Kinetic Geodesic Voronoi Diagrams in a Simple Polygon

Matias Korman
Department of Computer Science, Tufts University, Medford, MA, USA
matias.korman@tufts.edu

André van Renssen
School of Computer Science, University of Sydney, Australia
andre.vanrenssen@sydney.edu.au

Marcel Roeloffzen
Department of Mathematics and Computer Science,
Eindhoven University of Technology, The Netherlands
m.j.m.roeloffzen@tue.nl

Frank Staals
Department of Information and Computing Sciences, Utrecht University, The Netherlands
f.staals@uu.nl

Abstract

We study the geodesic Voronoi diagram of a set S of n linearly moving sites inside a static simple polygon P with m vertices. We identify all events where the structure of the Voronoi diagram changes, bound the number of such events, and then develop a kinetic data structure (KDS) that maintains the geodesic Voronoi diagram as the sites move. To this end, we first analyze how often a single bisector, defined by two sites, or a single Voronoi center, defined by three sites, can change. For both these structures we prove that the number of such changes is at most $O(m^3)$, and that this is tight in the worst case. Moreover, we develop compact, responsive, local, and efficient kinetic data structures for both structures. Our data structures use linear space and process a worst-case optimal number of events. Our bisector KDS handles each event in $O(\log m)$ time, and our Voronoi center handles each event in $O(\log^2 m)$ time. Both structures can be extended to efficiently support updating the movement of the sites as well. Using these data structures as building blocks we obtain a compact KDS for maintaining the full geodesic Voronoi diagram.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases kinetic data structure, simple polygon, geodesic voronoi diagram

Digital Object Identifier 10.4230/LIPIcs.ICALP.2020.75

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `http://arxiv.org/abs/2002.05910`.

Funding *Matias Korman*: M.K. was partially supported by MEXT KAKENHI No. 17K12635 and the NSF award CCF-1422311.
André van Renssen: A.V.R. was supported by JST ERATO Grant Number JPMJER1201, Japan.
Marcel Roeloffzen: M.R. was supported by JST ERATO Grant Number JPMJER1201, Japan and the Netherlands Organisation for Scientific Research (NWO) under project no. 628.011.005.
Frank Staals: F.S. was supported by the Netherlands Organisation for Scientific Research under project no. 612.001.651.

Acknowledgements We would like to thank Man-Kwun Chiu and Yago Diez for interesting discussions during the initial stage of this research.

47th International Colloquium on Automata, Languages, and Programming (ICALP 2020).
Editors: Artur Czumaj, Anuj Dawar, and Emanuela Merelli; Article No. 75; pp. 75:1–75:17

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Polygons are one of the most fundamental objects in computational geometry. As such, they have been used for many different purposes in different contexts. Within the path planning community, polygons are often used to model different regions. A simple example is when we have a robot moving within a building: in such a case we model all possible locations that a robot can reach by a polygon (the walls or any obstacle in the way form the boundary of this polygon). Then, the goal is to find a path that connects the source point and the destination and that minimizes some objective function. There are countlessly many results that depend on the exact function used (distance traveled [10], time needed to reach [18], number of required turns [27], etc.) Paths that minimize distance are often called *geodesics*.

Two of the most fundamental problems in this setting are constructing *shortest path maps* and *augmented Voronoi diagrams*. A shortest path map (or SPM for short) is a partition of the space into regions so that points in the same region travel in the same way to the fixed source [10, 13]. The exact definition of "in the same way" depends on the exact problem setting, but it often means that paths are *combinatorially* the same, that is, they have the same internal vertices. Augmented Voronoi diagrams are a generalization of SPMs for the case in which we have more than one fixed source and we are interested in the topology of the path to the closest source [3]. See Fig. 1 for an illustration. These structures are of critical importance in obtaining efficient solutions to related problems such as finding center points, closest pairs, nearest neighbors, and constructing spanners [22, 23].

It often happens that while we are moving to our destination, that destination is also moving. For example, when two agents try to meet, one wants to evade the other, or one simply needs to meet up with a second one that is doing a different task [17]. Since it is very costly to recompute the solution after each infinitesimal movement, the aim is to somehow maintain some information from which we can easily obtain the solution, and update this information only when the solution has significant changes. A data structure that can handle such a setting is known as a *kinetic data structure* (or KDS for short) [6]. There is a wide range of problems that have been studied in this setting. We refer to the survey by Basch et al. [6] for an overview of these results.

Surprisingly, there is very little work that combines all three of the above concepts (polygons, shortest paths, and kinetic data structures). We are aware of only two results. Aronov et al. [5] present a KDS for maintaining the shortest path map of a single point

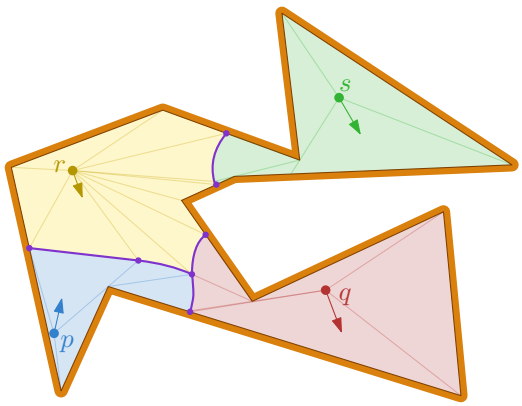


Figure 1 The (augmented) geodesic Voronoi diagram of four moving sites p, q, r, and s.

moving inside a simple polygon, and Karavelas and Guibas [14] give a KDS to maintain a *constrained Delaunay triangulation* of a set of moving points. This allows them to maintain nearest neighbors and the geodesic hull.

We present the first KDS to maintain the full (augmented) geodesic Voronoi diagram of a set of point sites moving inside a simple polygon, thus generalizing the above results. We carefully analyze when and how often it can change. To this end, we prove tight bounds on the number of combinatorial changes in a single bisector, and on the trajectory of a Voronoi center. Our results provide an important tool for maintaining related structures in which the agents (sites) move linearly within the simple polygon (e.g. minimum spanning trees, nearest-neighbors, closest pairs, etc.).

Related Work. Our data structures are based on the Kinetic Data Structures (KDS) framework introduced by Basch et al. [6]. In this framework motions are assumed to be known in advance. Each KDS maintains a set of *certificates* that together certify that the KDS currently correctly represents the target structure. Typically these certificates involve a few objects each and represent some simple geometric primitive. For example a certificate may indicate that three points form a clockwise oriented triangle. As the points move these certificates may become invalid, requiring the KDS to update. This requires repairing the target structure and creating new certificates. Such a certificate failure is called an (*internal*) *event*. An event is *external* if the target structure also changes. The performance of a KDS is measured according to four measures. A KDS is considered *compact* if it requires little space, generally close to linear, *responsive* if each event is processed quickly, generally polylogarithmic time, *local* if each site participates in few events, and *efficient* if the ratio between external and internal events is small, generally polylogarithmic. Note that for efficiency it is common to compare the worst-case number of events for either case.

