Computing the Fréchet Distance with a Retractable Leash

Kevin Buchin[1], Maike Buchin[2], Rolf van Leusden[1], Wouter Meulemans[1⋆], and Wolfgang Mulzer[3⋆⋆]

[1] Technical University Eindhoven, The Netherlands, k.a.buchin@tue.nl, r.v.leusden@student.tue.nl, w.meulemans@tue.nl
[2] Ruhr Universität Bochum, Germany, Maike.Buchin@ruhr-uni-bochum.de
[3] Freie Universität Berlin, Germany, mulzer@inf.fu-berlin.de

Abstract. All known algorithms for the Fréchet distance between curves proceed in two steps: first, they construct an efficient oracle for the decision version; then they use this oracle to find the optimum among a finite set of critical values. We present a novel approach that avoids the detour through the decision version. We demonstrate its strength by presenting a quadratic time algorithm for the Fréchet distance between polygonal curves in $\mathbb{R}^d$ under polyhedral distance functions, including L_1 and L_∞. We also get a $(1+\epsilon)$-approximation of the Fréchet distance under the Euclidean metric. For the exact Euclidean case, our framework currently gives an algorithm with running time $O(n^2 \log^2 n)$. However, we conjecture that it may eventually lead to a faster exact algorithm.

1 Introduction

Measuring the similarity of curves is a classic problem in computational geometry with many applications. For example, it is used for map-matching tracking data [3, 15] and moving objects analysis [5, 6]. In all these applications it is important to take the continuity of the curves into account. Therefore, the *Fréchet distance* and its variants are popular metrics to quantify (dis)similarity.

The Fréchet distance between two curves is defined by taking a homeomorphism between the curves that minimizes the maximum pairwise distance. It is commonly described using the *leash*-metaphor: a man walks on one curve and has a dog on a leash on the other curve. Both man and dog can vary their speeds, but they may not walk backwards. The Fréchet distance is the length of the shortest leash with which man and dog can walk from the beginning to the end of the respective curves.

Related work. The algorithmic study of the Fréchet distance was initiated by Alt and Godau [1]. For polygonal curves, they give an algorithm to solve the

⋆ Supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 639.022.707.
⋆⋆ Supported in part by DFG project MU/3501/1.

decision version in $O(n^2)$ time, and then use parametric search to find the optimum in $O(n^2 \log n)$ time. Several randomized algorithms have been proposed which are based on the decision version in combination with sampling possible values for the distance, one running in $O(n^2 \log^2 n)$ time [10] and the other in $O(n^2 \log n)$ time [13]. Recently, Buchin et al. [8] showed how to solve the decision version in subquadratic time, resulting in a randomized algorithm for computing the Fréchet distance in $O(n^2 \log^{1/2} n \log\log^{3/2} n)$ time. In terms of the leash-metaphor these algorithms simply give a leash to the man and his dog to try if a walk is possible. By cleverly picking the different leash-lengths, one then finds the Fréchet distance in an efficient way. Several algorithms exist to approximate the Fréchet distance (e.g. [2, 12]). However, these rely on various assumptions of the input curve; no approximation algorithm is known for the general case.

Contribution. We present a novel approach that does not use the decision problem as an intermediate stage. We give the man a "retractable leash" which can be lengthened or shortened as required. To this end, we consider monotone paths on the *distance terrain*, a generalization of the free space diagram typically used for the decision problem. Similar concepts have been studied before, but without the monotonicity requirement (e.g., [11] or the *weak* Fréchet distance [1]).

We show that it is sufficient to focus on the boundaries of cells of the distance terrain (defined by the vertices of the curves). It seems natural to propagate through the terrain for any point on a boundary the minimal "height" (leash length) ε required to reach that point. However, this may lead to an amortized linear number of changes when moving from one boundary to the next, giving a lower bound of $\Omega(n^3)$. We therefore do not maintain these functions explicitly. Instead, we maintain sufficient information to compute the lowest ε for a boundary. A single pass over the terrain then finds the lowest ε for reaching the other end, giving the Fréchet distance.

We present the core ideas for our approach in Section 2. This framework gives a choice of distance metric, but it requires an implementation of a certain data structure. We apply this framework to the Euclidean distance (Section 3) and polyhedral distances (Section 4). We also show how to use the latter to obtain a $(1+\epsilon)$-approximation for the former. This is the first approximation algorithm for the general case. We conclude with two open problems in Section 5.

2 Framework

2.1 Preliminaries

Curves and distances. Throughout we wish to compute the dissimilarity of two polygonal curves, P and Q. For simplicity, we assume that both curves consist of n segments. This represents the computational worst case; of course our algorithm can also cope with asymmetric cases. Both curves are given as piecewise-linear functions $P, Q\colon [0, n] \to \mathbb{R}^d$. That is, $P(i+\lambda) = (1-\lambda)P(i) + \lambda P(i+1)$ holds for any integer $i \in [0, n)$ and $\lambda \in [0, 1]$, and similarly for Q.

