Data Structures for Fréchet Queries in Trajectory Data[*]

Mark de Berg Ali D. Mehrabi Tim Ophelders

Abstract

Let π be a trajectory in the plane, represented as a polyline with n edges. We show how to preprocess π into a data structure such that for any horizontal query segment σ in the plane and a subtrajectory between two vertices of π, one can quickly determine the Fréchet distance between σ and that subtrajectory. We provide data structures for these queries that need $O(n^2 \log^2 n)$ preprocessing time, $O(n^2 \log^2 n)$ space, and $O(\log^2 n)$ query time. If we are interested only in the Fréchet distance between the complete trajectory π and a horizontal query segment σ, we can answer these queries in $O(\log^2 n)$ time using only $O(n^2)$ space.

1 Introduction

Comparing the shapes of polygonal trajectories—or time series in general—is an important task that arises in many contexts and is an active line of research in computational geometry and several other research areas [12]. A basic question here is how one can formally compare two given trajectories and to measure how similar they are to each other. To this end, several similarity measures have been developed in the past, and the Fréchet distance [1] is probably the most popular one: it has been used in various applications including speech recognition [11], signature and handwriting recognition [13], geographic applications such as map-matching of vehicle tracking data [5], and moving object analysis [6]. The Fréchet distance is commonly described using the following "leash" metaphor: a man walks on one trajectory and has a dog on a leash on the other trajectory. Both man and dog can vary their speeds, but they may not walk backwards. The Fréchet distance between the two trajectories is the length of the shortest leash with which man and dog can walk from the beginning to the end of the respective trajectories.

There has been a vast amount of work on algorithmic aspects of similarity measures and the Fréchet distance in particular, and a complete review is beyond our possibilities here. (We refer an interested reader to [12] for a comprehensive discussion on this topic.)

Most of the existing works dealing with similarity measures for trajectories study the following algorithmic question: how quickly can one compute or approximate the similarity measure for two given trajectories? However, in several applications it is helpful to store the trajectories into a data structure that allows a user to quickly compute the similarity between trajectories and a query trajectory. The results in this paper contribute to this research direction.

Background. There are several papers that study the problem of designing data structures for querying a trajectory (or a set of trajectories), to find subtrajectories (or the subset of trajectories) that are similar to a given query trajectory with respect to Fréchet distance. Due to space limitations we are able to only highlight a few of the works.

De Berg *et al.* [3] showed how to store a trajectory π into a data structure such that, given a query segment σ and a threshold δ_{max}, one can count all subtrajectories of π whose Fréchet distance to σ is at most δ_{max}. However, their work has several drawbacks: (i) in addition to all the correct subtrajectories their data structure may include additional subtrajectories whose Fréchet distance to σ can be up to a factor $2+3\sqrt{2}$ times larger than δ_{max}, (ii) their data structure is a complicated multi-level structure which is difficult to implement and unlikely to be efficient in practice, and finally (iii) it is unclear how to actually report the subtrajectories in an efficient manner. In a closely related work Gudmundsson and Smid [10] studied a more general version of the problem (where the data structure stores a geometric tree instead of a trajectory and the query is also a trajectory), but their solution makes several assumptions on the input: the tree must be c-packed and the edges of the tree and query must be relatively long compared to δ_{max}. Along the same line of research, De Berg and Mehrabi [4] presented data structures for preprocessing a given trajectory π in the plane representing the movement of a player during a game, such that the following queries can be answered: given two points s and t in the plane, report all subtrajectories of π in which the player has moved in a more or less straight line from s to t. They consider two measures of straightness, namely *dilation* and *direction deviation*, and presented efficient and easy-to-implement data structures with fast construction procedures and provable space

[*]Department of Mathematics and Computer Science, TU Eindhoven, the Netherlands. The authors are supported by the Netherlands Organization for Scientific Research (NWO) under grants 024.002.003, 612.001.118, and 639.023.208, respectively.

and error guarantees. As far as we are aware, the work by Driemel and Har-Peled [8] is the most related work to our work. They presented a data structure for preprocessing a trajectory π such that given a query trajectory σ with k vertices and two vertices p and q of π, one is able to approximate the Fréchet distance between σ and the subtrajectory of π from p to q, up to a constant factor. In this paper we study the same problem as Driemel and Har-Peled, but our goal is to compute the *exact* Fréchet distance. To be able to still obtain fast query times, we restrict our attention to the special case where σ is a horizontal segment.