Let S be a set of n point sites moving linearly in a space P, that is, each point moves with a fixed speed and direction. The *Voronoi diagram* $\mathrm{VD}_P(S)$ of S is a partition of P into n regions, one per site $s \in S$, such that for any point q in such a region $\mathcal{V}_s$ is closer to s than to any other site from S. Guibas et al. [11] studied maintaining the Voronoi diagram in case $P = \mathbb{R}^2$ and distance is measured by the Euclidean distance. They prove that $\mathrm{VD}_{\mathbb{R}^2}(S)$ may change $\Omega(n^2)$ times, and present an a KDS that handles at most $O(n^3\beta_4(n))$ events, each in $O(\log n)$ time. Here, $\beta_z(n) = \lambda_z(n)/n$ and $\lambda_z(n)$ is the maximum length of a Davenport-Schinzel sequence of n symbols of order z [25]. Their results actually extend to slightly more general types of movement. It is one of the long outstanding open problems if this bound can be improved [8, 9]. Only recently, Rubin [24] showed that if all sites move linearly and with the same speed, the number of changes is at most $O(n^{2+\varepsilon})$ for some arbitrarily small $\varepsilon > 0$. For arbitrary speeds, the best known bound is still $O(n^3\beta_4(n))$. When the distance function is specified by a convex k-gon the number of changes is $O(k^4n^2\beta_z(n))$ [2]. Here, and throughout the rest of the paper z denotes some small constant.

Let P be a simple polygon with m vertices, and let $\pi(s, q)$ be the shortest path between s and q that stays entirely inside P. We measure length of a path by the sum of the Euclidean edge lengths. Such a shortest path $\pi(s, q)$ is known as a geodesic, and its length as the *geodesic distance* between s and q. With some abuse of notation we use $\pi(s, q)$ to denote both the shortest path and its length.

Aronov was the first to study the geodesic Voronoi diagram [3]. He proved that when the sites in S are static, $\mathrm{VD}_P(S)$ has complexity $O(n + m)$. The same bound applies for the augmented geodesic Voronoi diagram. Moreover, he presented an $O((n+m)\log(n+m)\log n)$ time algorithm for constructing $\mathrm{VD}_P(S)$, which was improved to $O((n + m)\log(n + m))$ by

Table 1 The different types of events at which the geodesic Voronoi diagram changes, and their number. At an a, b-collapse event two vertices of $\mathrm{VD}_P(S)$ with degrees a and b collide and one disappears. Similarly, at an a, b-expand one such a vertex appears. At a vertex event a vertex of $\mathrm{VD}_P(S)$ collides with a vertex of P.

Event	Lower bound	Upper bound
$1,2$-collapse/expand	$\Omega(m^2 n)$	$O(m^2 n^2)$
$1,3$-collapse/expand	$\Omega(mn \min\{n, m\})$	$O(m^2 n^2 \min\{m\beta_z(n), n\})$
$2,2$-collapse/expand	$\Omega(m^3 n)$	$O(m^3 n \beta_4(n))$
$2,3$-collapse/expand	$\Omega(mn^2 + m^3 n)$	$O(m^3 n^2 \beta_4(n)\beta_z(n))$
$3,3$-collapse/expand	$\Omega(mn^2 + m^2 n)$	$O(m^3 n^3 \beta_z(n))$
vertex	$\Omega(m^2 n)$	$O(m^2 n \beta_4(n))$

Papadopoulou and Lee [23]. Recently, there have been several improved algorithms [16, 21] which ultimately lead to an optimal $O(m + n \log n)$ time algorithm by Oh [20]. Furthermore, Agarwal et al. [1] recently showed that finding the site in S closest to an arbitrary query point $q \in P$ – a key application of geodesic Voronoi diagrams – can be achieved efficiently even if sites may be added to, or removed from, S. Note however, that their result cannot be used directly to maintain a substructure of the Voronoi diagram (e.g. an MST).

There are no known results on maintaining an (augmented) geodesic Voronoi diagram when multiple sites S move continuously in a simple polygon P. In case there is only one site s, Aronov et al. [5] presented a KDS that maintains the shortest path map SPM_s of s. Their data structure uses $O(m)$ space, and processes a total of $O(m)$ events in $O(\log m)$ time each[1]. Karavelas and Guibas [14], provide a KDS to maintain a constrained Delaunay triangulation of S. This allows them to maintain the geodesic hull of S w.r.t. P, and the set of nearest neighbors in S (even in case P has holes). Their KDS processes $O((m+n)^3 \beta_z(n+m))$ events in $O(\log(n+m))$ time each.

Organization and Results. We present a kinetic data structure to maintain the geodesic Voronoi diagram $\mathrm{VD}_P(S)$ of a set S of n sites moving linearly inside a simple polygon P with m vertices. To this end, we prove a tight $O(m^3)$ bound on the number of combinatorial changes in a single bisector, and develop a compact, efficient, and responsive KDS to maintain it (Section 3). Our KDS for the bisector uses $O(m)$ space and processes events in $O(\log m)$ time. We then show that the movement of the Voronoi center c_{pqs} – the point equidistant to three sites $p, q, s \in S$ – can also change $O(m^3)$ times (Section 4). We again show that this bound is tight, and develop a compact, efficient, and responsive KDS to maintain c_{pqs}. The space usage is linear, and handling an event takes $O(\log^2 m)$ time. Both our KDSs can be made local as well, and therefore efficiently support updates to the movement of the sites. Building on these results we then analyze the full Voronoi diagram $\mathrm{VD}_P(S)$ of n moving sites (Section 5). We identify the different types of events at which $\mathrm{VD}_P(S)$ changes, and bound their number. Table 1 gives an overview of our bounds. We then develop a compact KDS to maintain $\mathrm{VD}_P(S)$. Omitted proofs are in the full version of this paper [15].

[1] The original description by Aronov et al. [5] uses a dynamic convex hull data structure that supports $O(\log^2 m)$ time queries and updates. Instead, we can use the data structure by Brodal and Jacob [7] which supports these operations in $O(\log m)$ time.

2 Preliminaries

We first review some properties of geodesic Voronoi diagrams and shortest path maps that we will use. Let SPM_s be the shortest path map of s, hence for all points in a region of SPM_s the shortest path from s has the same internal vertices. Each such region R is star-shaped with respect to the last internal vertex v on the shortest path. Often it will be useful to refine R into triangles incident to v. We refer to the resulting subdivision of P as the *extended* shortest path map. With some abuse of notation we will use SPM_s to denote this subdivision as well. An edge in SPM_s that starts in a vertex v that is colinear with the last edge in $\pi(s, v)$ is called an *extension segment* $E_{vs} = E_v$.

Let $\mathbb{T} = \mathbb{R}$ denote the time domain. We consider each site $s \in S$ as a function from $\mathbb{T}$ to P. For functions we will not distinguish between the function itself and its graph. We say that a function is *simple* if it is continuous, i.e. if it has no break points.

Given two sites p and q, the bisector B_{pq} is the set of all points that are equidistant to p and q. If no vertex of P lies on the bisector, then B_{pq} is a piecewise curve connecting two points on ∂P. Each curve on B_{pq} is a subarc of a hyperbola [3, 19].

