chains bounding τ. We denote the total number of such events during the period $I(\tau)$ by E_τ.

(ii) *Visibility events*, at which a vertex q of $\mathcal{R}(\tau)$ becomes collinear with an edge pr of $\mathcal{L}(\tau) \cup \mathcal{L}^-(\tau)$, or vice versa. See Fig. 4 (left) ($\mathcal{L}^-(\tau)$ is relevant only for visibility events that affect $\mathsf{bridge}(\tau)$ and then pr has to be its rightmost edge, and

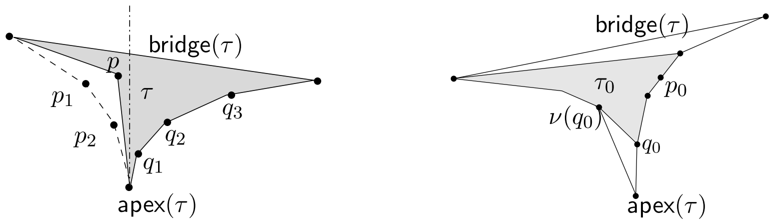

Fig. 5. Left: Swap event. The funnel of τ immediately before the x-swap between p and $\mathsf{apex}(\tau)$, which causes the vertices p_1 and p_2 to appear on $\mathcal{L}(\tau)$, and the vertices q_1, q_2, q_3 to disappear from $\mathcal{R}(\tau)$. Right: Envelope event. The case in which q_0 lies on $\mathcal{R}(\tau)$.

symmetrically for $\mathcal{R}^+(\tau)$). This happens when $\nu(q)$ changes from p to r, or vice versa. In particular, each (discrete) change of $\mathsf{bridge}(\tau)$ corresponds to a visibility event in which the bridge becomes collinear with an edge of $\mathcal{L}^-(\tau) \cup \mathcal{L}(\tau)$ or of $\mathcal{R}(\tau) \cup \mathcal{R}^+(\tau)$ that is incident to the respective endpoint of the bridge. We denote the total number of visibility events during $I(\tau)$ by V_τ.

A special case of this event occurs when $\mathsf{bridge}(\tau)$ is created (resp., destroyed), so that just before (resp., after) the event, the funnel of τ is empty. Note that immediately after (resp., before) the creation (resp., destruction) of $\mathsf{bridge}(\tau)$, the funnel of τ is a triangle.

(iii) *Swap events*, at which some point $p \in P$, satisfying $\mathsf{priority}(p) > \mathsf{priority}(\mathsf{apex}(\tau))$, crosses one of the vertical lines through $\mathsf{left}(\tau)$, $\mathsf{right}(\tau)$ or $\mathsf{apex}(\tau)$. Note that a single swap event of this kind may cause massive discrete changes, of highly nonlocal nature, in the chains $\mathcal{L}(\tau)$, $\mathcal{R}(\tau)$, in the visibility pointers $\nu(q)$ of the vertices of τ, and in $\mathsf{bridge}(\tau)$. See Fig. 5 (left) for an illustration.

Note that a swap between any other pair of points p, q within the x-range of τ can be ignored in the present analysis, since the lower of the two points cannot belong to the funnel of τ at the time of swap.

Assuming general position of the trajectories of the points, these events occur at distinct time instances (except that the same event may show up, in different forms, in several pseudo-triangles). Degeneracies in the point trajectories can be handled, both algorithmically and combinatorially, by any of the standard symbolic perturbation techniques, such as simulation of simplicity [14].

A visibility event happens when $\nu(p)$ changes for some point p; we then say that p is *involved* in the visibility event. An envelope event happens when a point p joins or leaves one of the chains $\mathcal{L}(\tau)$, $\mathcal{R}(\tau)$; we then say that p is *involved* in the envelope event.

Lemma 3.1. *The only point p for which $\nu(p)$ changes in an envelope event is the point p involved in the event.*

Proof. The lemma follows since at the moment following (resp., preceding) the appearance of p on (resp., disappearance from) its chain, say $\mathcal{L}(\tau)$, its two incident edges are almost collinear. Thus, all vertices q on the opposite chain satisfy $\nu(q) \neq p$ both before and after the event, and $\nu(q)$ is not affected by the event. □

3.3. The number of changes in $\mathcal{T}(\tau)$: overview

Here is a brief informal overview of the analysis. There are three types of critical events to consider: swap events, envelope events, and visibility events. We consider a fixed pseudo-triangle τ, defined, say, by 5 points of P, and estimate (a) the probability that τ arises in the pseudo-triangulation, and (b) conditioned on such an appearance, the overall expected number of changes that the triangulation of τ experiences during the motion of the points.

Swap events might potentially cause many changes in the structure and triangulation of τ. The overall cost of these events is $O(N_\tau^2)$, where N_τ is the number of points of P that participate in τ during its lifespan. What "saves" us here is the fact that pseudo-triangles τ with large N_τ are unlikely to arise in $\mathcal{PT}^+(P)$, making the overall expected contribution of swap events satisfy the asserted bound.

Envelope and visibility events require a more careful analysis. Each such event may completely change the triangulation, but only within the sub-pseudo-triangle where the event occurs. We show that the expected size of this sub-pseudo-triangle is only $O(\log N_\tau)$ (see Proposition 3.2), so this bounds the expected contribution of an envelope of visibility event. We then show, using standard arguments based on lower envelopes and Davenport–Schinzel sequences, that the number of envelope and visibility events in τ is $O(N_\tau^2 \beta_{s+2}(N_\tau))$, where s is the maximum number of times in which a triple of points can become collinear (see Theorem 3.4). Putting everything together, the bound on the expected number of changes in the triangulation follows.

3.4. The cost of swap events within τ

We define P_τ as the set of points $p \in P$, other than $\mathsf{apex}(\tau)$, that appear on $\mathcal{C}(P)$ between $\mathsf{left}(\tau)$ and $\mathsf{right}(\tau)$, at any time during the life span $I(\tau)$, and put $N_\tau = |P_\tau|$. (Note that the points of P_τ may enter or leave the interval between

$\mathsf{left}(\tau)$ and $\mathsf{right}(\tau)$ in the middle of $I(\tau)$, at x-swaps with either $\mathsf{left}(\tau)$ or $\mathsf{right}(\tau)$.) As noted above, every point $p \in P_\tau$ satisfies $\mathsf{priority}(p) > \mathsf{priority}(\mathsf{apex}(\tau))$. Clearly, our triangulation undergoes $O(N_\tau)$ swap events during $I(\tau)$ (recall that we only consider swaps with $\mathsf{left}(\tau), \mathsf{right}(\tau)$ or $\mathsf{apex}(\tau)$), and each of them leads to $O(N_\tau)$ edge insertions and deletions to $\mathcal{T}(\tau)$ (the maximum number of edges in the whole triangulation $\mathcal{T}(\tau)$), for a total of $O(N_\tau^2)$ such updates. We next bound the number of discrete changes in $\mathcal{T}(\tau)$ caused by events of the remaining two types.