Contributions. We study the problem of preprocessing a trajectory π with n edges in the plane into a data structure that is able to quickly answer the following type of queries: given a horizontal line segment σ in the plane and two vertices p, q of π, compute the Fréchet distance between σ and the subtrajectory of π from p to q. Our main result states that such a π can be preprocessed, affording $O(n^2 \log^2 n)$ time, into a data structure such that the desired queries can be answered in $O(\log^2 n)$ time. The data structure needs $O(n^2)$ space if p, q are the endpoints of π, and needs $O(n^2 \log^2 n)$ space otherwise.

2 Preliminaries

For a point p in the plane, we denote its coordinates by $(p.x, p.y)$. For two point sets P and Q, let $d_{\vec{H}}(P,Q) := \sup_{p\in P} \inf_{q\in Q} \|p-q\|$ be the directional Hausdorff distance from P to Q.

Let $p_0, p_1, \ldots, p_n$ be a sequence of $n+1$ points in the plane. We denote the polyline (or trajectory) defined by this sequence by $\mathcal{P}(p_0, \ldots, p_n)$. We generally view a polyline $\pi := \mathcal{P}(p_0, \ldots, p_n)$ as a piecewise-linear function. Namely, the function $\pi : [0,n] \to \mathbb{R}^2$ such that $\pi(i+t) := (1-t)p_i + tp_{i+1}$ for all $i \in \{0, \ldots, n-1\}$ and all $0 \le t \le 1$. With a slight abuse of notation, we sometimes also use π to denote the image of this function, which is a point set in $\mathbb{R}^2$; namely, the union over i of the segments between p_i and p_{i+1}.

For two polylines π and σ of n and m edges, respectively, their Fréchet distance is defined as

$$d_F(\pi,\sigma) := \inf_{\substack{\alpha:\,[0,1]\to[0,n] \\ \beta:\,[0,1]\to[0,m]}} \sup_{t\in[0,1]} \|\pi(\alpha(t)) - \sigma(\beta(t))\|,$$

where α and β range over continuous nondecreasing surjections. Given such α and β, the point $\pi(\alpha(t))$ is said to be *matched* with $\sigma(\beta(t))$.

Let ℓ_y be the horizontal line $\mathbb{R} \times \{y\}$. For two points p and q in the plane, the point on ℓ_y minimizing the maximum distance to p or q is the point where ℓ_y intersects the perpendicular bisector of the line segment from p to q, if the intersection lies inside the vertical strip

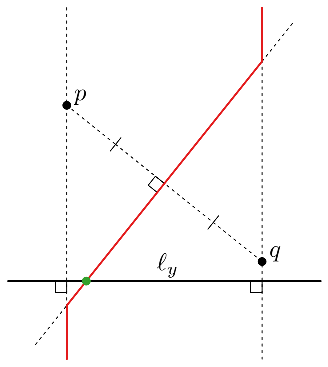

Figure 1: The point (green) on the line $\ell_y := \mathbb{R} \times \{y\}$ minimizing the distance to the farthest of p and q, and its trajectory (red) as y varies.

$[p.x, q.x]$; see Figure 1. Otherwise, it is either $(p.x, y)$ or $(q.x, y)$. Furthermore, we define the distance

$$B_{(p,q)}(y) = \min_{x\in\mathbb{R}} \max\{\|p-(x,y)\|, \|q-(x,y)\|\}$$

as the minimum distance over all points on ℓ_y to the farthest of p and q.

3 Fréchet distance to a segment

In this section, we show that the Fréchet distance between π and σ can be captured in an alternative expression $F_y(\pi,\sigma)$ in the special case where σ is a single (horizontal) segment. This alternative expression forms the basis for our data structures.