Let Ψ be the set of all continuous and nondecreasing functions $\psi\colon [0,n] \to [0,n]$ with $\psi(0) = 0$ and $\psi(n) = n$. Then the *Fréchet distance* is defined as

$$d_{\mathrm{F}}(P,Q) = \inf_{\psi\in\Psi} \max_{t\in[0,n]} \{\delta(P(t), Q(\psi(t)))\}.$$

Here, δ may represent any distance function between two points in $\mathbb{R}^d$. Typically, the Euclidean distance function is used; we consider this scenario in Section 3. Another option we shall consider are polyhedral distance functions (Section 4). For our framework, we require that the distance function is convex.

Distance terrain. Let us consider the joint parameter space $R = [0,n] \times [0,n]$ of P and Q. A pair $(s,t) \in R$ corresponds to the points $P(s)$ and $Q(t)$, and the distance function δ assigns a value $\delta(P(s), Q(t))$ to (s,t). We interpret this value as the "height" at point $(s,t) \in R$. This gives a *distance terrain* T, i.e., $T : R \to \mathbb{R}$ with $T(s,t) = \delta(P(s), Q(t))$. We segment T into n^2 *cells* based on the vertices of P and Q . For integers $i, j \in [0,n)$, the cell $C[i,j]$ is defined as the subset $[i, i+1] \times [j, j+1]$ of the parameter space. The cells form a regular grid, and we assume that i represents the column and j represents the row of each cell. An example of two curves and their distance terrain is given in Fig. 1.

A path $\pi : [0,1] \to R$ is called *bimonotone* if it is both x- and y-monotone. For $(s,t) \in R$, we let $\Pi(s,t)$ denote the set of all bimonotone continuous paths from the origin to (s,t). The *acrophobia function* $\widetilde{T} : R \to \mathbb{R}$ is defined as

$$\widetilde{T}(s,t) = \inf_{\pi\in\Pi(s,t)} \max_{\lambda\in[0,1]} T(\pi(\lambda)).$$

Intuitively, $\widetilde{T}(s,t)$ represents the lowest height that an acrophobic climber needs to master in order to reach (s,t) from the origin on a bimonotone path. Clearly, we have $d_{\mathrm{F}}(P,Q) = \widetilde{T}(n,n)$.

Let $x \in R$ and $\pi \in \Pi(x)$ be a bimonotone path from $(0,0)$ to x. Let ε be a value greater than zero. We call π an *ε-witness* for x if $\max_{\lambda\in[0,1]} T(\pi(\lambda)) \le \varepsilon$. We call π a *witness* for x if $\max_{\lambda\in[0,1]} T(\pi(\lambda)) = \widetilde{T}(x)$, i.e., π is an optimal path for the acrophobic climber.

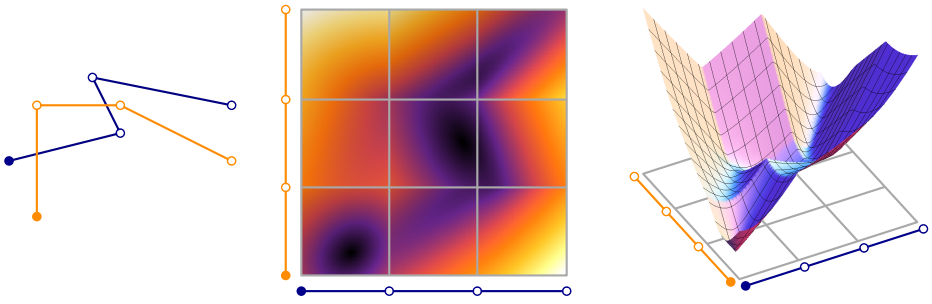

Fig. 1. Distance terrain with the Euclidean distance in $\mathbb{R}^2$. (left) Two curves. (middle) Cells as seen from above. Dark colors indicate low "height". (right) Perspective view.

2.2 Analysis of the distance terrain

To compute $\widetilde{T}(n,n)$, we show that it is sufficient to consider only the cell boundaries. For this, we generalize the fact that cells of the free space diagram are convex [1] to convex distance functions. For the proof, we refer to [7].

Lemma 2.1. *For a convex distance function δ and $\varepsilon \in \mathbb{R}$, the set of points (s,t) in a given cell $C[i,j]$ with $T(s,t) \leq \varepsilon$ is convex.*

Lemma 2.1 has two important consequences. First, it implies that it is indeed sufficient to consider only the cell boundaries. Second, it tells us that the distance terrain along a boundary is well-behaved. In this corollary and in the remainder of the paper, we refer to a function with a single local minimum as *unimodal*.

Corollary 2.2. *Let $C[i,j]$ be a cell of the distance terrain, and let x_1 and x_2 be two points on different boundaries of $C[i,j]$. For any y on the line segment x_1x_2, we have $T(y) \leq \max\{T(x_1), T(x_2)\}$.*

Corollary 2.3. *The distance along every boundary of a cell in distance terrain T is a unimodal function.*

For any cell $C[i,j]$, we denote with $L[i,j]$ and $B[i,j]$ its left and bottom boundary respectively (and their height functions in T). The right and top boundary are given by $L[i+1,j]$ and $B[i,j+1]$.[4] With $\widetilde{L}[i,j]$ and $\widetilde{B}[i,j]$ we denote the acrophobia function along the boundary. All these restricted functions have a single parameter in the interval $[0,1]$ that represents the boundary.