3.5. The cost of an envelope or visibility event

Fix a set of at most five points that can potentially define a pseudo-triangle for some set of priorities. This set has an associated time interval $[t_1, t_2]$, and consists of three points a, b, and c, such that, at all times $t_1 < t < t_2$, $x(a(t)) < x(b(t)) < x(c(t))$, and of two additional points d_1 and d_2 (each of which could be equal to b), so that the x-coordinate of d_i swaps with either a or c at times t_i, for $i = 1, 2$. For some drawings of the random priorities, τ appears as a pseudo-triangle, and for other drawings it does not. For τ to appear in $\mathcal{PT}^+(P)$, the priorities of $a = \mathsf{left}(\tau)$ and $c = \mathsf{right}(\tau)$ should be smaller than the priority of $b = \mathsf{apex}(\tau)$. The priorities of d_1 and d_2 have to be at most the priority of $b = \mathsf{apex}(\tau)$, and the priorities of all other points in P_τ should be larger than the priority of b. The probability of this to happen, assuming a, b, c, d_1, d_2 are all distinct, is easily seen to be $O(1/N_\tau^5)$ (for $N_\tau > 0$).

When we condition on drawings in which τ indeed appears in $\mathcal{PT}^+(P)$, the following holds.

Proposition 3.2. *Let τ be a pseudo-triangle in the kinetic triangulation $\mathcal{PT}^+(P)$. Then the expected number of discrete changes in the triangulation $\mathcal{T}(\tau)$ of τ, after any single envelope or visibility event that happens during the period $I(\tau)$, and conditioned on τ appearing in $\mathcal{PT}^+(P)$, is $O(\log N_\tau) = O(\log n)$.*

Proof. Clearly, the chords of $\mathcal{T}(\tau)$ (the additional edges that partition τ into triangles) admit a total vertical order, because they all cross the vertical line through $\mathsf{apex}(\tau)$. Consider a time value $t_0 \in \mathbb{R}$ when an envelope or a visibility event occurs, and let $p_0 \in \mathcal{L}(\tau) \cup \mathcal{R}(\tau)$ be the point involved in the event. Let t_0^- (resp., t_0^+) be the time right before (resp., after) the event. Note that p_0 cannot be the apex of τ (unless the funnel of τ is already, or is going to become, a triangle). Note also that p_0 is not a vertex of $\mathsf{bridge}(\tau)$, neither at t_0^+ nor at t_0^-, unless p_0 is involved in a visibility event that changes $\mathsf{bridge}(\tau)$. In the latter case, $\mathcal{T}(\tau)$ gains or loses its topmost triangle at time t_0 but there are no other changes in the triangulation, as is easily checked. We may therefore assume that $\mathsf{bridge}(\tau)$ does not change at time t_0, and that p_0 is not a vertex of $\mathsf{bridge}(\tau)$.

With no loss of generality, we assume that p_0 is a vertex of $\mathcal{R}(\tau)$ at time t_0^+, and treat the remaining cases symmetrically (for a visibility event, p_0 belongs to $\mathcal{R}(\tau)$ also at time t_0^-). Consider the triangulation $\mathcal{T}(\tau)$ at time t_0^+ (which we would have obtained if we were to reconstruct $\mathcal{T}^+(P)$ statically at time t_0^+). Let τ_0 be the sub-pseudo-triangle of τ within which the edge $p_0\nu^*(p_0)$ is generated during the construction of $\mathcal{T}^+(P)$ (see Fig. 4 (right)). Note that the event at time t_0 leaves unchanged the visibility vertex $\nu(p)$ of each vertex p in τ other than p_0. Indeed, this follows from Lemma 3.1 for envelope events and is obvious for visibility events, using our assumption that $\mathsf{bridge}(\tau)$ does not change. The recursive construction of $\mathcal{T}(\tau)$ is easily seen to imply that τ_0 appears as a sub-pseudo-triangle in the construction also at time t_0^-. Indeed, an easy inductive argument on the order of the ranks of the funnel vertices implies that the modified visibility vertices $\nu^*(p)$, and the resulting chords $p\nu^*(p)$, also do not change, up to the point where τ_0 is constructed. Right after this step, the chord from p_0 is drawn, so the rest of the construction of $\mathcal{T}(\tau)$ might change completely, but only within τ_0. Hence, τ_0 contains every edge that is inserted to or deleted from $\mathcal{T}(\tau)$ at time t_0. Therefore, the number of changes in $\mathcal{T}(\tau)$ is bounded by $O(W_0)$, where W_0 denotes the number of vertices of τ_0 at the time of the event.

Note that W_0 is a random variable depending (only) on the permutation $\pi(P_\tau)$ of P_τ, which is obtained by restricting π to P_τ. Of course, we also depends on the geometry of the motion of the points, but this dependence is fully deterministic and does not depend on random draws. Recall that we condition the analysis on permutations π such that τ indeed appears in $\mathcal{PT}^+(P)$. In these permutations, the points of P_τ have to follow all the (at most) five points defining τ, but as long as they obey this restriction they can appear in any order. It follows that, in our conditional probability subspace, the restriction of π to P_τ is a random permutation of P_τ.

To bound the expected value of W_0, we fix an arbitrary threshold $k \geqslant 10$ and prove that the event $\{W_0 > k\}$ occurs with probability at most $O(1/k)$. The expected value of W_0 is then bounded by

$$\sum_{i=0}^{\log N_\tau} 2^{i+1}\Pr\{W_0 > 2^i\} = O\left(\sum_{i=0}^{\log N_\tau} 1\right) = O(\log N_\tau). \tag{2}$$

To show that $\Pr\{W_0 > k\} = O(1/k)$, we proceed through the following cases. In each case, except for the last one, we find a set S_0 of $\Omega(k)$ points which does not depend on $\pi(P_\tau)$, so that all its elements must appear in $\pi(P_\tau)$ after p_0. This readily implies the asserted bound. The last case is more involved but it is still based on the same general idea.

3.5.1. Visibility event

If $\nu^*(p_0)$ is a vertex of the base of τ_0, both at time t_0^- and at time t_0^+, then $\mathcal{T}(\tau)$ does not change combinatorially at time t_0. Otherwise, as follows from the discussion in Section 2, all three vertices that become collinear in the event appear in τ_0, both before and after the event, which implies that $\nu(p_0) = \nu^*(p_0)$ at both times t_0^- and t_0^+ (although they assume different values of these times).