Let $\pi := \mathcal{P}(p_0, \ldots, p_n)$ be a polygonal trajectory of n edges in the plane. Let $x_0 \le x_1$ and $y \in \mathbb{R}$, and define $s_0 := (x_0, y)$ and $s_1 := (x_1, y)$, so that $\sigma := \mathcal{P}(s_0, s_1)$ is a horizontal segment in the plane. We can now define $F_y(\pi,\sigma)$ as follows:

$$\begin{aligned} F_y(\pi,\sigma) := \max\{&\|p_0 - s_0\|, \\ &\|p_n - s_1\|, \\ &d_{\vec{H}}(\pi,\sigma), \\ &\max_{i\le j,\ p_i.x \ge p_j.x} B_{(p_i,p_j)}(y)\}. \end{aligned} \tag{1}$$

The last term in this equation involves the term $B_{(p_i,p_j)}$ between pairs of vertices of π with $p_j.x \ge p_i.x$ even though p_j appears later than p_i along the trajectory (as $i \le j$). Note that σ is directed to the right, since we assume that $x_0 \le x_1$; so in a sense, these pairs (p_i, p_j) are those that "go backwards" relative to σ.

To show that $d_F(\pi,\sigma) = F_y(\pi,\sigma)$, we first show that $d_F(\pi,\sigma) \ge F_y(\pi,\sigma)$.

Lemma 1 *$d_F(\pi,\sigma) \geq \max_{i\leq j,\ p_i.x\geq p_j.x} B_{(p_i,p_j)}(y)$.*

Proof. Suppose not, then $d_F(\pi,\sigma) < B_{(p_i,p_j)}(y)$ for some $i \leq j$ with $p_i.x \geq p_j.x$. Let $d = d_F(\pi,\sigma)$, and let $e = B_{(p_i,p_j)}(y)$ for such i and j. Let $x^* = \arg\min_{x\in\mathbb{R}} \max\{\|p_i-(x,y)\|, \|p_j-(x,y)\|\}$. Then $e = \max\{\|p_i-(x^*,y)\|, \|p_j-(x^*,y)\|\} > d$. We have $p_i.x \geq x^* \geq p_j.x$ (otherwise one can decrease e by replacing x^* by $p_i.x$ or $p_j.x$). We have $\|p_i-(x',y)\| > d$ for all $x' \leq x^*$ and $\|p_j-(x',y)\| > d$ for all $x' \geq x^*$. Therefore, for any α and β, with $\|\pi(\alpha(t))-\sigma(\beta(t))\| \leq d$, we must have $\beta(t) > \beta(t')$ when $\alpha(t) = i$ and $\alpha(t') = j$. Hence α and β cannot both be monotone nondecreasing surjections. This contradicts that $d < e$. $\square$

Lemma 2 *$d_F(\pi,\sigma) \geq F_y(\pi,\sigma)$.*

Proof. For all nondecreasing surjections α and β, the point p_0 is matched with s_0, and p_n is matched with s_1. Therefore $d_F(\pi,\sigma) \geq \|p_0-s_0\|$ and $d_F(\pi,\sigma) \geq \|p_n-s_1\|$. As all points of π are matched with some point of σ, we have $d_F(\pi,\sigma) \geq d_{\overrightarrow{H}}(\pi,\sigma)$. Combining this with Lemma 1, we get that $d_F(\pi,\sigma) \geq F_y(\pi,\sigma)$. $\square$

To show that $d_F(\pi,\sigma) \leq F_y(\pi,\sigma)$ we use free space diagrams, which are a tool commonly used to compute the Fréchet distance. For a given distance threshold ε, the free space diagram $\mathcal{F}_\varepsilon(\pi,\sigma) = \{(a,b) \in [0,n]\times[0,1] \mid \|\pi(a)-\sigma(b)\| \leq \varepsilon\}$ indicates the pairs of points on π and σ that are at most distance ε apart. The product parameter space $[0,n]\times[0,1]$ consists of a row of n cells $[i,i+1]\times[0,1]$, each of which has a convex intersection with $\mathcal{F}_\varepsilon(\pi,\sigma)$. The Fréchet distance between π and σ is at most ε if and only if $(0,0) \in \mathcal{F}_\varepsilon(\pi,\sigma)$, $(n,1) \in \mathcal{F}_\varepsilon(\pi,\sigma)$ and for each $i < j$, there exists $0 \leq b \leq b' \leq 1$ such that (i,b) and (j,b') lie in $\mathcal{F}_\varepsilon(\pi,\sigma)$.