▶ **Lemma 1** (Aronov [3]). *$\mathrm{VD}_P(S)$ consists of $O(n)$ vertices with degree 1 or 3, and $O(m)$ vertices of degree 2. For each degree 2 vertex v there is are $p, q \in S$ so that v lies on the bisector B_{pq} and v lies on extension segment of SPM_p or SPM_q. All edges of $\mathrm{VD}_P(S)$ are hyperbolic arc segments. Every vertex v of P contributes at most one extension segment E_v.*

▶ **Lemma 2** (Aronov et al. [5]). *Let s be a point moving linearly inside a simple polygon P with m vertices. The extended shortest path map SPM_s changes at most $O(m)$ times.*

▶ **Lemma 3.** *Let v be a vertex of P, there are $O(mn\beta_4(n))$ time intervals in which v has a unique closest site $s \in S$, and the distance from v to s over time is a hyperbolic function.*

3 A Single Bisector

Fix a pair of sites p and q, and let $b_{pq}(t)$ and $b_{qp}(t)$ be the endpoints of the bisector B_{pq} defined so that p lies to the right of $B_{pq}(t)$ when following the bisector from $b_{pq}(t)$ to $b_{qp}(t)$. As p and q move, the structure of B_{pq} changes at discrete times, or events. We distinguish between the following types of events (see Fig. 2):

- *vertex* events, at which an endpoint of B_{pq} coincides with a vertex of P,
- $1, 2$-*collapse* events, at which a degree 2 vertex (an interior vertex) of B_{pq} disappears as it collides with a degree 1 vertex (an endpoint),
- $1, 2$-*expand* events, at which a new degree 2 vertex appears from a degree 1 vertex,

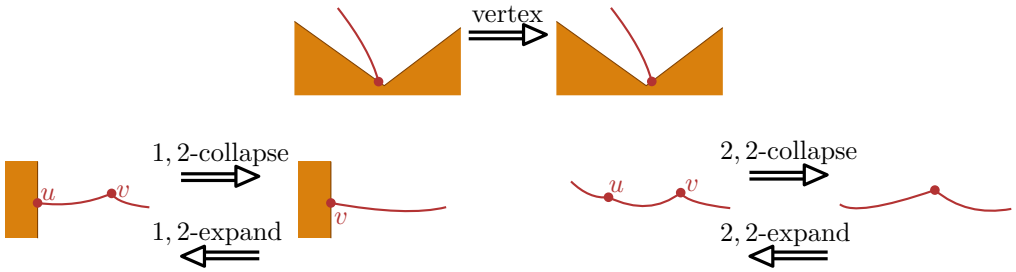


Figure 2 The types of events during which the structure of B_{pq} changes.

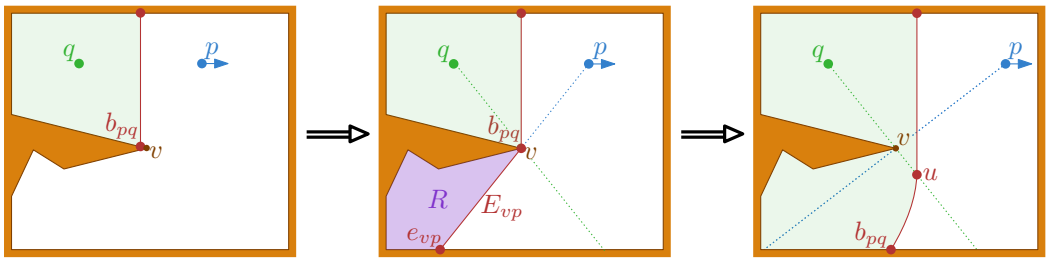


Figure 3 A vertex event at v may coincide with a $1,2$-expand event. At the time of the event all points in R are equidistant to p and q, and b_{pq} jumps from v to e_{vp}.

- $2,2$-*collapse* events at which a degree 2 vertex disappears by colliding with an other degree 2 vertex, and
- $2,2$-*expand* events, at which a new degree 2 vertex appears from a degree 2 vertex.

In Section 3.1 we prove that there are at most $O(m^2)$ vertex and $1,2$-collapse events, and at most $O(m^3)$ $2,2$-collapse events. The number of expand events can be similarly bounded. Some of these events may actually happen simultaneously. See for example Fig. 3, where B_{pq} changes when a vertex event and a $1,2$-expand event coincide. So we are double-counting these simultaneous events. Despite this, we show that our $O(m^3)$ bound on the number of changes of B_{pq} is tight in the worst case. In Section 3.2 we then argue that there is a KDS that can maintain B_{pq} efficiently.

3.1 Bounding the Number of Events

We start by showing that a bisector B_{pq} of p and q may change $\Omega(m^3)$ times. We then argue that there is also an $O(m^3)$ upper bound on the number of such changes.

▶ **Lemma 4.** *The bisector $B_{pq}(t)$ can change $\Omega(m^3)$ times.*

Proof. The main idea is to construct a bisector B_{pq}, a piecewise hyperbolic curve, of complexity $\Omega(m)$ in the middle of a region that consists of $\Omega(m^2)$ cells. These cells are defined by the extension segments in SPM_p and SPM_q that extend from the vertices on two convex chains of the polygon; one chain on either side of B_{pq}. See the top area in Fig. 4. In each cell the distance functions to p and q are different, thus if B_{pq} moves across a cell this causes a change in the bisector. The two upper convex chains in P have size $\Omega(m)$ and are placed such that as p moves towards the left (and q remains in place), the bisector sweeps from left to right over $\Omega(m^2)$ middle cells, thus causing $\Omega(m^2)$ changes to B_{pq}.

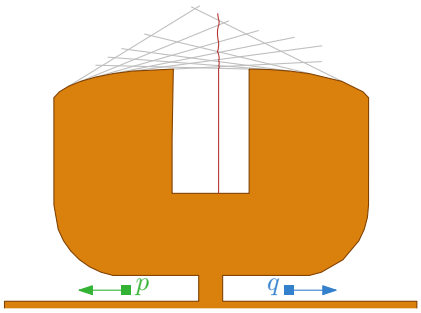


Figure 4 The bisector B_{pq} may be involved in $\Omega(m^3)$ $2,2$-collapse events.

Next, we argue that B_{pq} can be moved back and forth across these cells $\Omega(m)$ times by adding two convex chains of size $\Omega(m)$ to the bottom of the polygon, just above p and q. This way we can ensure that p and q alternate being the closest to the top of the polygon. Thus, when p is closest the bisector will move to the right and when q is the closest the bisector will move to the left. By making p and q move at the same speed and having the segments defining the convex chain on q's side start and end in the middle of where the segments of the convex chain on p's side, we can cause this alternation. It follows that the bisector sweeps over the $\Omega(m^2)$ middle cells $\Omega(m)$ times and thus it changes $\Omega(m^3)$ times. ◀

In the above proof one can observe that the counted events are only the 2,2-collapse events. For 2,2-collapse and 2,2-expand events there is also a $O(m^3)$ upper bound. For other events there is a $O(m^2)$ upper bound (proofs are in the full version [15]). Hence the bisector B_{pq} may change at most $O(m^3)$ times. Even though the entire bisector may change $O(m^3)$ times, the trajectories of its intersection points with the boundary of P have complexity at most $O(m^2)$. The following theorem summarizes these results.

▶ **Theorem 5.** *Let p and q be two points moving linearly inside P. The bisector B_{pq} of p and q can change $O(m^3)$ times. This bound is tight in the worst case. The trajectories of the endpoints of B_{pq} have $O(m^2)$ edges, each corresponding to a low-degree algebraic curve.*

3.2 A Kinetic Data Structure to Maintain a Bisector

We first describe a simple, yet naive, KDS to maintain B_{pq} that is not responsive and then show how to improve it to obtain a responsive KDS.

A Non-Responsive KDS to Maintain a Bisector. Our naive KDS for maintaining B_{pq} stores: (i) the extended shortest path maps of p and q using the data structure of Aronov et al. [5], (ii) the vertices of B_{pq}, ordered along B_{pq} from b_{pq} to b_{qp} in a balanced binary search tree, and (iii) for every vertex u of B_{pq}, the cell of SPM_p and of SPM_q that contains u. Since all cells in SPM_p and SPM_q are triangles, this requires only $O(1)$ certificates per vertex. We store these certificates in a priority queue $\mathcal{Q}$.