Recall that p is assumed to be a vertex of $\mathcal{R}(\tau)$, and suppose that $W_0 > k$. If τ_0 contains at least $k/2$ vertices of $\mathcal{R}(\tau)$, then it also contains a sequence S_0 of $k/4 - 1$ consecutive vertices of $\mathcal{R}(\tau)$ either immediately to the left or immediately to the right of p_0. Otherwise, τ_0 contains $\nu(p_0)$ together with at least $k/2 - 1$ other vertices of $\mathcal{L}(\tau)$, so it must contain a sequence S_0 of $k/4 - 1$ consecutive vertices of $\mathcal{L}(\tau)$ lying either immediately to the left or immediately to the right of $\nu(p_0)$. In both cases, the key observation is that S_0 depends only very mildly on $\pi(P_\tau)$. Moreover, regardless of which S_0 we use, p_0 precedes all the $\Theta(k)$ vertices of S_0 in $\pi(P_\tau)$ (except possibly for one extremal vertex, which is a corner of τ_0). That is, assume that π causes τ to appear in $\mathcal{PT}^+(P)$. At the time t_0 of the event, the vertices forming $\mathcal{R}(\tau)$ and $\mathcal{L}(\tau)$ are known, and do not depend on π. S_0 is one of a constant number of contiguous subsequences of one of these chains, defined by p_0 and by $\mathsf{apex}(\tau)$. Thus, conditioned on τ appearing in $\mathcal{PT}^+(P)$, the collection of $O(1)$ possible choices for S_0 is fixed, and π can only affect the choice of which of these sequences is S_0. As noted above, this establishes the asserted bound.

3.5.2. Envelope event

Again, suppose that $W_0 > k$. If τ_0 contains at least $k/2$ vertices of $\mathcal{R}(\tau)$, the bound follows by exactly the same argument as in the case of a visibility event. Otherwise, if τ_0 contains $\mathsf{apex}(\tau)$ we set S_0 to be the first $k/2 - 2$ points of $\mathcal{L}(\tau)$ to the left of $\mathsf{apex}(\tau)$. Again, S_0 does not depend on $\pi(P_\tau)$, and all its elements must appear in $\pi(P_\tau)$ after p_0, so the bound follows.

We therefore assume that τ_0 contains at most $k/2$ vertices of $\mathcal{R}(\tau)$, and that its apex q_0 is distinct from $\mathsf{apex}(\tau)$. Thus, the edge $q_0\nu^*(q_0)$ that q_0 generates is the lowest edge of τ_0 that is a chord of τ. We argue that $\nu^*(q_0) = \nu(q_0)$ (before and after t_0; the definition of τ_0 implies that q_0 precedes p_0 in $\pi(P_\tau)$). Indeed, otherwise, by the definition of $\mathcal{T}(\tau)$, $\nu^*(q_0)$ is a vertex of the base of τ_0, which happens only if one of the chains of τ_0 consists of the single edge $q_0\nu^*(q_0)$. Since $p_0 \in \mathcal{R}(\tau)$ and is involved in an envelope event, the edge $q_0\nu^*(q_0)$ must be the only edge of the left chain of τ_0, which contradicts the fact that $\mathcal{L}(\tau)$ must contain at least $k/2$ vertices of τ_0 (for $k \geqslant 10$). We distinguish between the following two cases.

(i) $q_0 \in \mathcal{L}(\tau)$ (as depicted in Fig. 4 (right)). Then the entire left chain of τ_0 is contained in $\mathcal{L}(\tau)$. Let ℓ_0 be the line passing through p_0 and the other two vertices of $\mathcal{R}(\tau)$ participating in the envelope event, and let e_0 be the edge of $\mathcal{L}(\tau)$ intersected by ℓ. Clearly, e_0 is contained in τ_0, because otherwise $\mathcal{R}(\tau_0)$ would not be convex. If τ_0 contains $k/4 - 1$ consecutive vertices of $\mathcal{L}(\tau)$ that lie immediately to the left of e_0, we set S_0 to be the set of these points, except for the leftmost one (which may be the endpoint of the base of τ_0). Otherwise we set S_0 to be the set of $k/4 - 2$ points lying on $\mathcal{L}(\tau)$ to the right of e_0. Since the definition of e_0 does not depend on $\pi(P_\tau)$, the set S_0 too does not depend on $\pi(P_\tau)$.

(ii) $q_0 \in \mathcal{R}(\tau)$ (as depicted in Fig. 5 (right)). In this case we define at most $k/2$ sets, each consisting of $\Omega(k)$ points and independent of $\pi(P_\tau)$, such that all the points in at least one of these sets appear after both p_0 and q_0 in $\pi(P_\tau)$. We fix q_0 on $\mathcal{R}(\tau)$ to the left of p_0 and define S_{q_0} as the set of $k/2 - 2$ consecutive vertices of $\mathcal{L}(\tau)$ that appear at time t_0 (along $\mathcal{L}(\tau)$) immediately to the left of $\nu^*(q_0) = \nu(q_0)$. By the current assumptions, if q_0 is indeed the apex of τ_0 then all points $q \in S_{q_0}$ belong to τ_0 and, hence, satisfy $\mathsf{priority}(q) > \mathsf{priority}(p_0) > \mathsf{priority}(q_0)$. Since q_0 is fixed, S_{q_0} is also fixed and is independent of $\pi(P_\tau)$. Hence, the above event happens with probability $O(1/k^2)$. Moreover, q_0 is one of the at most $k/2$ vertices of $\mathcal{R}(\tau)$ that lie to the left of p_0. Hence, by the probability union bound, the total probability of this scenario (over all the appropriate vertices $q_0 \in \mathcal{R}(\tau)$) is $O(1/k)$.

We have proved that $\Pr(W_0 > k) = O(1/k)$ for any $k \geqslant 10$. This implies Eq. (2) and completes the proof of Proposition 3.2. □

3.6. The overall cost of envelope and of visibility events within τ

Recall, for the following corollary of Proposition 3.2, that E_τ and V_τ denote, respectively, the number of envelope events and of visibility events that occur within τ during its lifespan.

Corollary 3.3. *Let τ be a pseudo-triangle in the kinetic pseudo-triangulation $\mathcal{PT}^+(P)$. Then the expected number of edge insertions and deletions in $\mathcal{T}(\tau)$ during the period $I(\tau)$, conditioned upon the event that τ appears in $\mathcal{PT}^+(P)$, is $O((E_\tau + V_\tau)\log N_\tau + N_\tau^2)$.*

For a fixed pseudo-triangle τ (including the choice of the connected life span $I(\tau)$), V_τ and E_τ are 2-valued random variables: They are 0 if τ does not appear in $\mathcal{PT}^+(P)$, and assume a fixed "deterministic" value if τ does appear. The following theorem gives an upper bound on these values.

Theorem 3.4. *For each pseudo-triangle τ we have $V_\tau = O(N_\tau^2 \beta_{s+2}(N_\tau))$ and $E_\tau = O(N_\tau^2 \beta_{s+2}(N_\tau))$, where s is the maximum number of times at which any fixed triple of points of P becomes collinear.*

Proof. We show the bound for visibility events. The bound for envelope events is known (see [1,5]) and can be proved similarly.