Lemma 3 *$d_F(\pi,\sigma) \leq F_y(\pi,\sigma)$.*

Proof. Let $d = F_y(\pi,\sigma)$ and let $\mathcal{F} = \mathcal{F}_d(\pi,\sigma)$. It suffices to show that $(0,0) \in \mathcal{F}$, $(n,1) \in \mathcal{F}$ and for each $i < j$, there exists $0 \leq b \leq b' \leq 1$ such that (i,b) and (j,b') lie in $\mathcal{F}$. Indeed, because $d \geq \|p_0-s_0\|$ and $d \geq \|p_n-s_1\|$, both $(0,0)$ and $(n,1)$ lie in $\mathcal{F}$.

We will show that for all $i < j$, there exist $0 \leq b \leq b' \leq 1$ for which (i,b) and (j,b') lie in $\mathcal{F}$. Because $d_{\overrightarrow{H}}(\pi,\sigma) \leq d$, we have for each i that some $(i,b) \in \mathcal{F}$. So let $b \geq 0$ be the minimum value for which $(i,b) \in \mathcal{F}$ and let $b' \leq 1$ be the maximum value for which $(j,b') \in \mathcal{F}$. We show that $b \leq b'$. Suppose for a contradiction that $b' < b$, then $p_j.x \leq \sigma(b').x < \sigma(b).x \leq p_i.x$. But then since $i < j$, we have $d \geq B_{(p_i,p_j)}(y)$. However, by convexity of free space cells, there is no value b'' for which both (i,b'') and (j,b'') lie in $\mathcal{F}$, so $B_{(p_i,p_j)}(y) > d$, which is a contradiction. $\square$

Theorem 4 follows readily from Lemmas 2 and 3.

Theorem 4 *Let $x_0 \leq x_1$ and $y \in \mathbb{R}$. Let $s_0 = (x_0,y)$ and $s_1 = (x_1,y)$ so that $\sigma = \mathcal{P}(s_0,s_1)$ is a horizontal segment in the plane. Let π be an arbitrary polygonal trajectory in the plane. Then $d_F(\pi,\sigma) = F_y(\pi,\sigma)$.*

4 The basic data structure

In this section we describe our data structure for the case where we want to compute the Fréchet distance from a horizontal query segment Q to the entire trajectory π. In the next section we then show how to generalize the solution to the case where a query also specifies two indices q, q', with $0 \leq q \leq q' \leq n$, and we want to compute the Fréchet distance between Q and the subtrajectory $\pi_{q,q'}$ of π. We use $\pi_{q,q'}$, for $0 \leq q \leq q' \leq n$, to denote the subtrajectory of π from vertex p_q to vertex $p_{q'}$.

Our data structure consists of three components each of which is based on one of the terms in Equation 1. For the first two terms of Equation 1, we simply store the endpoints p_0 and p_n of π, so that we can compute their distance to respectively s_0 and s_1 in constant time during the query procedure. To handle the third term of Equation 1, we provide the following lemma.

Lemma 5 *For a polyline $\pi := \mathcal{P}(p_0,\ldots,p_n)$ and a horizontal segment $\sigma = \mathcal{P}(s_0,s_1)$ with $s_0 := (x_0,y)$, $s_1 := (x_1,y)$ and $x_0 \leq x_1$, we have*

$$d_{\overrightarrow{H}}(\pi,\sigma) = \max\{\max_{p_i.x\in(-\infty,x_0]} \|s_0-p_i\|, \max_{p_i.x\in[x_1,+\infty)} \|s_1-p_i\|, \max_i |y-p_i.y|\}. \tag{2}$$

Proof. Recall that the directed Hausdorff distance from π to σ is the distance from the point on π farthest from σ. This distance is attained at a vertex p_i of π, because one of the endpoints of each edge of π is at least as far from σ as all points interior to that edge. If $p_i.x \leq x_0$, then its distance to σ is $\|p_i-s_0\|$. If $p_i.x \geq x_1$, then its distance to σ is $\|p_i-s_1\|$. Otherwise, the point on σ closest to p_i is $(p_i.x, y)$, at distance $|p_i.y-y|$. Since $|p_i.y-y| \leq \|p_i-s_0\|$ and $|p_i.y-y| \leq \|p_i-s_1\|$, the claim follows. $\square$

Lemma 5 suggests we compute $d_{\overrightarrow{H}}(\pi,\sigma)$ using a data structure D with the following components. The proof of Theorem 7 shows how these components are combined to answer a given query.