Assuming that the distance function δ is symmetric, computing values for rows and columns of T is symmetric as well. Hence, we present only how to compute with rows. If δ is asymmetric, our methods still work, but some extra care needs to be taken when computing distances.

Consider a vertical boundary $L[i,j]$. We use $\widetilde{L}^*[i,j]$ to denote the minimum of the acrophobia function $\widetilde{L}[i,j]$ along $L[i,j]$. An analogous definition is used for horizontal boundaries. Our goal is to compute $\widetilde{L}^*[i,j]$ and $\widetilde{B}^*[i,j]$ for all cell boundaries of the grid. We say that an ε-witness π *passes through* an edge $B[i,j]$, if there is a $\lambda \in [0,1]$ with $\pi(\lambda) \in B[i,j]$.

Lemma 2.4. *Let $\varepsilon > 0$, and let x be a point on $L[i,j]$. Let π be an ε-witness for x that passes through $B[a,j]$, for $1 \leq a < i$. Suppose further that there exists a column b with $a < b < i$ and $\widetilde{B}^*[b,j] \leq \varepsilon$. Then there exists an ε-witness for x that passes through $B[b,j]$.*

Proof. Let y be the point on $B[b,j]$ that achieves $\widetilde{B}^*[b,j]$, and let π_y be a witness for y. Since π is bimonotone and since π passes through $B[a,j]$, it follows that π must also pass through $L[b+1,j]$. Let z be the (lowest) intersection point, and π_z the subpath of π from z to x. Let π' be the path obtained by concatenating π_y, line segment yz, and π_z. By our assumption on ε and by Corollary 2.2, path π' is an ε-witness for x that passes through $B[b,j]$. □

[4] Note that there need not be an actual cell $C[i+1,j]$ or $C[i,j+1]$.

Lemma 2.4 implies that there are always *rightmost* witnesses for any point x on $L[i,j]$. For such witness, if it passes through $B[a,j]$ for some $a < i$, then $\widetilde{B}^*[b,j] > \widetilde{T}(x)$ for any $a < b < i$.

Corollary 2.5. *Let x be a point on $L[i,j]$. Then there is a witness for x that passes through some $B[a,j]$ such that $\widetilde{B}^*[b,j] > \widetilde{T}(x)$ for any $a < b < i$.*

Next, we argue that there is a witness for $\widetilde{L}^*[i+1,j]$ that enters row j at or after the horizontal boundary point used by the witness for $\widetilde{L}^*[i,j]$. In other words, the rightmost witnesses behave "monotonically" in the terrain.

Lemma 2.6. *Let π be a witness for $\widetilde{L}^*[i,j]$ that passes through $B[a,j]$, for a $1 \le a < i$. Then there exists a $a \le b \le i$ such that $\widetilde{L}^*[i+1,j]$ has a witness that passes through $B[b,j]$.*

Proof. Choose b maximal such that $\widetilde{L}^*[i+1,j]$ has a witness that passes through $B[b,j]$. Suppose $b < a$. Let π' be such a witness. We know that $\widetilde{L}^*[i+1,j] \ge \widetilde{L}^*[i,j]$, since π' passes through $L[i,j]$. However, we can now construct a witness for $\widetilde{L}^*[i+1,j]$ that passes through $B[a,j]$: follow π to $B[a,j]$ and then switch to the intersection of π' and $L[a+1,j]$. This contradicts the choice of b. □

We now characterize $\widetilde{L}[i,j]$ via a *witness envelope*, defined as follows. Fix a row j and two columns $a < i$. Suppose that $\widetilde{L}^*[i-1,j]$ has a witness that passes through $B[a',j]$ with $a' \le a$. The *witness envelope* for the column interval $[a,i]$ in row j is the upper envelope of the following functions on the interval $[0,1]$:

(i) the terrain function $L[i,j](\lambda)$;
(ii) the constant function $\widetilde{B}^*[a,j]$;
(iii) the constant function $\widetilde{L}^*[i-1,j]$ if $a \le i-2$;
(iv) the *truncated terrain functions* $\overline{L}[b,j](\lambda) = \min_{\mu \in [0,\lambda]} L[b,j](\mu)$ for $a < b < i$.

Lemma 2.7. *Fix a row j and two columns $a < i$ as above. Let $\alpha \in [0,1]$ and $\varepsilon > 0$. The point $x = (i, j+\alpha)$ has an ε-witness that passes through $B[a,j]$ if and only if (α,ε) lies above the witness envelope for $[a,i]$ in row j.*

Proof. Let π be an ε-witness for x that passes through $B[a,j]$. Then clearly $\varepsilon \ge \widetilde{B}^*[a,j]$ and $\varepsilon \ge L[i,j](\alpha)$. If $a \le i-2$, then π must pass through $L[i-1,j]$, so $\varepsilon \ge \widetilde{L}^*[i-1,j]$. Since π is bimonotone, it has to pass through $L[b,j]$ for $a < b < i$. Let $y_1 = (a+1, j+\alpha_1), y_2 = (a+2, j+\alpha_2), \ldots, y_k = (a+k, j+\alpha_k)$ be the points of intersection, from left to right. Then $\alpha_1 \le \alpha_2 \le \cdots \le \alpha_k \le \alpha$ and $\varepsilon \ge T(y_l) = L[a+l,j](\alpha_i) \ge \overline{L}[a+l,j](\alpha)$, for $l = 1, \ldots, k$. Hence (α,ε) is above the witness envelope.