We fix a point $p \in P_\tau$ and count the number of visibility events where p is a vertex of $\mathcal{L}(\tau)$ that is collinear with an edge of $\mathcal{R}(\tau)$. To do so, we define, for each $q \in P'_\tau = P_\tau \cup \{\mathsf{right}(\tau)\} \setminus \{p\}$, a partially defined function $\varphi_{p,q}(t)$ that measures the angle between pq and the y-axis, and whose domain consists of all $t \in \mathbb{R}$ at which $x(\mathsf{left}(\tau)) \leqslant x(p) \leqslant x(\mathsf{apex}(\tau)) \leqslant x(q) \leqslant x(\mathsf{right}(\tau))$. Clearly, each visibility event under consideration corresponds to a breakpoint of the lower envelope of $\{\varphi_{p,q}\}_{q \in P'_\tau}$ (but not necessarily vice versa; for example, such a breakpoint can arise at a time when p is not a vertex of the funnel of τ). Since any pair φ_{p,q_1}, φ_{p,q_2} of these functions can intersect in at most s points (these are times at which p, q_1, and q_2 are collinear), and for each q the domain of $\varphi_{p,q}(t)$ consists of a constant number of intervals (delimited by times at which either p or q swap with $\mathsf{left}(\tau)$, $\mathsf{right}(\tau)$, or $\mathsf{apex}(\tau)$), it follows that the number of breakpoints is $O(N_\tau \beta_{s+2}(N_\tau))$ [22]. A symmetric argument holds for the number of visibility events where p is a vertex of $\mathcal{R}(\tau)$ that is collinear with an edge of $\mathcal{L}(\tau)$. Repeating this analysis for each $p \in P$ yields the asserted overall bound. □

Fix a pseudo-triangle τ. Conditioned on priorities that cause τ to appear in $\mathcal{PT}^+(P)$, Corollary 3.3 and Theorem 3.4 imply that the expected number of discrete changes in $\mathcal{T}(\tau)$ is $O(N_\tau^2 \beta_{s+2}(N_\tau)\log N_\tau)$. Let $\mathsf{Pr}(\tau)$ be the probability that τ indeed appears in $\mathcal{PT}^+(P)$. Then the total expected number of discrete changes in $\mathcal{PT}^+(P)$ is

$$O\left(\sum_\tau \mathsf{Pr}(\tau) N_\tau^2 \beta_{s+2}(N_\tau)\log N_\tau\right) = O\left(\beta_{s+2}(n)\log n \sum_\tau \mathsf{Pr}(\tau) N_\tau^2\right).$$

3.7. Putting everything together

We need the following lemma to bound the expected number of discrete changes in $T(\tau)$, *over all pseudo-triangles* τ in our kinetic pseudo-triangulation $\mathcal{PT}^+(P)$.

Lemma 3.5. $\sum_\tau \mathsf{Pr}(\tau) N_\tau^2 = O(n^2 \log n)$, *where the sum is over all* (*possible sets of* $1 \leqslant h \leqslant 5$ *points defining*) *possible pseudo-triangles* τ.

Proof. Our analysis is based on the standard probabilistic technique of Clarkson and Shor [10]. Without loss of generality, we consider only pseudo-triangles τ with $N_\tau > 0$ that are defined by five distinct points of $P \setminus \{p_{-\infty}, p_\infty\}$. (Pseudo-triangles defined by fewer than five distinct points, or those whose defining 5-tuple includes $p_{-\infty}$ and/or p_∞ are analyzed similarly, replacing the exponent 5 by the appropriate $1 \leqslant h \leqslant 4$.) Thus, as already noted, $\mathsf{Pr}(\tau) = O(1/N_\tau^5)$, because τ appears in $\mathcal{PT}^+(P)$ if and only if the priorities of the five points that define τ are smaller than the priorities of all other points in P_τ (and $\mathsf{apex}(\tau)$ has the largest priority among the defining points). Therefore $\sum_\tau \mathsf{Pr}(\tau) N_\tau^2 = O(\sum_\tau 1/N_\tau^3)$.

In what follows, we call N_τ the *level* of τ. Let $M_k(n)$ (resp. $M_{\leqslant k}(n)$) denote the maximum number of pseudo-triangles of level k (resp., of level at most k), defined by 5 points, in a set of n moving points. We claim that $M_{\leqslant k}(n) = O(n^2k^3)$. To see this, consider all the pseudo-triangles τ (defined by five points) whose birth time is determined by a fixed x-swap event occurring at some time t_0, between some pair of points $a, b \in P$. Assume without loss of generality that $a = \mathsf{left}(\tau)$. Then $\mathsf{apex}(\tau)$ and $\mathsf{right}(\tau)$ are among the $k+2$ points whose x-coordinates lie at time t_0 immediately to the right of $x(a) = x(b)$. Similarly, the fifth point, which is responsible for the destruction of τ, is one of the first $k+1$ points whose x-coordinates enter the interval between $x(\mathsf{left}(\tau)) = x(a)$ and $x(\mathsf{right}(\tau))$. Thus, each of the $O(n^2)$ x-swap events defines the creation time of at most $O(k^3)$ pseudo-triangles of level at most k, which readily implies the asserted bound on $M_{\leqslant k}(n)$. We thus have

$$\sum_\tau \mathsf{Pr}(\tau) N_\tau^2 = O\left(\sum_\tau \frac{1}{N_\tau^3}\right) = O\left(\sum_{k\geqslant 1} \frac{M_k(n)}{k^3}\right) = O\left(\sum_{k=1}^{n} \frac{M_{\leqslant k}(n) - M_{\leqslant_{k-1}}(n)}{k^3}\right)$$
$$= O\left(\sum_{k=1}^{n-1} \frac{M_{\leqslant k}(n)}{k^4} + \frac{M_{\leqslant n}(n)}{n^3}\right) = O\left(\sum_{k\geqslant 1} \frac{M_{\leq k}(n)}{k^4} + n^2\right) = O\left(\sum_{k\geqslant 1} \frac{n^2}{k}\right) = O\big(n^2 \log n\big).$$

When τ is defined by $h \leqslant 4$ points, a similar argument shows that $M_{\leqslant k}(n) = O(n^2k^{h-3})$, for $h \geqslant 2$, and $M_{\leqslant k}(n) = O(n)$ for $h = 1$. Since $\mathsf{Pr}(\tau) = O(1/N_\tau^h)$, the analysis leads to the same upper bound as above. □

The combination of Corollary 3.3, Theorem 3.4, and Lemma 3.5 implies the following summary theorem.

Theorem 3.6. *The total expected number of discrete changes in the kinetic triangulation* $\mathcal{T}(P(t))$ *is* $O(n^2\beta_{s+2}(n)\log^2 n)$.

4. Kinetic maintenance of $\mathcal{T}(P)$

In this section we describe a kinetic data structure that supports efficient maintenance of $\mathcal{T}^+(P(t))$ under motion. The structure satisfies[4] the standard requirements of *efficiency, compactness, responsiveness*, and *locality*, as reviewed in the introduction.

[4] As in [5], all properties (except for compactness) hold *in expectation*, with respect to the random permutation π.