At any time where B_{pq} changes combinatorially (i.e. at an event) the shortest path to a vertex v of B_{pq} changes combinatorially, which indicates a change in the SPM cells that contain v. Hence, we detect all events. Conversely, when any vertex v of B_{pq} moves to a different SPM cell there is a combinatorial change in the bisector, so each event triggered by parts (ii) and (iii) of the KDS is an external event. The events at which SPM_p or SPM_q changes are internal (unless they also cause a change in a shortest path to a vertex of B_{pq}).

The changes to SPM_p and SPM_q are handled as in Aronov et al. and we update our other structures accordingly (see the full version for details [15]). This leads to a compact and efficient KDS to maintain the bisector. However, when the shortest path from p or q to some polygon vertex v changes this affects all bisector vertices whose shortest paths go through v. Hence, a single change in SPM_p or SPM_q may lead to a large number of updates to the certificates of bisector vertices. Therefore, the KDS is not responsive. To solve this issue we need a more refined data structure.

A Responsive KDS to Maintain a Bisector. First we dissect in some more detail the anatomy of a bisector. Each bisector consists of two endpoints which are degree 1 vertices and a chain of degree 2 vertices connecting them. We can further divide this chain based on which parts are directly visible from the sites defining the bisector. This division results in

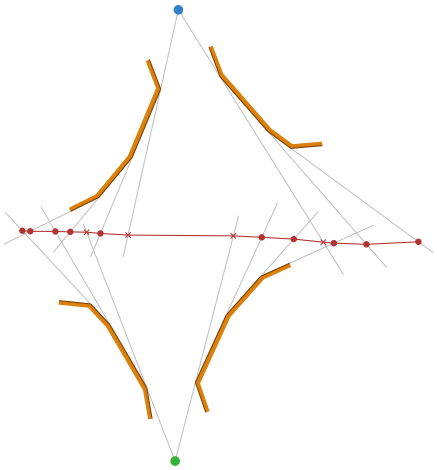


Figure 5 A bisector can be split into at most five pieces, here separated by degree 2 vertices marked as crosses.

at most 5 pieces, as illustrated in Fig. 5; some pieces may not be present in every bisector. First there is a *double-visible* piece that is visible from both sites p and q. Since P is a simple polygon, this piece consists of a single line segment. Adjacent to the double-visible piece on either side there may be a *single-visible* piece that is only visible to p or to q, but not both. Lastly, there are up to two *non-visible* pieces that are not directly visible from either p or q.

We will still store the bisector vertices in a balanced binary tree ordered along the bisector, but we will store the certificates for the degree 2 vertices a little differently. For each of the at most four degree 2 vertices that separate the pieces as well as the degree 1 endpoints, we store the cells of SPM_p and SPM_q that contain them. Then we observe that for internal vertices of the single-visible piece there can be no events. Each of these internal vertices lies on an extension segment of a single convex chain of vertices in the simple polygon and these extension segments do not intersect. Therefore no 2,2-collapses can occur.

The non-visible pieces are trickier, since 2,2-collapses may occur when a vertex moving on an extension segment of SPM_q moves to a different cell of SPM_p. Fortunately such potential events on a single non-visible bisector piece are related and form a strict ordering, regardless of the exact distance functions of the various vertices to p and q.

We define *event points* to be the locations at which 2,2-collapses that may occur. Consider two degree 2 vertices v and w that are internal to a non-visible piece of bisector between sites p and q, such that v and w are adjacent on the bisector and we have that v is on an extension segment of SPM_p and w is on an extension segment of SPM_q. Let the *event point* $\mathrm{ep}_{v,w}$ denote the intersection between these two extension segments. A 2,2-event between v and w corresponds to the event point being on the bisector between p and q. Without loss of generality assume that the event point currently lies in the Voronoi cell of p. We can then use the certificate $\pi(\mathrm{ep}_{v,w}, p) < \pi(\mathrm{ep}_{v,w}, q)$ to detect the 2,2-event between v and w. As we saw above maintaining these certificates explicitly is not efficient as any change in the shortest path towards p or q requires us to recompute the failure time. Instead we will store all event points in the Voronoi cell of p in one balanced binary tree ordered along the bisector and those in q in another. For each node in such a tree, we maintain the event point in its subtree that will be the first to be on the bisector, similar to a kinetic tournament (this, in turn requires maintaining distances between some of the relevant event points in the subtree). With this representation, we have to compute explicit failure times only for the two event points stored in the roots of the trees. Using these ideas we obtain the following result:

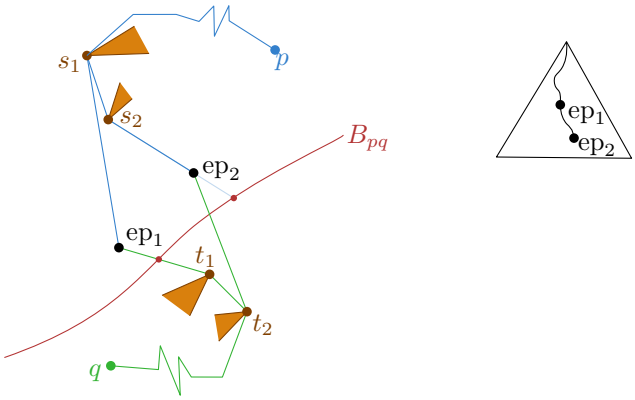


Figure 6 On the left is a schematic drawing of the event points ep_1 and ep_2 with their shortest paths towards p and q. On the right how (the certificates of) ep_1 and ep_2 are stored in the BST.

▶ **Theorem 6.** *Let p and q be two sites moving linearly inside a simple polygon P with m vertices. There is a KDS that maintains the bisector B_{pq} that uses $O(m)$ space and processes at most $O(m^3)$ events, each of which can be handled in $O(\log m)$ time. Additionally it can support movement changes of p and q in $O(\log m)$ time and splitting the bisector at any given vertex in $O(\log^2 m)$ time.*

Proof. For a single non-visible bisector piece between sites p and q, Consider event points ep_1 and ep_2 where ep_2 is a child of ep_1 in the tree. Let s_1 and t_1 denote the first polygon vertex on the shortest path from ep_1 towards p and q respectively and let s_2 and t_2 be defined symmetrically. See Fig. 6. Then we can rewrite the certificate for ep_1 as

$$\pi(\text{ep}_1, s_1)+\pi(s_1, p) < \pi(\text{ep}_1, t_1)+\pi(t_1, q) \quad \equiv \quad \pi(\text{ep}_1, s_1)-\pi(\text{ep}_1, t_1) < \pi(t_1, q)-\pi(s_1, p),$$

and the certificate for ep_2 similarly. Then observe that if $s_1 = s_2$ and $t_1 = t_2$, then ep_1 will be on the bisector before ep_2 if and only if $\pi(\text{ep}_1, s_1) - \pi(\text{ep}_1, t_1) > \pi(\text{ep}_2, s_2) - \pi(\text{ep}_2, t_2)$. This creates a strict ordering of the event points in the Voronoi cell of p. Unfortunately in many cases the first vertex on the path towards p or q will not be the same for every vertex on the bisector. Therefore we introduce an offset value to allow comparing event points that have different first vertices on their paths towards p and q.