Suppose (α,ε) is above the witness envelope. The conclusion follows directly if $a = i-1$. Otherwise, $\varepsilon \ge \widetilde{L}^*[i-1,j]$ holds. Let α' be such that the witness for $\widetilde{L}^*[i-1,j]$ that passes through $B[a',j]$ reaches $L[i-1,j]$ at point $(i-1, j+\alpha')$. If $\alpha \ge \alpha'$, we construct an appropriate ε-witness π' for x by following the witness for $\widetilde{B}^*[a,j]$, then passing to the witness for $\widetilde{L}^*[i-1,j]$ and then taking the

line segment to x. If $\alpha < \alpha'$, we construct a curve π' as before. However, π' is not bimonotone (the last line segment goes down). To fix this, let p and x be the two intersection points of π' with the horizontal line $y = j + \alpha$. We shortcut π' at the line segment px. The resulting curve π is clearly bimonotone and passes through $B[a, j]$. To see that π is an ε-witness, it suffices to check that along the segment px, the distance terrain never goes above ε. For this, we need to consider only the intersections of px with the vertical cell boundaries. Let $L[b, j]$ be such a boundary. We know that $L[b, j]$ is unimodal (Corollary 2.3) and let α^* denote the value where the minimum is obtained. By definition of the truncated terrain function, $L[b, j](\alpha) = \overline{L}[b, j](\alpha)$ if $\alpha \leq \alpha^*$. By assumption, the witness for $\widetilde{L}^*[i-1, j]$ passes $L[b, j]$ at α or higher. Hence, if $\alpha \geq \alpha^*$, then $\widetilde{L}^*[i-1, j] \geq L[b, j](\alpha)$. It follows that $\max\{\overline{L}[b, j](\alpha), \widetilde{L}^*[i-1, j]\} \geq L[b, j](\alpha)$. By definition $\varepsilon \geq \max\{\overline{L}[b, j](\alpha), \widetilde{L}^*[i-1, j]\}$ holds and thus $\varepsilon \geq L[b, j](\alpha)$. □

2.3 Algorithm

We are now ready to present the algorithm. We walk through the distance terrain, row by row, in each row from left to right. When processing a cell $C[i, j]$, we compute $\widetilde{L}^*[i+1, j]$ and $\widetilde{B}^*[i, j+1]$. For each row j, we maintain a double-ended queue (deque) Q_j that stores a sequence of column indices. We also store a data structure U_j that contains a set of (truncated) terrain functions on the vertical boundaries in j. It supports insertion, deletion, and a minimum-point query that, given up to two additional constants, returns the lowest point on the upper envelope of the terrain functions and the given constants.

The data structures fulfill the following invariant. Suppose that $\widetilde{L}^*[i, j]$ is the rightmost optimum we have computed so far in row j, and suppose that a rightmost witness for $\widetilde{L}^*[i, j]$ passes through $B[a, j]$. A point (α, β) *dominates* a point (γ, δ) if $\alpha > \gamma$ and $\beta \leq \delta$. Then Q_j stores the first coordinates of the points in the sequence $(a, \widetilde{B}^*[a, j]), (a+1, \widetilde{B}^*[a+1, j]), \ldots, (i-1, \widetilde{B}^*[i-1, j])$ that are not dominated by any other point in the sequence. Furthermore, the structure U_j stores the terrain functions for the boundaries from column $a+1$ to i. We maintain analogous data structures for each column i.

The algorithm proceeds as follows (see Algorithm 1): since $(0, 0)$ belongs to any path through the distance terrain, we initialize $C[0, 0]$ to use $(0, 0)$ as its lowest point and compute the distance accordingly. The left- and bottommost boundaries of the distance terrain are considered unreachable. Any path to such a point also goes through the adjacent horizontal boundaries or vertical boundaries respectively. These adjacent boundaries therefore ensure a correct result.