4.1. The static structure

We store the pseudo-triangulation tree $\mathcal{PT}^+(P(t))$ as a treap over P, as described in Theorem 2.2, whose inorder is the x-order of the points and where the heap order is according to their random priorities. Each node v in $\mathcal{PT}^+(P(t))$ corresponds to the pseudo-triangle τ_v whose apex is the point stored at v. We also store at v, as auxiliary data, the endpoints $\mathsf{left}(\tau_v)$ and $\mathsf{right}(\tau_v)$, which are inherited from appropriate ancestors of v.

In addition, we also store at v the combinatorial description of the funnel of τ_v, and of its triangulation $\mathcal{T}(\tau_v)$. This includes $\mathsf{bridge}(\tau_v)$, two ordered lists storing the vertices of $\mathcal{L}(\tau_v)$, and $\mathcal{R}(\tau_v)$ in their left-to-right order, and the list of the chords of $\mathcal{T}(\tau_v)$, sorted in their vertical order (i.e., the order of their intersections with the vertical line through $\mathsf{apex}(\tau_v)$). We represent every sorted list of vertices or edges[5] as a balanced binary tree supporting each of the operations search, split, and concatenate, in $O(\log n)$ time [19,24]. To facilitate efficient kinetic maintenance of $\mathcal{T}^+(P(t))$, we also store the vertices of the upper hull of P, in their left-to-right order in a balanced search tree. Note that each edge of the triangulation (not on $\mathcal{C}(P)$) appears twice in our structure, once as $\mathsf{bridge}(\tau_v)$ for some pseudo-triangle τ_v, and once on $\mathcal{L}(\tau_w)$ or $\mathcal{R}(\tau_w)$ for some ancestor w of v or on the convex hull of P.

Theorem 4.1. *Let P be a set of n points in the plane. The pseudo-triangulation tree $\mathcal{PT}^+(P(t))$, augmented with the auxiliary data items, as above, uses $O(n)$ space, and it can be initialized in $O(n\log n)$ time.*

Proof. The asserted bound on the overall storage follows from the easy observation that $\mathcal{PT}^+(P(t))$ contains $O(n)$ nodes, and every point $p \in P$ appears as a non-corner vertex on at most one chain $\mathcal{L}(\tau_v)$, $\mathcal{R}(\tau_v)$, over all nodes v of $\mathcal{PT}^+(P(t))$.

We construct the pseudo-triangulation tree $\mathcal{PT}^+(P(t))$ (excluding the auxiliary items $\mathsf{bridge}(\tau_v)$, $\mathcal{L}(\tau_v)$, $\mathcal{R}(\tau_v)$ and the chords of $\mathcal{T}(\tau_v)$) in a single *top–down* pass, which implements the recursive construction given in Section 2. Clearly, this can be done in $O(n)$ time, after an initial sorting of the points of P, by their x-coordinates and by their priorities; sorting the points takes $O(n\log n)$ time.

We next compute the items $\mathcal{L}(\tau_v)$, $\mathcal{R}(\tau_v)$, and $\mathsf{bridge}(\tau_v)$ stored at the nodes v of $\mathcal{PT}^+(P(t))$, by a single *bottom–up* traversal of $\mathcal{PT}^+(P(t))$, which computes for every node v the upper hull $\mathcal{U}(v)$ of the set $P_v \cup \{\mathsf{left}(\tau_v), \mathsf{right}(\tau_v)\}$. When we process a new non-leaf node v, we have already visited its respective left and right children v_ℓ and v_r, so their hulls $\mathcal{U}(v_\ell)$ and $\mathcal{U}(v_r)$ are already available. We compute $\mathsf{bridge}(\tau_v)$ in $O(\log n)$ time by a simultaneous binary search over $\mathcal{U}(v_\ell)$ and $\mathcal{U}(v_r)$, in the manner described in [20]. Then we use $\mathsf{bridge}(\tau_v)$ to split $\mathcal{U}(v_\ell)$ (resp., $\mathcal{U}(v_r)$) into $\mathcal{L}^-(\tau_v)$ and $\mathcal{L}(\tau_v)$ (resp., $\mathcal{R}(\tau_v)$ and $\mathcal{R}^+(\tau_v)$). We store explicitly the chains $\mathcal{L}(\tau_v)$ $\mathcal{R}(\tau_v)$ at v, and compute $\mathcal{U}(v)$ by concatenating the three edge lists $\mathcal{L}^-(\tau)$, $\{\mathsf{bridge}(\tau)\}$, and $\mathcal{R}^+(\tau)$, in a similar manner to that described in [5]. Overall, we spend $O(\log n)$ time at each node of $\mathcal{PT}^+(P(t))$, for a total of $O(n\log n)$ time.

Finally, for each node v in $\mathcal{PT}^+(P(t))$, we compute the list of chords of $\mathcal{T}(\tau_v)$ using the recursive mechanism described in Section 2. Recall that every non-corner vertex p of the funnel of τ_v generates exactly one edge e_p which recursively splits the unique sub-pseudo-triangle τ_p of τ_v. We process the non-corner vertices of $\mathcal{T}(\tau_v)$ in the increasing order of their priorities, and store the edges constructed so far in a list, in the order of their intersections with the vertical line through $\mathsf{apex}(\tau_v)$.

It takes $O(\log n)$ time to process a non-corner vertex p of τ_v, for a total of $O(n\log n)$ time. Indeed, we can determine the corners of τ_p in $O(\log n)$ time, by a binary search over the list of the previously generated edges. In addition, we can determine $\nu(p)$ by a binary search over the appropriate chain $\mathcal{L}(\tau_v)$ or $\mathcal{R}(\tau_v)$, obtain $\nu^*(p)$ in $O(1)$ additional time, and insert the chord $p\nu^*(p)$ into the list of chords in $O(\log n)$ time. $\square$

4.2. The kinetic certificates

To ensure the validity of $\mathcal{PT}^+(P(t))$ and its triangulation $\mathcal{T}^+(P(t))$, we use three types of *certificates*, denoted by CT, CE and CV. Each certificate is a predicate on a constant number of points. As long as all the certificates remain true, the validity of $\mathcal{PT}^+(P(t))$ and $\mathcal{T}^+(P(t))$ is ensured. Each certificate contributes a critical event to the global event priority queue $\mathcal{Q}$, which is the first future time (past the current time t at which the certificate is created) at which the certificate becomes invalid (if there is such a time).

4.2.1. CT-*certificates*

To ensure the validity of the tree $\mathcal{PT}^+(P(t))$ (ignoring the auxiliary data), each pair of points $p, q \in P$ with consecutive x-coordinates contributes a CT-certificate asserting that the order of $x(p)$ and $x(q)$ remains unchanged. This certificate fails at the first future moment of an x-swap between p and q. According to Lemma 2.1, CT-certificates (together with the chosen priorities) are sufficient to ensure the validity of the "bare" tree $\mathcal{PT}^+(P(t))$.

---

[5] Note that we do not store explicitly the tails $\mathcal{L}^-(\tau), \mathcal{R}^+(\tau)$, because the overall storage that they would require could be too large, as they can be shared by many pseudo-triangles.