- We store the vertices of π, ordered by x-coordinate, in a balanced binary tree $T(\pi)$. For each node ν in $T(\pi)$, we store a farthest-point Voronoi diagram $FVD(\nu)$ on the vertices of π in the subtree rooted at ν.

- We let $top(\pi)$ and $bottom(\pi)$, respectively, store the topmost and the bottommost vertices of π.

It remains to deal with the last term in Equation 1. To this end, we first observe that for a fixed pair (p_i, p_j) of vertices of π with $i \leq j$ and $p_i.x \geq p_j.x$, the function $B_{(p_i,p_j)}(y)$ consists of two half-lines with slopes -1 and 1, respectively, and possibly a hyperbolic arc connecting their endpoints; see Figure 2. The hyperbolic arc captures the distances from p_i to the intersection of ℓ_y with the perpendicular bisector of the line segment from p_i to p_j. The two half-lines correspond to the distance to p_i and p_j, respectively. The endpoint of such a half-line lies at the value of y for which the perpendicular bisector intersects the vertical line through p_i, or p_j, respectively. As a consequence, computing the last term in Equation 1 for all possible values of y-coordinates of σ corresponds to computing the upper envelope of quadratically many hyperbolic arcs (and line segments); see Figure 3.

We let $\mathcal{E}(\pi)$ denote the upper envelope and we store it into a list $\mathcal{L}$. The list $\mathcal{L}$ represents $\mathcal{E}(\pi)$ as an ordered list of t pieces, where we next show that $t = O(n^2)$.

Lemma 6 *The complexity of $\mathcal{E}(\pi)$ is $O(n^2)$.*

Proof. We show that any two functions $B_{(p_i,p_j)}$ and $B_{(p_k,p_l)}$ intersect at most twice. Then according to Davenport-Schinzel sequences [9, Chapter 21] the complexity of $\mathcal{E}(\pi)$ will be linear in the number of hyperbolic arcs (and line segments), which is $O(n^2)$.

First recall that each function $B_{(p,q)}$ consists of three pieces: two half-lines of slopes of -1,+1, and one hyperbolic arc. The two hyperbolic arcs $arc(p_i, p_j)$ and $arc(p_k, p_l)$ of any two functions $B_{(p_i,p_j)}$ and $B_{(p_k,p_l)}$ intersect at most twice as they are quadratic functions. In

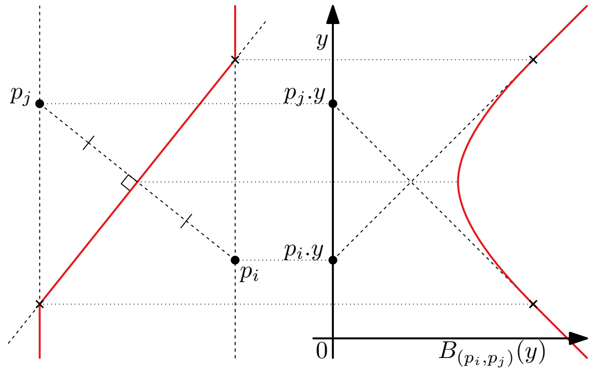

Figure 2: Left: the trajectory of the point on ℓ_y minimizing the distance to the farthest of p_i and p_j as y varies. Right: the distance from this point to the farthest of p_i and p_j as a function of y.