If $s_1 \neq s_2$ and $t_1 \neq t_2$, we should compare based on a common node on the paths towards p and q, which may be any combination of s_1 or s_2 and t_1 or t_2. As these cases are analogous, we consider the case where s_1 and t_2 are on the shortest paths towards p and q respectively for both event points. (Intuitively s_1 and t_2 are further towards p and q). Now the values we would like to compare are $\pi(\text{ep}_1, s_1) - \pi(\text{ep}_1, t_2) > \pi(\text{ep}_2, s_1) - \pi(\text{ep}_2, t_2)$. However these are not what we stored. With some rewriting, we find that the above inequality holds if and only if

$$\pi(\text{ep}_1, s_1) - \pi(\text{ep}_1, t_1) > \pi(\text{ep}_2, s_2) - \pi(\text{ep}_2, t_2) - \pi(s_1, s_2) - \pi(t_1, t_2).$$

We call $-\pi(s_1, s_2) - \pi(t_1, t_2)$ the offset of ep_2 with respect to ep_1. Each node will store the maximum event value in its subtree as follows. For a leaf the maximum is its own event value. For an internal node, it is the maximum over its own event value and the maximum values of its children with the offset added. The maximum value of the root can then be used to determine the first time an 2, 2-event happens among the bisector vertices stored in the tree.

Note that the above data structure stores only a constant number of certificates directly involving p or q, all of which are stored at the root of the tree. Therefore, it can be made to support changes in the movement of p and q in $O(\log m)$ time. Furthermore, we can support splitting the bisector at a vertex in $O(\log^2 m)$ time, since a split affects $O(\log m)$ nodes in the balanced binary search tree, and recomputing the offsets (and thus updating the certificates) takes $O(\log m)$ time per node.

By replacing part (iii) of the naive structure with this data-structure we are still guaranteed to detect all events, but now when SPM_p changes, we have to update only a constant number of certificates (rather than $\Theta(m)$). As the certificates are stored in a binary tree it is easy to add or remove vertices when the bisector is expanded or shrinks. This proves the theorem. ◀

4 A Voronoi Center

Let $c_{pqs}(t)$ be the point equidistant to $p(t)$, $q(t)$, and $s(t)$ if it exists. By Aronov et al. [4] (Lemma 2.3.5) there is indeed at most one such a point. We refer to c_{pqs} as the *Voronoi center* of p, q, and s. Note that there may be times at which c_{pqs} does not exist. We identify five types of events at which c_{pqs} may appear or disappear, or at which the movement of c_{pqs} can change (see Fig. 7). They are:

- $1,3$-*collapse* events in which c_{pqs} collides with the boundary of the polygon (in a bisector endpoint) and disappears from P,
- $1,3$-*expand* events in which c_{pqs} appears on the boundary of P as two bisector endpoints intersect, creating a point equidistant to all three sites,
- *vertex-events* where c_{pqs} appears or disappears strictly inside P, as two sites, say p and q, are equidistant to a vertex v that appears on the shortest paths to c_{pqs},
- $2,3$-*collapse* events where one of the geodesics from either p, q, or s to c_{pqs} loses a vertex,
- $2,3$-*expand* events where one of the shortest paths gains a new vertex.

Observe that, as the name suggests, at a $1,3$-collapse event the Voronoi center (a degree 3 vertex in $\mathrm{VD}_P(\{p,q,s\})$) disappears as it collides with the endpoint of a bisector (a degree 1 vertex). Similarly, at a $2,3$-collapse event a degree 2 vertex on one of the bisectors disappears as it collides with a degree 3 vertex (the Voronoi center c_{pqs}). As in case of the bisector, some of these events may coincide. In the next section, we bound the number of events, and thus the complexity of the trajectory of c_{pqs}. We then present a kinetic data structure to maintain c_{pqs} in Section 4.2.

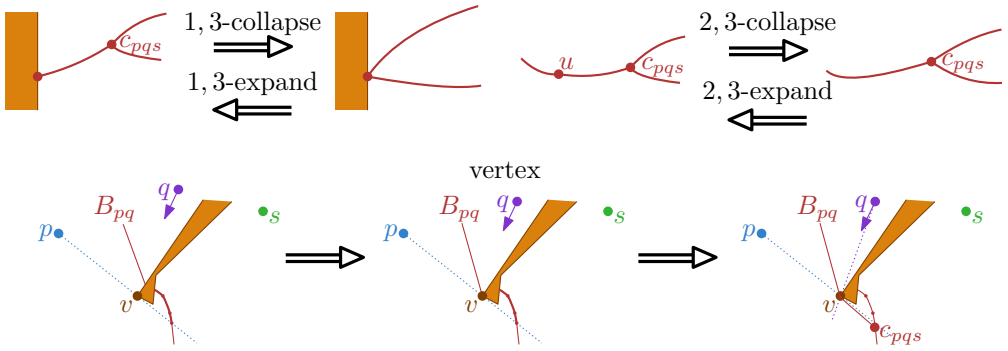


Figure 7 The events that can happen during the movement of a Voronoi center.

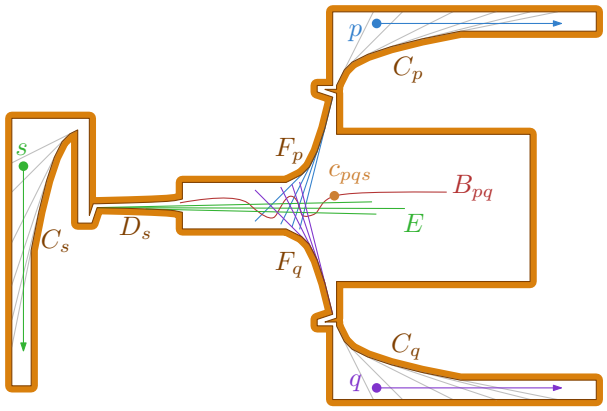


Figure 8 A polygon in which the trajectory of a voronoi center c_{pqs} has complexity $\Omega(m^3)$.

4.1 Bounding the Number of Events

We give a construction in which the trajectory of c_{pqs} has complexity $\Omega(m^3)$, and then prove a matching upper bound.

▶ **Lemma 7.** *The trajectory of the Voronoi center c_{pqs} of three points p, q, and s, each moving linearly, may have complexity $\Omega(m^3)$.*

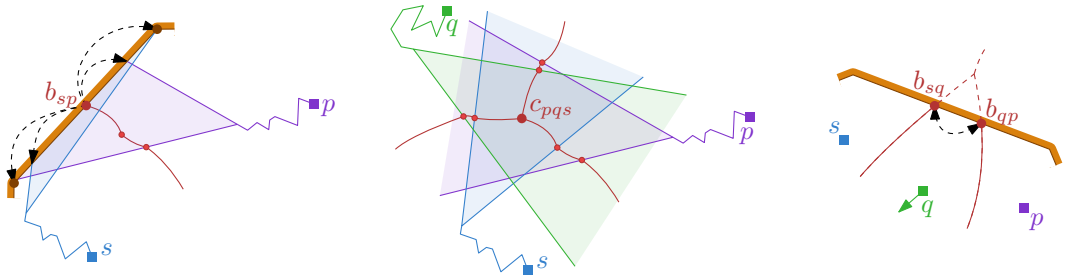
Proof. The main idea is that we can construct a trajectory for c_{pqs} of complexity $\Omega(m^2)$, even when two of the three sites, say p and q, are static. We place p and q so that their bisector B_{pq}, a piecewise hyperbolic curve of complexity $\Omega(m)$, intersects an (almost) horizontal line E $\Omega(m)$ times. We can realize this using two convex chains F_p and F_q in ∂P. See Fig. 8 for an illustration. We now construct a third convex chain D_s in ∂P and place the third site s so that the extension segments in SPM_s incident to the vertices of D_s all lie very close to E. Thus, each such segment intersects B_{pq} $\Omega(m)$ times. We choose the initial distances so that the voronoi center c_{pqs} lies on the rightmost segment of B_{pq}. Now observe that as s moves away from D_s, the center $c_{pqs}(t)$ will move to the left on B_{pq}, and thus it will pass over all $\Omega(m^2)$ intersection points of B_{pq} with the extension segments of the vertices in D_s. At each such time, the structure of one of the shortest paths $\pi(p(t), c_{pqs}(t))$, $\pi(q(t), c_{pqs}(t))$, or $\pi(s(t), c_{pqs}(t))$ changes (they gain or lose a vertex from F_p, F_q, or D_s, respectively). Hence, the trajectory of c_{pqs} changes $\Omega(m^2)$ times.