4.2.2. CE-*certificates*

For each node v in $\mathcal{PT}^+(P(t))$, the edge $\mathsf{bridge}(\tau_v) = pq$ contributes a CE-certificate ensuring that the (current) neighbors of p and q on $\mathcal{L}^-(\tau_v) \cup \mathcal{L}(\tau_v)$ and $\mathcal{R}(\tau_v) \cup \mathcal{R}^+(\tau_v)$ remain below the line through p and q. This certificate involves[6] at most six points and fails at the first future time of collinearity between p, q, and one of their four neighbor vertices on $\mathcal{L}^-(\tau_v) \cup \mathcal{L}(\tau_v)$ and on $\mathcal{R}(\tau_v) \cup \mathcal{R}^+(\tau_v)$.

So far, we have ensured the validity of the tree $\mathcal{PT}^+(P(t))$ and of the edges $\mathsf{bridge}(\tau_v)$ stored at its nodes v. Moreover, the validity of all the chains $\mathcal{L}(\tau_v), \mathcal{R}(\tau_v)$ is also ensured because each one of their edges either belongs to $\mathcal{C}(P)$ or appears as $\mathsf{bridge}(\tau_w)$ at some descendant w of v. Here a collinearity between three consecutive points on $\mathcal{L}(\tau_v)$ or on $\mathcal{R}(\tau_v)$ (an envelope event) will be detected as a change in $\mathsf{bridge}(\tau_w)$, for the appropriate descendant w. Similarly, the validity of the upper hull of P follows since each of its edges either belongs to $\mathcal{C}(P)$ or appears as $\mathsf{bridge}(\tau_v)$ at some node v. See [5] and [20] for more details.

4.2.3. CV-*certificates*

It only remains to ensure the validity of the triangulations $\mathcal{T}(\tau_v)$, over all nodes $v \in \mathcal{PT}^+(P(t))$. For this we need the third type of certificates, denoted by CV. Fix a node v in $\mathcal{PT}^+(P(t))$. Every internal point p of $\mathcal{L}(\tau_v)$ or $\mathcal{R}(\tau_v)$ contributes a CV certificate ensuring the validity of $\nu(p)$. This certificate involves p, $\nu(p)$, and the two points adjacent to $\nu(p)$ on its chain. It fails when one of the points adjacent to $\nu(p)$ becomes collinear with p and $\nu(p)$.

Clearly, all of the above certificates use $O(n)$ storage, and can be initialized, including the construction of the event queue Q of their first failure times, by the algorithm of Theorem 4.1, without increasing its overall asymptotic running time, i.e., in $O(n \log n)$ time.

4.3. *Handling critical events*

We next describe the repair operations required when an event happens, that is, some certificate fails.

4.3.1. CT-*certificates*

Failure of a CT-certificate occurs at an x-swap. That is, the order of the x-coordinates of two consecutive points along $\mathcal{C}(P)$ switches, at some time $t = t_0$.

With no loss of generality we assume that $\mathsf{priority}(p) < \mathsf{priority}(q)$, implying that $q(t_0^-)$ is a descendant of $p(t_0^-)$, where t_0^-, t_0^+ denote the time just before and just after t_0, respectively. To update $\mathcal{PT}^+(P(t_0^+))$ we reconstruct from scratch the subtree rooted at the node v containing p, and recompute the kinetic certificates associated with its nodes and the points that they contain. We remove the failure times of the expired certificates from $\mathcal{Q}$, and insert the new ones. All this can be done in $O(n_v \log n)$ time using the algorithm of Theorem 4.1, where $n_v = |P_v|$. We prove that $\mathsf{E}\{n_v\} = O(\log n)$ by applying a simplified version of the analysis used in the proof of Proposition 3.2. As above, it suffices to show that $\mathsf{Pr}\{n_v > k\} \leqslant 4/k$, for any $k \geqslant 1$. Indeed, $n_v > k$ implies that either each of the $k/2$ points w whose x-coordinates immediately precede $x(p)$ or each of the $k/2$ points w whose x-coordinates immediately follow $x(p)$ at time t_0 satisfies $\mathsf{priority}(p) < \mathsf{priority}(w)$. This happens[7] with probability at most $4/k$. Thus, we can reconstruct the subtree rooted at v in $O(\log^2 n)$ expected time.

As can easily be checked, if neither p nor q is the leftmost or the rightmost point of P (excluding the points at infinity, which we added) then no further updates outside the subtree of v are needed, and no additional certificates need to be created or destroyed. (That is because the upper hull $\mathcal{U}(v)$ contains at most one of p, q, and it does not change as a result of the swap. Indeed, this later property can fail only when p and q are the two leftmost or two rightmost points of τ_v. This however is impossible because the leftmost and rightmost points of τ_v have smaller priorities than $\mathsf{priority}(p)$ (and $\mathsf{priority}(q)$).)

We next describe the necessary modifications in the setting, depicted in Fig. 6, in which case we assume that (i) p and q are the two points with the smallest x-coordinates, (ii) $x(q(t_0^-)) > x(p(t_0^-))$, and (iii) the y-coordinate of p is larger than that of q when they swap; the other cases are treated symmetrically. The x-swap between p and q causes q to appear on the upper hull of P, below and to the left of p. We add q to the upper hull in $O(\log n)$ time. Similarly, q becomes part of the tail $\mathcal{L}^-(\tau_w)$ of every ancestor w of v (both w and v lie on the leftmost path of the treap). If w is such an ancestor whose bridge is incident to p (from the right), then we have to incorporate q into the certificate of $\mathsf{bridge}(\tau_w)$, and possibly replace its old failure time in $\mathcal{Q}$ with a new one. Since the expected number of ancestors w of v in the treap $\mathcal{PT}^+(P(t))$ is $O(\log n)$ (see, e.g., [21]), any swap event can be processed in $O(\log^2 n)$ expected time.

4.3.2. CE-*certificates*

Consider a time t_0 when a CE-certificate at some node v fails. We assume without loss of generality that at time t_0 the leftmost vertex p of $\mathcal{L}(\tau_v)$ becomes collinear with the leftmost edge qr of $\mathcal{R}^+(\tau_v)$, so that $\mathsf{bridge}(\tau_v)$ was pq before the event and is pr afterwards, and treat the remaining cases symmetrically. See Fig. 7 for an illustration. As a result of

---

[6] If $\mathsf{bridge}(\tau_v)$ does not exist then we have an even simpler certificate which fails when the two edges of $\mathcal{L}^-(\tau_v), \mathcal{R}^+(\tau_v)$ incident to $\mathsf{apex}(\tau_v)$ become collinear.

[7] We emphasize again that arguments of this kind are based on the assumption that the motion of the points is oblivious to the choice of priorities.