In the body of the for-loop, we compute $\widetilde{L}^*[i+1, j]$ and $\widetilde{B}^*[i, j+1]$. Let us describe how to find $\widetilde{L}^*[i+1, j]$. First, we add index i to Q_j and remove all previous indices that are dominated by it from the back of the deque. We add $L[i+1, j]$ to upper envelope U_j. Let h and h' be the first and second element of Q_j. We perform a minimum query on U_j in order to find the smallest ε_α for which a point on $L[i+1, j]$ has an ε_α-witness that passes through $B[h, j]$. By Lemma 2.7, this query requires the height at which the old witness enters the

Algorithm 1 FrechetDistance(P, Q, δ)

Input: P and Q are polygonal curves with n edges in $\mathbb{R}^d$;
δ is a convex distance function in $\mathbb{R}^d$
Output: Fréchet distance $d_{\mathrm{F}}(P, Q)$

{We show computations only within a row, column computations are analogous}
1: $\widetilde{L}^*[0,0] \leftarrow \delta(P(0), Q(0))$
2: $\widetilde{L}^*[0,j] \leftarrow \infty$ for all $0 < j < n$
3: For each row j, create empty deque Q_j and upper envelope structure U_j
4: **for** $j \leftarrow 0$ **to** $n-1$; $i \leftarrow 0$ **to** $n-1$ **do**
5: Remove any values x from Q_j with $\widetilde{B}^*[x,j] \geq \widetilde{B}^*[i,j]$ and append i to Q_j
6: **if** $|Q_j| = 1$ **then** Clear U_j
7: Add $L[i+1,j]$ to U_j
8: Let h and h' be the first and second element in Q_j
9: $(\alpha, \varepsilon_\alpha) \leftarrow U_j$.minimumQuery$(\widetilde{L}^*[i,j], \widetilde{B}^*[h,j])$
10: **while** $|Q_j| \geq 2$ **and** $\widetilde{B}^*[h',j] \leq \varepsilon_\alpha$ **do**
11: Remove all $L[x,j]$ from U_j with $x \leq h'$ and remove h from Q_j
12: Let h and h' be the first and second element in Q_j
13: $(\alpha, \varepsilon_\alpha) \leftarrow U_j$.minimumQuery$(\widetilde{L}^*[i,j], \widetilde{B}^*[h,j])$
14: $\widetilde{L}^*[i+1,j] \leftarrow \varepsilon_\alpha$
15: **return** $\max\{\delta(P(n), Q(n)), \min\{\widetilde{L}^*[n-1,n-1], \widetilde{B}^*[n-1,n-1]\}\}$

row ($\widetilde{B}^*[h,j]$) and the value of the previous boundary $\widetilde{L}^*[i,j]$. (The latter is needed only for $h < i$, i.e. if $|Q_j| \geq 2$. For simplicity, we omit this detail in the overview.) If $\varepsilon_\alpha \geq \widetilde{B}^*[h',j]$, there is an ε_α witness for $L[i+1,j]$ through $B[h',j]$, so we can repeat the process with h' (after updating U_j). If h' does not exist (i.e., $|Q_j| = 1$) or $\varepsilon_\alpha < \widetilde{B}^*[h',j]$, we stop and declare ε_α to be optimal. We prove that this process is correct and maintains the invariant. Since the invariant is clearly satisfied at the beginning, correctness then follows by induction.

Lemma 2.8. *Algorithm 1 computes $\widetilde{L}^*[i+1,j]$ and maintains the invariant.*

Proof. By the invariant, a rightmost witness for $\widetilde{L}^*[i,j]$ passes through $B[h_0,j]$, where h_0 is initial head of Q_j. Let h^* be the column index such that a rightmost witness for $\widetilde{L}^*[i+1,j]$ passes through $B[h^*,j]$. Then h^* must be contained in Q_j initially, because by Lemma 2.6, we have $h_0 \leq h^* \leq i$, and by Corollary 2.5, there can be no column index a with $h^* < a \leq i$ that dominates $(h^*, B[h^*,j])$. (Note that if $h^* = i$, it is added at the beginning of the iteration.)

Now let h be the current head of Q_j. By Lemma 2.7, the minimum query on U_j gives the smallest ε_α for which there exists an ε_α-witness for $L[i+1,j]$ that passes through $B[h,j]$. If the current h is less than h^*, then $\varepsilon_\alpha \geq \widetilde{L}^*[i+1,j]$ (definition of $\widetilde{L}^*$); $\widetilde{L}^*[i+1,j] \geq \widetilde{B}^*[h^*,j]$ (there is a witness through $B[h^*,j]$); and $\widetilde{B}^*[h^*,j] \geq \widetilde{B}^*[h',j]$ (the dominance relation ensures that the $\widetilde{B}^*$-values for the indices in Q_j are increasing). Thus, the while-loop in line 10 proceeds to the next iteration. If the current h equals h^*, then by Corollary 2.5, we have $\widetilde{B}^*[a,j] > \widetilde{B}^*[h^*,j]$ for all $h^* < a \leq i$, and the while-loop terminates with

the correct value for $\widetilde{L}^*[i,j]$. It is straightforward to check that Algorithm 1 maintains the data structures Q_j and U_j according to the invariant. □

Theorem 2.9. *Algorithm 1 computes $d_F(P,Q)$ for convex distance function δ in $\mathbb{R}^d$ in $O(n^2 \cdot f(n,d,\delta))$ time, where $f(n,d,\delta)$ represents the time to insert into, delete from, and query the upper envelope data structure.*

Proof. The correctness of the algorithm follows from Lemma 2.8. For the running time, we observe that we insert values only once into Q_j and U_j. Hence, we can remove elements at most once, leading to an amortized running time of $O(1 + f(n,d,\delta))$ for a single iteration of the loop. Since there are $O(n^2)$ cells, the total running time is $O(n^2 \cdot f(n,d,\delta))$ assuming that $f(n,d,\delta)$ is $\Omega(1)$. □

In the generic algorithm, we must take care that U_j uses the (full) unimodal function only for $L[i+1,j]$ and the truncated versions for the other boundaries. As it turns out, we can use the full unimodal distance functions if these behave as pseudolines (i.e., they intersect at most once). Since we compare only functions in the same row (or column), functions of different rows or columns may still intersect more than once. For this approach to work, we must remove from U_j any function that is no longer relevant for our computation. This implies that U_j no longer contains all functions $L[k,j]$ with $h < k \leq i+1$ but a subset of these. We prove the following (see [7] for a full proof).