Next, we argue that we can make c_{pqs} "swing" back and forth $\Omega(m)$ times by having p and q move as well. The voronoi center c_{pqs} will then encounter every intersection point on B_{pq} $\Omega(m)$ times. It follows that the complexity of the trajectory of c_{pqs} is $\Omega(m^3)$ as claimed.

The idea is to add two additional convex chains, C_s and C_p, that make the bisector B_{ps} between p and s "zigzag" $\Omega(m)$ times throughout the movement of p and s. We can achieve this using a similar construction as in Lemma 4. To make sure that the bisector $B_{pq} = B_{pq}(t)$ between p and q remains static, we create a third chain C_q, which is a mirrored copy of C_p, and we make q move along a trajectory identical to that of p. See Fig. 8. Finally, observe that $c_{pqs}(t) = B_{pq} \cap B_{ps}(t)$, and thus $c_{pqs}(t)$ will indeed encounter all $\Omega(m^2)$ intersection points on B_{pq} $\Omega(m)$ times. The lemma follows. ◀

▶ **Lemma 8.** *The number of $1,3$-collapse events is at most $O(m^2)$.*

▶ **Theorem 9.** *The trajectory of the Voronoi center c_{pqs} has complexity $O(m^3)$. Each edge is a constant degree algebraic curve.*



(a) (b) (c)

Figure 9 In black the certificates that we maintain in order to detect: (a) events where b_{sp} changes movement, (b) $1,3$-collapse, $2,3$-collapse and $2,3$-expand events, and (c) $1,3$-expand events.

4.2 A Kinetic Data Structure to Maintain a Voronoi Center

Our KDS for maintaining c_{pqs} stores: (i) the extended shortest path maps of p, q, and s, (ii) the cells of these shortest path maps containing c_{pqs} (when c_{pqs} lies inside P), and (iii) the endpoints of all bisectors (for all pairs), and their cyclic order on ∂P. In particular, for each such endpoint b_{sp} we keep track of the cells of SPM_p and SPM_s that contain it. See Fig. 9. At any time we maintain $O(m)$ certificates, which we store in a global priority queue.

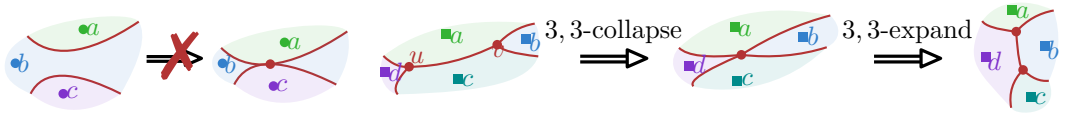
Observe that at $1,3$-collapse, $2,3$-collapse, and $2,3$-expand events the shortest path from c_{pqs} to one of the sites changes combinatorially. Hence, we can detect all such events. At a vertex event a vertex is equidistant to two sites, say p and q. At such a time, one of the two endpoints of B_{pq} leaves an edge of P, and thus exits a shortest path map cell in SPM_p (and SPM_q). Since we explicitly track all bisector endpoints, we can thus detect this vertex event of c_{pqs}. Finally, at every $1,3$-expand event two such bisector endpoints collide, and thus change their cyclic order along ∂P. We detect such events due to certificates of type (iii).

Any time at which c_{pqs} changes cells in a shortest path map the movement of c_{pqs} changes combinatorially. Hence, any failure of a certificate of type (ii) is an external event (a $1,3$-collapse, $2,3$-collapse, or $2,3$-expand). The certificates of types (i) and (iii) may be internal or external.

▶ **Theorem 10.** *Let p, q and s be three sites moving linearly inside a simple polygon P with m vertices. There is a KDS that maintains the Voronoi center c_{pqs} that uses $O(m)$ space and processes at most $O(m^3)$ events, each of which can be handled in $O(\log^2 m)$ time. Updates to the movement of p, q, and s, can be handled in $O(\log^2 m)$ time.*

Proof. Certificate failures of type (i) are handled exactly as described by Aronov et al. [5]. This takes $O(\log^2 m)$ time. Note that changes to the shortest path maps may affect the certificates that guarantee that c_{pqs} or a bisector endpoint lies in a particular SPM cell. In these cases we trigger a type (ii) or type (iii) certificate failure. At a certificate failure of type (ii) at which c_{pqs} exits a shortest path map cell, we remove all certificates of type (ii) from the event queue. Next, for each site p, q, and s, we compute the new cell in the shortest path map containing c_{pqs} (if c_{pqs} still lies inside P). Finally, we create the appropriate new type (ii) certificates. Since all cells have constant complexity, the total number of certificates affected is also $O(1)$. Computing them can easily be done in $O(\log^2 m)$ time.

Certificate failures of type (iii) where the movement of a bisector endpoint changes are handled using the same approach as in Section 3.2. Furthermore, at such an event we check if c_{pqs} appears or disappears, that is, if the event is actually a vertex event of c_{pqs}. This can be done in $O(\log^2 m)$ time [21]. If c_{pqs} disappears then we delete all type (ii) certificates.



(a) (b)

Figure 10 (a) Voronoi edges cannot intersect in their interior. (b) The 3, 3-collapse/expand events.

If c_{pqs} appears then we locate the cell of SPM_p, of SPM_q, and of SPM_s that contains c_{pqs}, and insert new type (ii) certificates that certify this. Finding the cells and updating the certificates can be done in $O(\log^2 m)$ time. At a certificate failure of type (iii) where two bisector endpoints collide, we check if the intersection point is equidistant to all three sites, and is thus a 1, 3-expand event. Similarly to the approach described above, we add new type (ii) certificates in this case. Again this takes $O(\log^2 m)$ time.

Maintaining the extended shortest path maps requires handling $O(m)$ events [5]. Events where c_{pqs} crosses a boundary of an extended SPM correspond to changes in the trajectory of c_{pqs}. By Theorem 9 there are at most $O(m^3)$ such events. This dominates the $O(m^2)$ events that we have to handle to maintain the bisector endpoints in cyclic order around ∂P (Theorem 5 and Lemma 8).