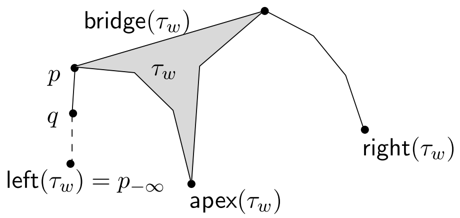

Fig. 6. The view after a swap event between a pair of points p, q with the smallest x-coordinates. Points p and q are stored in the left subtree of a node w, whose CE-certificate has to be updated.

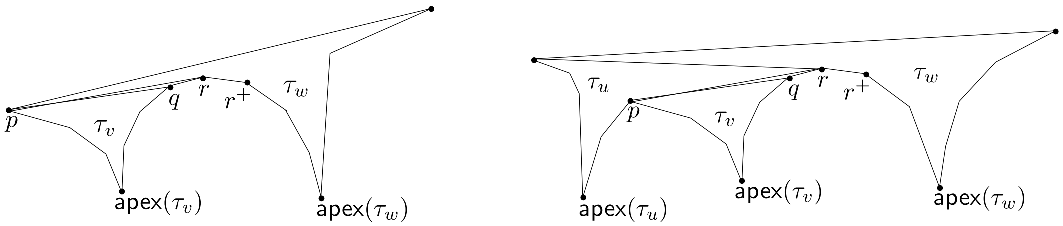

Fig. 7. Failure of the CE-certificate at v (shown at time t_0^+ right after the event). Left: The case where the ancestor w that stores rr^+ coincides with the ancestor u that has lost q. Right: The case where u is distinct from w.

this event, the edge pr replaces pq as bridge(τ_v), the edge qr is added to the end of $\mathcal{R}(\tau_v)$, and the triangulation $\mathcal{T}(\tau_v)$ gains the new triangle $\triangle pqr$. We need $O(\log n)$ time to update the edge lists of $\mathcal{R}(\tau_v)$ and $\mathcal{T}(\tau_v)$, and to compute the CV-certificate of q (which ceases to be the endpoint of $R(\tau_v)$) and add its failure time to $\mathcal{Q}$. (Note that the CV-certificate of q is part of the former CE-certificate at v.)

To recompute the new certificate of bridge(τ_v), we have to determine the next edge rr^+ of $\mathcal{R}^+(\tau_v)$ that is incident to r from the right. This edge is either stored in one of the lists $\mathcal{L}(\tau_w)$ or $\mathcal{R}(\tau_w)$ at some ancestor w of v, or it belongs to the upper hull of P. See Fig. 7 (left). We find rr^+ by doing a binary search on the lists $\mathcal{L}(\tau_w)$ and $\mathcal{R}(\tau_w)$ for the ancestors w of v, and if necessary also on the convex hull of P.

If pq and qr were part of the upper hull at time t_0^-, we replace them by a single edge pr, in $O(\log n)$ time. Otherwise, v has some ancestor u such that pq and qr are stored in the edge list of $\mathcal{L}(\tau_u)$ or $\mathcal{R}(\tau_u)$. (There is exactly one such ancestor u, which is equal to w unless r is incident to bridge(τ_u); see Fig. 7. In the terminology of Section 3, τ_u experiences an envelope event at time t_0.) We find u in $O(\log^2 n)$ expected time by searching the edge lists stored at all ancestor nodes of v, whose expected number is bounded by $O(\log n)$. We then replace pq and qr by pr in the edge list of the respective chain $\mathcal{L}(\tau_u)$ or $\mathcal{R}(\tau_u)$, and remove from $\mathcal{Q}$ the failure time of the CV-certificate of q (within τ_u). Moreover, we have to retriangulate a suitable sub-pseudo-triangle τ_0 of τ_u whose boundary, according to Proposition 3.2, has expected complexity $O(\log n)$ (see also Fig. 5). To do so, we first determine τ_0, by locating the edge $q\nu^*(q)$ in the edge list of $\mathcal{T}(\tau_u)$, and then looking for the lowest (resp., highest) edge above (resp., below) $q\nu^*(q)$ that is generated by a vertex whose priority is smaller than priority(q). We then recursively triangulate τ_0, as described in the proof of Theorem 4.1. All this can be done in $O(\log^2 n)$ expected time. Therefore, we can process any CE-certificate failure in $O(\log^2 n)$ expected time.

4.3.3. CV-*certificates*

We finally consider the case when a visibility event, involving some point p within the funnel of τ_v, for some node v of $\mathcal{PT}^+(P(t))$, causes the failure of the corresponding CV-certificate at some time t_0. Since the failed certificate is associated with an internal vertex of $\mathcal{L}(\tau_v)$ or $\mathcal{R}(\tau_v)$, all the necessary updates are local to the funnel of τ_v, and to its triangulation $\mathcal{T}(\tau_v)$. We update the CV-certificate of p and insert its new failure time into $\mathcal{Q}$, in $O(\log n)$ time (the new neighbor of $\nu(p)$ is easily obtained in $O(\log n)$ from the respective edge list). In addition, we may have to determine and re-triangulate a suitable sub-pseudo-triangle τ_0 of τ_v, whose boundary has expected complexity $O(\log n)$ (see Proposition 3.2). As in the case of a failure of a CE-certificate, this can be done in $O(\log^2 n)$ expected time, by searching the edge list of $\mathcal{T}(\tau_v)$.

We thus obtain our main theorem.

Theorem 4.2. *Let $P(t)$ be a collection of n moving points, as above. We can maintain the triangulation $\mathcal{T}(P(t))$ under motion in a kinetic data structure of linear size, which processes an expected number of $O(n^2\beta_{s+2}(n)\log n)$ events, each in $O(\log^2 n)$ expected time, where s is the maximum number of times that any single triple of points of $P(t)$ can become collinear.*

4.4. *Enforcing locality*

As implied by Theorem 4.2, the proposed data structure for maintaining $\mathcal{T}(P)$ is compact, efficient, and responsive, where the last two properties hold in expectation. (Efficiency is implied by the fact that the convex hull of P may change

$\Omega(n^2)$ times; see, e.g., [1].) To make it also local (in expectation), it is sufficient to ensure that at any moment of time the expected number of kinetic certificates involving any single point is $O(\log n)$. Clearly, each point is associated with at most two CT-certificates. Since the expected depth of $\mathcal{PT}^+(P(t))$ is $O(\log n)$ and each pseudo-triangle of $\mathcal{PT}^+(P(t))$ defines a single CE-certificate, each point p participates in an expected number of $O(\log n)$ CE-certificates (namely, only those whose pseudo-triangles τ are such that $p \in P_\tau$; these pseudo-triangles are associated with the ancestors of p in the treap).