Lemma 2.10. *Assume that distance functions $L[x,j]$ in row j intersect pairwise at most once. Let h denote a candidate bottom boundary. Let $(\alpha, \varepsilon_\alpha)$ denote the minimum on the upper envelope of the full unimodal distance functions in U_j. Then one of the following holds:*

(i) *$(\alpha, \varepsilon_\alpha)$ is the minimum of the upper envelope of $L[i+1,j]$ and the truncated $\overline{L}[k,j]$ for $h < k \leq i$.*
(ii) *$(\alpha, \varepsilon_\alpha)$ lies on two functions $L[a,j]$ and $L[b,j]$, one of which can be removed from U_j.*
(iii) *$\varepsilon_\alpha \leq \widetilde{L}^*[i,j]$.*

Proof (sketch). $(\alpha, \varepsilon_\alpha)$ either lies on the minimum of a function or on the intersection of two, one increasing and one decreasing. In the first case, it is easy to see that case (i) holds. In the second case, it depends on whether the increasing function is from an earlier or later column than the decreasing one. If the increasing one comes first, then we can argue that the truncated function is never part of the witness envelope for $L[i+1,j]$ or later boundaries. Hence, case (ii) is applicable. If the decreasing one comes first, then we argue that case three must hold: the given intersection is a lower bound for the minimum of the acrophobia function on the second boundary and therefore a lower bound on $\widetilde{L}^*[i,j]$. □

From this lemma, we learn how to modify a minimum-point query. We run the query on the full unimodal functions, ignoring the given constants. If case (ii) holds, that is, the minimum lies on an increasing $L[a,j]$ and a decreasing $L[b,j]$ with $a < b$, we remove $L[a,j]$ from U_j and repeat the query. In both of the other cases, the minimum is either the computed minimum or one of the constants $\widetilde{L}^*[i,j]$ and $\widetilde{B}^*[h,j]$. We take the maximum of these three values.

3 Euclidean distance

In this section we apply our framework to the Euclidean distance measure δ_{E}. Obviously, δ_{E} is convex (and symmetric), so our framework applies. However, instead of computing with the Euclidean distance, we use the squared Euclidean distance $\delta_{\mathrm{E}}^2 = \delta_{\mathrm{E}}(x,y)^2$. Squaring does not change any relative order of height on the distance terrain T, so computing the Fréchet distance with the squared Euclidean distance is equivalent to the Euclidean distance: if $\varepsilon = d_{\mathrm{F}}(P,Q)$ for δ_{E}^2, then $\sqrt{\varepsilon} = d_{\mathrm{F}}(P,Q)$ for δ_{E}. We now show that for δ_{E}^2, the distance functions in a row or column behave like pseudolines. We argue only for the vertical boundaries; horizontal boundaries are analogous.

Lemma 3.1. *For $\delta = \delta_E^2$, each distance terrain function $L[i,j]$ is part of a parabola, and any two functions $L[i,j]$ and $L[i',j]$ intersect at most once.*

Proof. Function $L[i,j]$ represents part of the distance between point $p = P(i)$ and line segment $\ell = Q(j)Q(j+1)$. Assume ℓ' is the line though ℓ, uniformly parameterized by $\lambda \in \mathbb{R}$, i.e. $\ell(\lambda) = (1-\lambda)Q(j) + \lambda Q(j+1)$. Let λ_p denote the λ such that $\ell'(\lambda)$ is closest to p. We see that $L[i,j](\lambda) = |p - \ell'(\lambda_p)|^2 + |\ell'(\lambda) - \ell'(\lambda_p)|^2$. Since ℓ' is uniformly parameterized according to ℓ, we get that the last term is $|\ell|^2(\lambda - \lambda_p)^2$. Hence, the function is equal to $|\ell|^2\lambda^2 - 2|\ell|^2\lambda_p\lambda + |\ell|^2\lambda_p^2 + |p - \ell'(\lambda_p)|^2$, which is a parabolic function in λ. The quadratic factor depends only on ℓ. For two functions in the same row, this line segment is the same, and thus the parabolas intersect at most once. □