Since in addition to SPM_p, SPM_q, and SPM_s, we maintain only a constant amount of extra information. Since the KDS to maintain such a shortest path map SPM_s is local and can be updated to changes in the movement of s in $O(\log^2 m)$ time. The same applies for our data structure as well. Thus we obtain a compact, responsive, local, and efficient KDS. ◀

5 The Geodesic Voronoi Diagram

In this section we consider maintaining the geodesic Voronoi diagram $\mathrm{VD}_P(S)$ as the sites in S move. As a result of the sites in S moving, the Voronoi vertices and edges in $\mathrm{VD}_P(S)$ will also move. However, we observe that all events involving Voronoi edges involve their endpoints; two edges cannot start to intersect in their interior as this would split a Voronoi region, see Fig. 10(a). Similarly, the interior of a Voronoi edge cannot start to intersect the polygon boundary. This means we can distinguish the following types of events that change the combinatorial structure of the Voronoi diagram.

- Edge collapses, at which an edge between vertices u and v shrinks to length zero. Let d_u, d_v, with $d_u \leq d_v$, be the degrees of u and v, respectively. We then have a d_u, d_v-*collapse*.
- Edge expands. These are symmetric to edge collapses.
- Vertex events, where a degree 1 vertex of $\mathrm{VD}_P(S)$ crosses over a polygon vertex.

Indeed, we have seen most of these events when maintaining an individual bisector or Voronoi center (a degree 3 vertex in $\mathrm{VD}_P(S)$). The only new types of events are the 3, 3-collapse and 3, 3-expand events which involve two degree 3 vertices. They are depicted in Fig. 10.(b). We again note that some of these events may happen simultaneously.

▶ **Theorem 11.** *Let S be a set of n sites moving linearly inside a simple polygon P with m vertices. During the movement of the sites in S, the combinatorial structure of the geodesic Voronoi diagram $\mathrm{VD}_P(S)$ changes at most $O(m^3 n^3 \beta_z(n))$ times. In particular, the events at which $\mathrm{VD}_P(S)$ changes, and the number of such events, are listed in Table 1.*

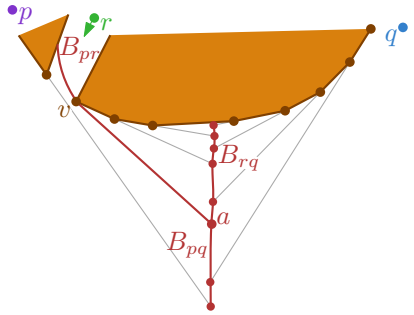


Figure 11 A vertex event may cause a bisector (here B_{pq}) to split, and a degree 3 vertex crossing an SPM extension segment may merge two bisectors.

We prove these bounds in the full version [15]. For most of the lower bounds we generalize the constructions from Sections 3 and 4. For the upper bounds we typically fix a site or vertex (or both), and map the remaining sites to a set of functions in which we are interested in the lower envelope. In Section 5.1 we develop a kinetic data structure to maintain $\mathrm{VD}_P(S)$.

5.1 A KDS for a Voronoi Diagram

In this section we develop a KDS to maintain the Voronoi diagram of S. Our KDS essentially stores for each site the extended shortest path map of its Voronoi cell, and a collection of certificates that together guarantee that the shortest paths from the sites to all Voronoi vertices remain the same (and thus the KDS correctly represents $\mathrm{VD}_P(S)$). The main difficulties that we need to deal with are shown in Fig. 11. Here, r becomes the site closest to vertex v, and as a result a part of the polygon moves from the Voronoi cell V_p of p to the Voronoi cell V_r of r. Our KDS should therefore support transplanting this region from the SPM representation of V_p into V_r or vice versa. Moreover, part of the bisector B_{pq} becomes a bisector B_{pr}, which means that any certificates internal to the bisector (such as those needed to detect 2,2-events) change from being dependent on the movement of p to being dependent on the movement of r. Next, we show how to solve the first problem, transplanting part of the shortest path map. Our KDS for the bisector from Theorem 6 essentially solves the second problem. All that then remains is to describe how to handle each event.

5.1.1 Maintaining Partial Shortest Path Maps

To support transplanting a part of SPM_s into SPM_q we extend the data structure of Aronov et al. [5]. Observe that SPM_s is a tree rooted at s, and we transplant only subtrees, rooted at some polygon vertex v. Our representation of SPM_s should support: (i) link operations in which we add the subtree rooted at v as a child of u, (ii) cut operations in which we cut an edge (u, v), (iii) shortest path queries in which we report the length of the shortest path from some vertex u to the root s, and (iv) principal-child queries in which we report the *principal child* c of some non-root node u. The principal child is the child of u for which the angle between $\overline{cu}$ and $\overline{up(u)}$, where $p(u)$ is the parent of u, is minimal. We need this operation to support updating the certificates of SPM_s[2]. To support these operations,

[2] Since the root is the only node storing a moving point, all certificates involve only nodes from the first three layers of the tree. Hence, it suffices to compute the principal child only for direct children of the root.

we store SPM_s twice: once in a link-cut tree [26] and once in an Euler tour tree [12]. Both these structures support link and cut operations in $O(\log m)$ time. The link-cut trees support query operations on node-to-root paths, and hence we use them to answer shortest path queries in $O(\log m)$ time (plus $O(k)$ time to report the actual path, if desired). The Euler tour trees support query operations on subtrees, and hence we use them to answer principal child queries. In particular, we maintain the children of u in cyclic order around u, starting with c. This way link and cut operations still take $O(\log m)$ time, and the principal child of u can be reported in constant time.

5.1.2 The data structure

The full KDS thus consists of an extended shortest path map for every Voronoi cell maintained as described above; and certificates for each degree three vertex, degree one vertex, and each bisector. For every degree three vertex c_{pqs} we maintain the cells of SPM_p, SPM_q and SPM_s that contain it and its distance to neighboring vertices. For every degree one vertex b_{pq}, we store the cells of SPM_p and SPM_q that contain it, which edge of P it is on, and if applicable its separation from neighboring degree one vertices on the same edge. For each bisector, we store the data structure of Theorem 6. Our data structure uses a total of $O(n+m)$ space.

It is not to difficult to see that this certificate structure captures all external events. For collapse and expand events involving degree three vertices we explicitly certify that the distance to its adjacent vertices is non-zero. For events involving degree one vertices we explicitly track which edge contains each such a vertex. This allows us to detect vertex events. Furthermore, we maintain distance of each degree one vertex to other degree one vertices on the same edge. Thus we can detect 1,3-expand events. Furthermore, we maintain which cells of the SPM the vertex is contained in, which allows us to detect 1,2-expand and $1,2$-collapse events. What remains are the 2,2-events. These are detected by the data structure of Theorem 6.

5.1.3 Handling events

Handling the events is similar to what we described in Sections 3.2 and 4.2. Hence, we describe only what is new or different here.

At all external-events we have to update the shortest path map representations of the Voronoi cells. In most cases, this involves adding or removing a single vertex to the shortest path map. This can easily be handled using local computations in $O(\log^2 m)$ time. In case of vertex events, we may have to move an entire region in SPM_s to SPM_p. Since all shortest paths in such a region go via the vertex involved, we can perform these updates in $O(\log^2 m)$ time using the above data structure.

Since there are now n sites, we maintain $O(n+m)$ certificates, and thus updating the event queue takes $O(\log(n+m))$ time. Furthermore, we now have multiple degree three vertices, and thus we have to handle $3,3$-collapse and expand events. These are handled in a similar fashion to the other events; we update the Voronoi regions, and compute new certificates certifying the movement of the vertices involved from scratch. All these updates can be done in $O(\log^2 m + \log n)$ time.