We next slightly modify the definition of CV-certificates, in order to ensure that at any moment of time the total expected number of CV-certificates involving any point is also $O(\log n)$. Consider a fixed moment of time t_0 and a fixed node v in $\mathcal{PT}^+(P(t_0))$, and choose any vertex p on, say, the left chain $\mathcal{L}(\tau_v)$. Currently, p participates in a single certificate that it generates (ensuring the validity of $\nu(p)$), and in an arbitrary number of certificates generated by all the vertices q of $\mathcal{R}(\tau_v)$ satisfying $\nu(q) = p$. We modify our algorithm by keeping (i.e., storing in $\mathcal{Q}$ the failure times of) only the certificates of p that are generated by the leftmost and the rightmost such vertices q in $\mathcal{R}(\tau_v)$. If p lies on $\mathcal{R}(\tau_v)$, we act symmetrically. We apply this modification to every node v and every vertex of $\mathcal{L}(\tau_v) \cup \mathcal{R}(\tau_v)$. This modification does not affect the correctness of the kinetic data structure because, as can be easily checked, among all the CV-certificates involving p and points q with $\nu(q) = p$, the first to fail must be among the extreme ones that we keep.

Now, at each node v, every vertex of $\mathcal{L}(\tau_v), \mathcal{R}(\tau_v)$ participates in at most three CV-certificates. Since the expected depth of $\mathcal{PT}^+(P(t))$ is $O(\log n)$, the asserted (expected) locality bound follows (again, only CV-certificates at the ancestors of the involved point can be affected). The kinetic maintenance of this restricted set of CV-certificates resembles that of the original set, with the following minor modification. Each time when we process a visibility event caused by the failure of some CV-certificate, generated by a vertex p at some node v, we also have to recompute the CV-certificates involving the old and the new points $\nu(p)$. This can be done in $O(\log n)$ time using a binary search over $\mathcal{L}(\tau_v)$ or $\mathcal{R}(\tau_v)$, which does not affect the time bounds in Theorem 4.2.

5. Conclusion

The kinetic triangulation scheme presented in this paper is fairly easy to define and to maintain as the input points move, and it satisfies all the basic requirements from a kinetic data structure. Compared with the preceding scheme of Agarwal, Wang and Yu [4], we believe that our scheme is conceptually simpler. Moreover, it yields a slightly better bound on the number of discrete changes in the triangulation. We note that both schemes are randomized, and the bounds only hold in expectation. It would be interesting to design a deterministic kinetic triangulation scheme with similar performance bounds. Finally, as noted in the introduction, the significance of our result (as well as that of the preceding work [4]) is temporary, until the conjectured near-quadratic upper bound on the number of topological changes in the kinetic Delaunay triangulation is established. Nevertheless, given how hard is this latter problem, our result is probably "safe" for quite some time.

Acknowledgements

We thank the anonymous referees for valuable suggestions that helped us to improve the presentation.

References

[1] P.K. Agarwal, J. Basch, L.J. Guibas, J. Hershberger, L. Zhang, Deformable free-space tilings for kinetic collision detection, Int. J. Robotics Research 21 (2002) 179–197.
[2] P.K. Agarwal, J. Gao, L.J. Guibas, H. Kaplan, V. Koltun, N. Rubin, M. Sharir, Kinetic stable Delaunay graphs, in: Proc. 26th Annu. ACM Symp. on Comput. Geom., 2010, pp. 127–136.
[3] P.K. Agarwal, H. Kaplan, M. Sharir, Kinetic and dynamic data structures for closest pairs and nearest neighbors, ACM Trans. Algorithms 5 (2008), Article 4 (37 pp.).
[4] P.K. Agarwal, Y. Wang, H. Yu, A two-dimensional kinetic triangulation with near-quadratic topological changes, Discrete Comput. Geom. 36 (2006) 573–592.
[5] G. Alexandron, H. Kaplan, M. Sharir, Kinetic and dynamic data structures for convex hulls and upper envelopes, Comput. Geom. Theory Appl. 36 (2007) 144–158.
[6] J. Basch, J. Erickson, L.J. Guibas, J. Hershberger, L. Zhang, Kinetic collision detection between two simple polygons, Comput. Geom. Theory Appl. 27 (3) (2004) 211–235.
[7] J. Basch, L.J. Guibas, J. Hershberger, Data structures for mobile data, J. Algorithms 31 (1999) 1–28.
[8] M.W. Bern, D. Eppstein, Mesh generation and optimal triangulation, in: D.-Z. Du, F.K.-M. Hwang (Eds.), Computing Euclidean Geometry, 2nd edition, World Scientific, 1995, pp. 47–123.
[9] B. Chazelle, H. Edelsbrunner, M. Grigni, L.J. Guibas, J. Hershberger, M. Sharir, J. Snoeyink, Ray shooting in polygons using geodesic triangulations, Algorithmica 12 (1994) 54–68.
[10] K.L. Clarkson, P.W. Shor, Applications of random sampling in computational geometry, II, Discrete Comput. Geom. 4 (1989) 387–421.
[11] E.D. Demaine, J.S.B. Mitchell, J. O'Rourke, The open problems project, http://www.cs.smith.edu/~orourke/TOPP/.
[12] J. Donea, Arbitrary Lagrangian–Eulerian finite element methods, in: T.B. Belytschko, T.J.R. Hughes (Eds.), Computational Methods for Transient Analysis, North-Holland, Elsevier, 1983, pp. 474–516.
[13] H. Edelsbrunner, Triangulations and meshes in computational geometry, Acta Numer. 9 (2000) 133–213.
[14] H. Edelsbrunner, E.P. Mücke, Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms, ACM Trans. Graphics 9 (1990) 66–104.
[15] J.-J. Fu, R.C.T. Lee, Voronoi diagrams of moving points in the plane, Internat. J. Comput. Geom. Appl. 1 (1994) 23–32.

[16] L.J. Guibas, J.S.B. Mitchell, T. Roos, Voronoi diagrams of moving points in the plane, in: Proc. 17th Internat. Workshop Graph-Theoret. Concepts Computer Science, in: Lecture Notes in Comput. Sci., vol. 570, Springer-Verlag, 1991, pp. 113–125.
[17] L.J. Guibas, Kinetic data structures: A state of the art report, in: Robotics: The Algorithmic Perspective, WAFR 1998, A.K. Peters, Natick, MA, 1998, pp. 191–209.
[18] Ø. Hjelle, M. Dæhlen, Triangulations and Applications, Springer-Verlag, Berlin, 2009.
[19] K. Mehlhorn, Data Structures and Algorithms 1: Sorting and Searching, Springer-Verlag, Berlin, 1984.
[20] M.H. Overmars, J. van Leeuwen, Maintenance of configurations in the plane, J. Comput. System Sci. 23 (2) (1981) 166–204.
[21] R. Seidel, C.R. Aragon, Randomized search trees, Algorithmica 16 (1996) 464–497.
[22] M. Sharir, P.K. Agarwal, Davenport–Schinzel Sequences and Their Geometric Applications, Cambridge University Press, New York, 1995.
[23] D. Kirkpatrick, J. Snoeyink, B. Speckmann, Kinetic collision detection for simple polygons, Internat. J. Comput. Geom. Appl. 12 (1–2) (2002) 3–27.
[24] R.E. Tarjan, Data Structures and Network Algorithms, Society for Industrial and Applied Mathematics, Philadelphia, 1983.