By Lemma 2.10, we know that data structure U_j can use the full parabolas. The parabolas of a single row share the same quadratic term, so we can treat them as lines by subtracting $|\ell|\lambda^2$. Now we can use for U_j a standard data structure for dynamic half-plane intersections, or its dual problem: dynamic convex hulls. The fastest dynamic convex hull structure is given by Brodal and Jacob [4]. However, it does not support a query to find a minimal point for the upper envelope; it is unclear whether the structure can support such a query. Instead, we use the slightly slower structure by Overmars and Van Leeuwen [14] with $O(\log^2 n)$ time insertions and deletions. For each insertion, we also have to compute the corresponding parabola, in $O(d)$ additional time. It remains to show how to perform the minimum-point query. The data structure by Overmars and Van Leeuwen maintains a concatenable queue for the upper envelope. We assume this to be implemented via a red-black tree that maintains predecessor and successor pointers. We perform a binary for the minimum point using the intersection pattern of a node, its predecessor, and its successor (see [7] for details). To include the constants, we take the maximum of the minimum point and the constants. Hence, a single query takes $O(\log n)$ time. We obtain the following result.

Theorem 3.2. *Algorithm 1 computes the Fréchet distance under the Euclidean distance in $\mathbb{R}^2$ in $O(n^2(d + \log^2 n))$ time.*

This is slightly slower than known results for the Euclidean metric. However, we think that our framework has potential for a faster algorithm (see Section 5).

4 Polyhedral distance

Here we consider the Fréchet distance with a (convex) polyhedral distance function δ_{P}, i.e., the "unit sphere" of δ_{P} is a convex polytope in $\mathbb{R}^d$. For instance, the L_1 and the L_∞ distance are polyhedral with the cross-polytope and the hypercube as respective unit spheres. Throughout we assume that δ_{P} has *complexity* k, i.e., its unit sphere has k facets. The distance terrain functions $L[i,j]$ and $B[i,j]$ are now piecewise linear with at most k parts; in each row and column the corresponding parts are parallel. Depending on the polytope, the actual maximum number k' of parts may be less. The distance δ_{P} has to be neither regular nor symmetric, but as before, we simplify the presentation by assuming symmetry.

We present three approaches. First, we use an upper envelope structure on piecewise linear functions. Second, we use a brute-force approach which is more efficient for small d and k. Third, we combine these methods.

Upper envelope data structure. For piecewise linear $L[i,j]$ and $B[i,j]$, we can relax the requirements for the upper envelope data structure U_j. There are no parabolas involved, so we only need a data structure that dynamically maintains the upper envelope of lines under insertions, deletions, and minimum queries. Every function contains at most k' parts, so we insert at most nk' lines into the upper envelope. Maintaining and querying the upper envelope per row or column takes $O(nk' \log(nk'))$ time [4]. Thus, the total running time is $O(n^2k' \log(nk') + n^2 g_\delta(d))$, where $g_\delta(d)$ is the time to find the parts of the function.

Brute-force approach. We implement U_j naively. For each segment of P, Q, we sort the facets of δ_{P} by the corresponding slope on the witness envelope, in $O(nk(d + \log k))$ total time. For each facet $l = 1, \ldots k$, we store a doubly linked list F_l of lines representing the linear parts of the unimodal functions in U_j corresponding to facet l, sorted from top to bottom (the lines are parallel). When processing a cell boundary $L[i,j]$, we update each list F_l: remove all lines below the line for $P(i)$ from the back, and append the line for $P(i)$. This takes amortized $O(d)$ time per facet, $O(kd)$ time per cell boundary. We then go through the top lines in the F_l in sorted order to determine (the minimum of) the upper envelope. This takes $O(k)$ time. The total time is $O(n^2kd + nk(d + \log k))$.

A hybrid approach. As in the brute force approach, we maintain a list F_l for each of the k slopes. For each segment in P, Q we initialize these lists, which takes $O(nkd)$ time. But instead of sorting the slopes initially, we maintain the upper hull of the top lines in each F_l. Thus, we only need a dynamic upper hull for k lines. At each cell boundary, we only update k' lines, so we need $O(k' \log k)$ time per cell boundary, $O(n^2k' \log k)$ in total. Therefore, the total time is $O(n^2k' \log k + nkd)$, an improvement for $k' \ll k$.

Combining the previous three paragraphs, yields the following result. The method that works best depends on the relation between n, k, k', and d.

Theorem 4.1. *Algorithm 1 computes the Fréchet distance with a convex polyhedral distance function δ of complexity k in $\mathbb{R}^d$ in $O(\min\{n^2k' \log(nk') + n^2 g_\delta(d), n^2kd + nk(d + \log k), n^2k' \log k + nkd\})$ time, where $g_\delta(d)$ is the time to find the parts of a distance function.*

Let us consider the implications for L_1 and L_∞. Let ℓ be the line segment and p the point defining $L[i,j]$. For L_1 at the breakpoints between the linear parts of $L[i,j]$ one of the coordinates of $\ell - p$ is zero: there are at most $k' = d+1$ parts. The facet of the cross-polytope is determined by the signs of the coordinates. For each linear part we compute the slope in $O(d)$ time, thus $g(d) = O(d^2)$. Hence, the hybrid approach outperforms the brute-force approach.