At a vertex event where p and r are equidistant to a vertex v, the region R that moves from SPM_p to SPM_r may now be bounded by a bisector B_{rq} rather than B_{pq} (see Fig. 11). Since, at the time of the event, the relevant parts of B_{pq} and B_{rq} coincide we can obtain the new part of B_{rq} by splitting B_{pq}, and updating the movement of the associated sites. In particular, replacing the function expressing the distance p to v by the distance from r to v. Our bisector KDS allows such updates in $O(\log^2 m)$ time.

Finally, we may have to update the certificates associated with the Voronoi vertices as a result of changes to the individual shortest path maps. For example, when a site s can no longer see polygon vertex v, this affects all Voronoi certificates of vertices for which the shortest path goes through v. While our KDS for the bisector (Theorem 6) can update the affected certificates of such a change efficiently, this unfortunately does not hold for the certificates associated with degree one or degree three vertices. Updating these requires $O(k(\log^2 m + \log n))$ time, where k is the number of neighbors of s in $\mathrm{VD}_S(P)$. It is an interesting open question to try and handle such events implicitly as well. We therefore obtain the following result:

▶ **Theorem 12.** *Let S be a set of n sites moving linearly inside a simple polygon P with m vertices. There is a KDS that maintains the geodesic Voronoi diagram $\mathrm{VD}_P(S)$ that uses $O(n+m)$ space and processes at most $O(m^3n^3\beta_z(n))$ events, each of which can be handled in $O(k(\log^2 m + \log n))$ time, where k is the number of neighbors of the affected Voronoi cell.*

References

1. Pankaj K. Agarwal, Lars Arge, and Frank Staals. Improved dynamic geodesic nearest neighbor searching in a simple polgyon. In *Proceedings of the 34th Annual Symposium on Computational Geometry*, volume 99 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 4:1–4:14. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018. doi:10.4230/LIPIcs.SoCG.2018.4.
2. Pankaj K. Agarwal, Haim Kaplan, Natan Rubin, and Micha Sharir. Kinetic voronoi diagrams and delaunay triangulations under polygonal distance functions. *Discrete & Computational Geometry*, 54(4):871–904, December 2015. doi:10.1007/s00454-015-9729-3.
3. Boris Aronov. On the Geodesic Voronoi Diagram of Point Sites in a Simple Polygon. *Algorithmica*, 4(1):109–140, 1989.
4. Boris Aronov, Steven Fortune, and Gordon Wilfong. The furthest-site geodesic voronoi diagram. *Discrete & Computational Geometry*, 9(3):217–255, March 1993.
5. Boris Aronov, Leonidas J. Guibas, Marek Teichmann, and Li Zhang. Visibility queries and maintenance in simple polygons. *Discrete & Computational Geometry*, 27(4):461–483, January 2002. doi:10.1007/s00454-001-0089-9.
6. Julien Basch, Leonidas J. Guibas, and John Hershberger. Data structures for mobile data. *J. Algorithms*, 31(1):1–28, 1999. doi:10.1006/jagm.1998.0988.
7. G. S. Brodal and R. Jacob. Dynamic planar convex hull. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, pages 617–626, November 2002. doi:10.1109/SFCS.2002.1181985.
8. Erik D. Demaine, Joseph S. B. Mitchell, and Joseph O'Rourke. The open problems project. http://maven.smith.edu/~orourke/TOPP/.
9. J. E. Goodman and J. O'Rourke. *Handbook of Discrete and Computational Geometry*. CRC Press series on discrete mathematics and its applications. Chapman & Hall/CRC, 2004.
10. Leonidas Guibas, John Hershberger, Daniel Leven, Micha Sharir, and Robert E. Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2(1):209–233, November 1987. doi:10.1007/BF01840360.
11. Leonidas J. Guibas, Joseph S. B. Mitchell, and Thomas Roos. Voronoi diagrams of moving points in the plane. In *Graph-Theoretic Concepts in Computer Science*, pages 113–125, Berlin, Heidelberg, 1992. Springer.
12. Monika R. Henzinger, Valerie King, and Valerie King. Randomized fully dynamic graph algorithms with polylogarithmic time per operation. *Journal of the ACM*, 46(4):502–516, July 1999. doi:10.1145/320211.320215.
13. John Hershberger and Subhash Suri. An Optimal Algorithm for Euclidean Shortest Paths in the Plane. *SIAM Journal on Computing*, 28(6):2215–2256, 1999.

14 Menelaos I. Karavelas and Leonidas J. Guibas. Static and kinetic geometric spanners with applications. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '01, pages 168–176, Philadelphia, PA, USA, 2001. Society for Industrial and Applied Mathematics.

15 Matias Korman, André van Renssen, Marcel Roeloffzen, and Frank Staals. Kinetic geodesic voronoi diagrams in a simple polygon. *CoRR*, abs/2002.05910, 2018. arXiv:2002.05910.

16 Chih-Hung Liu. A Nearly Optimal Algorithm for the Geodesic Voronoi Diagram of Points in a Simple Polygon. In *34th International Symposium on Computational Geometry (SoCG 2018)*, volume 99 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 58:1–58:14, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.SoCG.2018.58.

17 Anna Lubiw, Jack Snoeyink, and Hamideh Vosoughpour. Visibility graphs, dismantlability, and the cops and robbers game. *Computational Geometry*, 66:14–27, 2017. doi:10.1016/j.comgeo.2017.07.001.

18 J. S. B. Mitchell. Shortest paths among obstacles in the plane. In *Proceedings of the 9th Annual ACM Symposium on Computational Geometry*, pages 308–317, 1993.

19 Joseph S. B. Mitchell. A new algorithm for shortest paths among obstacles in the plane. *Annals of Mathematics and Artificial Intelligence*, 3(1):83–105, March 1991. doi:10.1007/BF01530888.

20 Eunjin Oh. Optimal algorithm for geodesic nearest-point voronoi diagrams in simple polygons. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 391–409. SIAM, 2019. doi:10.1137/1.9781611975482.25.

21 Eunjin Oh and Hee-Kap Ahn. Voronoi diagrams for a moderate-sized point-set in a simple polygon. *Discrete & Computational Geometry*, March 2019. doi:10.1007/s00454-019-00063-4.

22 Eunjin Oh, Jean-Lou De Carufel, and Hee-Kap Ahn. The 2-center problem in a simple polygon. In *Algorithms and Computation*, pages 307–317, Berlin, Heidelberg, 2015. Springer.

23 Evanthia Papadopoulou and Der-Tsai Lee. A New Approach for the Geodesic Voronoi Diagram of Points in a Simple Polygon and Other Restricted Polygonal Domains. *Algorithmica*, 20(4):319–352, 1998.

24 Natan Rubin. On kinetic delaunay triangulations: A near-quadratic bound for unit speed motions. *Journal of the ACM*, 62(3):25:1–25:85, June 2015. doi:10.1145/2746228.

25 Micha Sharir and Pankaj K Agarwal. *Davenport-Schinzel sequences and their geometric applications*. Cambridge university press, 1995.

26 Daniel D. Sleator and Robert Endre Tarjan. A data structure for dynamic trees. *Journal of Computer and System Sciences*, 26(3):362–391, 1983. doi:10.1016/0022-0000(83)90006-5.

27 Subhash Suri. A linear time algorithm with minimum link paths inside a simple polygon. *Computer Vision, Graphics and Image Processing*, 35(1):99–110, 1986. doi:10.1016/0734-189X(86)90127-1.