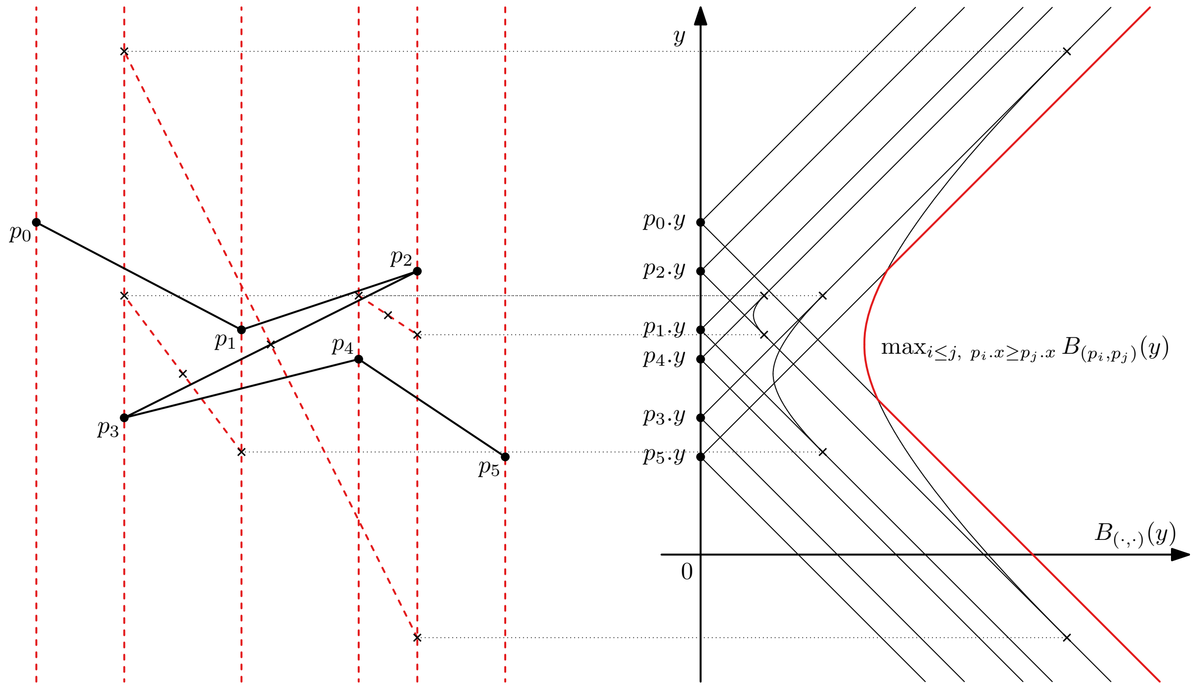

Figure 3: Left: a trajectory and all the perpendicular bisectors for pairs (p_i, p_j) with $i \leq j$ and $p_i.x \geq p_j.x$. Right: the corresponding hyperbolic arcs of the pairs p_i, p_j with $i \leq j$ and $p_i.x \geq p_j.x$. The arcs in red form the upper-envelope.

addition, the half-lines have slopes -1 and +1 and thus one can easily show that any two functions $B_{(p_i,p_j)}$ and $B_{(p_k,p_l)}$ intersect at most twice. $\square$

Putting everything together we obtain the following result.

Theorem 7 *Given a trajectory π with n edges in the plane, one can preprocess π into a data structure of size $O(n^2)$ such that queries that ask for the Fréchet distance between π and a given horizontal query line segment in the plane, can be answered in $O(\log^2 n)$ time. The preprocessing time of the data structure is $O(n^2 \log n)$.*

Proof. The preprocessing time of the data structure comes from the fact that it takes $O(t \log t)$ time to compute the upper envelope of t hyperbolas in the plane [9, Chapter 21]. The data structure consists of two data structures D and $\mathcal{L}$. The data structure D is essentially a balanced binary search tree in which each node ν stores a farthest-point Voronoi diagram on the vertices of π stored in the subtree rooted at ν. Since a farthest-point Voronoi diagram structure needs $O(n)$ space [2], for a point set of size $O(n)$, each level of the binary search tree uses $O(n)$ space and therefore D uses $O(n \log n)$ space in total as the binary search tree has $O(\log n)$ levels. In addition, since the list $\mathcal{L}$ uses $O(n^2)$ space, the overall space requirement of our data structure is $O(n^2)$.

A query with a segment $\sigma = [s_0, s_1]$ is answered as follows. First, compute the the distance between corresponding endpoints of π and σ. Second, compute the Hausdorff distance between π and σ using the different components of D: (i) Query the $FVD(\nu)$'s with point s_0, for canonical nodes ν in $T(\pi)$ whose union covers the range $(-\infty, s_0.x]$. Similarly query the $FVD(\nu)$'s with point s_1, for canonical nodes ν in $T(\pi)$ whose union covers the range $[s_1.x, +\infty)$. Maintain the maximum distance returned from the $O(\log n)$-many such queries. And, (ii) compute the maximum distance between the topmost and the bottommost vertices of π (stored in $top(\pi)$ and $bottom(\pi)$) and σ. Third, compute the intersection of ℓ_y and $\mathcal{E}(\pi)$ using a binary search on the quadratically many pieces of $\mathcal{E}(\pi)$ stored in $\mathcal{L}$. As the answer to the given query, return the maximum of the three values computed in the three steps mentioned above.

The correctness of the query procedure follows from Equation 1 and the query time is dominated by querying $O(\log n)$ farthest-point Voronoi diagrams each of which takes $O(\log n)$ time. $\square$

5 Querying Fréchet distance to subtrajectories

We will now refine our data structure to support queries for the Fréchet distance $d_F(\pi_{q,q'}, \sigma)$ between σ and a subtrajectory $\pi_{q,q'}$ between two vertices $p_q, p_{q'}$ of π. The query will, in addition to the horizontal segment σ, take two indices q, q' with $0 \le q \le q' \le n$.

In order to answer such a query, we build a data structure that, for each subtrajectory $\pi_{q,q'}$, can compute the terms of Equation 1 efficiently. The first two terms of this equation become $\|p_q - s_0\|$ and $\|p_{q'} - s_1\|$, respectively, and can, given a query (σ, q, q'), can be answered in constant time.

For the third term $d_{\overrightarrow{H}}(\pi_{q,q'}, \sigma)$, we build a balanced binary tree whose leaves represent the edges of π, ordered as they appear on π. Each node of this tree represents a subtrajectory $\pi_{i,j}$ of π consisting of the edges of π in that subtree. We store the indices i and j of the endpoints of this subtrajectory with the node. In addition, for each node, reuse the data structure we built based on Equation 2 for computing the Hausdorff distance from this subtrajectory to σ.

Given the indices q and q', one can then compute the Hausdorff distance from $\pi_{q,q'}$ to σ as follows. Search for the *maximal* nodes in the tree whose representative subtrajectory is contained in $\pi_{q,q'}$. Here, a node is maximal if there is no ancestor whose representative subtrajectory is contained in $\pi_{q,q'}$. The tree contains $O(\log n)$ such maximal nodes, and one can find them in $O(\log n)$ time given q and q'. Now, for each such node, query the Hausdorff distance from its representative subtrajectory to σ in $O(\log^2 n)$ time, and take the maximum of all the answers. We claim that this is the Hausdorff distance from $\pi_{q,q'}$ to σ. Indeed, each term of the maximum is a lower bound on $d_{\overrightarrow{H}}(\pi_{q,q'}, \sigma)$, and since the union of the subtrajectories represented by these nodes is exactly $\pi_{q,q'}$, the maximum is also an upper bound on $d_{\overrightarrow{H}}(\pi_{q,q'}, \sigma)$. So $d_{\overrightarrow{H}}(\pi_{q,q'}, \sigma)$ can be queried in $O(\log^2 n)$ time, using $O(n \log n)$ space and $O(n \log^2 n)$ preprocessing time.

It remains to build a data structure for the last term of Equation 1. For this we use a 2D range tree on the set $C := \{(i,j) \mid i \le j \text{ and } p_i.x \ge p_j.x\}$. That is, a balanced tree on the first coordinate of pairs in C, where for each subtree (say it is rooted at v), we store an additional tree $T'(v)$ on the second coordinate for the pairs of C in that subtree. $T'(v)$ is a balanced tree on the second coordinate, whose nodes store the upper envelope of the functions $B_{(p_i,p_j)}$ for the pairs $(i,j) \in C$ in their subtree. The upper envelope stored at the root v of a subtree $T'(v)$ of size k, requires $O(k)$ space, and it can be computed in $O(k)$ time by merging the envelopes of its children.

As $|C| = O(n^2)$, the complete data structure for the last term of Equation 1 uses $O(n^2 \log^2 n)$ space and one can build it in $O(n^2 \log^2 n)$ time.

Given a query (σ, q, q'), we can now use a range query that, in $O(\log^2 n)$ time, reports the upper envelopes of $O(\log^2 n)$ nodes, the upper envelope of whose union is exactly the upper envelope of the functions $B_{(p_i,p_j)}$ with $q \leq i \leq j \leq q'$. Instead of computing this envelope explicitly, we query each of the envelopes in $O(\log n)$ time at coordinate y, and take the maximum over the results for a total of $O(\log^3 n)$ time. The query time can be improved to $O(\log^2 n)$ time by applying fractional cascading [7].

Therefore, using $O(n^2 \log^2 n)$ preprocessing time and $O(n^2 \log^2 n)$ space, we can compute $d_F(\pi_{q,q'}, \sigma)$ in $O(\log^2 n)$ time for any query (σ, q, q').

6 Discussion

In this paper we showed how to preprocess a trajectory π with n edges in the plane into a data structure that is able to answer the following type of queries in $O(\log^2 n)$ time. Given two vertices p, q of π and a horizontal line segment σ in the plane, report the Fréchet distance between σ and the subtrajectory of π from p to q. The data structure can be constructed in $O(n^2 \log^2 n)$ time, it needs $O(n^2)$ space if p, q are the endpoints of π, and it needs $O(n^2 \log^2 n)$ space otherwise.

We conclude the paper by stating some interesting open questions:

- Can the quadratic upper bound of Lemma 6 for the complexity of the upper envelope be realized? If this upper bound is not tight, we might be able to reduce the space complexity of our data structure in Section 4.

- Can the data structure in Section 5 be extended to handle the case where p and q indicate a subtrajectory of π whose endpoints can lie in the interior of some edge of π?

- Can the data structure in Section 4 (or in Section 5) be extended to handle arbitrarily-oriented query segments in the plane?

References

[1] H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 5:75–91, 1995.

[2] M. de Berg and O. Cheong and M.v. Kreveld and M. Overmars. Computational Geometry: Algorithms and Applications (3rd edition). Springer-Verlag, 2008.

[3] M. de Berg and A.F. Cook and J. Gudmundsson. Fast Fréchet queries. *Computational Geometry: Theory and Applications*, 46:747–755, 2013.

[4] M. de Berg and A.D. Mehrabi. Straight-path queries in trajectory data. *Journal of Discrete Algorithms*, 36: 27–38, 2016.

[5] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On map-matching vehicle tracking data. *In Proc. 31th International Conference on Very Large Data Bases*, 853–864, 2005.

[6] K. Buchin, M. Buchin, J. Gudmundsson, M. Löffler and J. Luo. Detecting commuting patterns by clustering subtrajectories. *In Proc. 19th Annual International Symposium on Algorithmis and Computations*, 644–655, 2008.

[7] B. Chazelle and L. J. Guibas. Fractional Cascading: I. A data structuring technique. *Algorithmica*, 1:133–162, 1986.

[8] A. Driemel and S. Har-Peled. Jaywalking your dog: computing the Fréchet distance with shortcuts. *SIAM Journal on Computing*, 42:1830–1866, 2013.

[9] J.E. Goodman and J. O'Rourke. Handbook of Discrete and Computational Geometry. Second Edition, Chapman & Hall/CRC, 2004.

[10] J. Gudmundsson and M.H.M. Smid. Fast algorithms for approximate Fréchet matching queries in geometric trees. *Computational Geometry: Theory and Applications*, 48:479–494, 2015.

[11] S. Kwong and Q.H. He and K.F. Man and K.S. Tang and C.W. Chau. Parallel generic-based hybrid pattern matching algorithm for isolated word recognition. *Journal of Pattern Recognition and Artificial Intelligence*, 12:573–594, 1998.

[12] W. Meulemans. Similarity measures and algorithms for cartographic schematization. PhD Thesis. TU Eindhoven, 2014.

[13] E. Sriraghavendra and K. Karthik and C. Bhattacharayya. Fréchet distance based approach for searching online handwriting documents. *In Proc. 9th International Conference on Document Analysis and Recognition*, 461–465, 2007.