Corollary 4.2. *Algorithm 1 computes the Fréchet distance with the L_1 distance in $\mathbb{R}^d$ in $O(\min\{n^2 d \log(nd) + n^2 d^2, n^2 d^2 + nd2^d\})$ time.*

For the L_∞ distance the facet is determined by the maximum coordinate. We have $k' \leq k = 2d$. However a facet depends on only one dimension. Hence, for the brute-force method computing the slopes does not take $O(kd)$ time, but $O(k)$. Thus the brute-force method outperforms the other methods for L_∞.

Corollary 4.3. *Algorithm 1 computes the Fréchet distance with the L_∞ distance in $\mathbb{R}^d$ in $O(n^2 d + nd \log d)$ time.*

Approximating the Euclidean distance. We can use a polyhedral distance function to approximate the Euclidean distance. A line segment and a point span exactly one single plane in $\mathbb{R}^d$ (unless they are collinear, in which case we pick an arbitrary one). On this plane, the Euclidean unit sphere is a circle. We approximate this circle with a k-regular polygon in $\mathbb{R}^2$ that has one side parallel to the line segment. Simple geometry shows that for $k = O(\epsilon^{-1/2})$, we get a $(1+\epsilon)$-approximation. The computation is two-dimensional, but we must find the appropriate transformations, which takes $O(d)$ time per boundary. We no longer need to sort the facets of the polytope for each edge; the order is given by the k-regular polygon. This saves a logarithmic factor for the initialization. Again, the brute-force method is best, and Theorem 4.1 gives the following.

Corollary 4.4. *Algorithm 1 computes a $(1+\epsilon)$-approximation of the Fréchet distance with the Euclidean distance in $\mathbb{R}^d$ in $O(n^2(d + \epsilon^{-1/2}))$ time.*

Alternatively we can use Corollary 4.3 to obtain a $\sqrt{d}$-approximation for the Euclidean distance. If we are willing to invoke an algorithm for the decision version, we can go to a $(1+\epsilon)$-approximation by binary search.

Corollary 4.5. *We can calculate a $(1+\epsilon)$-approximation of the Fréchet distance with the Euclidean distance in $O(n^2 d + nd \log d + T(n) \log \frac{\sqrt{d}-1}{\epsilon})$ time, where $T(n)$ is the time needed to solve the decision problem for the Fréchet distance.*

5 Open problems

Faster Euclidean distance. Our framework computes the Fréchet distance for polyhedral distance functions in quadratic time. For the Euclidean distance we do not achieve this running time, but we conjecture that our result can be improved to an $O(n^2)$ algorithm. Currently we use the full power of dynamic upper envelopes, which seems unnecessary as all information is available upfront.

We can for instance determine the order in which the parabolas occur on the upper envelopes, in $O(n^2)$ time for all boundaries. From the proof of Lemma 3.1, we know that the order is given by the projection of the vertices onto the line. We compute the arrangement of the lines dual to the vertices of a curve in $O(n^2)$ time. We then determine the order of the projected points by traversing the zone of a vertical line. This takes $O(n)$ for one row or column. Unfortunately, this alone is insufficient to obtain the quadratic time bound.

Locally correct Fréchet matchings. A matching between two curves that is a Fréchet matching for any two matched subcurves is called a locally correct Fréchet matching [9]. It enforces a relatively "tight" matching. The algorithm in [9] uses a linear overhead on the algorithm of Alt and Godau [1] resulting in an $O(n^3 \log n)$ execution time. We conjecture that our framework is able to avoid this overhead. However, the information we currently propagate is insufficient: a large distance early on may "obscure" the rest of the computations.

References

[1] H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *IJCGA*, 5(1–2):78–99, 1995.

[2] H. Alt, C. Knauer, and C. Wenk. Matching Polygonal Curves with Respect to the Fréchet Distance. In *Proc. 18th STACS*, LNCS 2010, pages 63–74, 2001.

[3] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On map-matching vehicle tracking data. In *Proc. 31st Int. Conf. VLDBs*, pages 853–864, 2005.

[4] G. Brodal and R. Jacob. Dynamic planar convex hull. In *Proc. 43rd FOCS*, pages 617–626, 2002.

[5] K. Buchin, M. Buchin, and J. Gudmundsson. Constrained free space diagrams: a tool for trajectory analysis. *IJGIS*, 24(7):1101–1125, 2010.

[6] K. Buchin, M. Buchin, J. Gudmundsson, M. Löffler, and J. Luo. Detecting commuting patterns by clustering subtrajectories. *IJCGA*, 21(3):253–282, 2011.

[7] K. Buchin, M. Buchin, R. van Leusden, W. Meulemans, and W. Mulzer. Computing the Fréchet Distance with a Retractable Leash. *CoRR*, abs/1306.5527, 2013.

[8] K. Buchin, M. Buchin, W. Meulemans, and W. Mulzer. Four soviets walk the dog - with an application to Alt's conjecture. *CoRR*, abs/1209.4403, 2012.

[9] K. Buchin, M. Buchin, W. Meulemans, and B. Speckmann. Locally correct Fréchet matchings. In *Proc. 20th ESA*, LNCS 7501, pages 229–240, 2012.

[10] A. F. Cook and C. Wenk. Geodesic Fréchet distance inside a simple polygon. *ACM Trans. on Algo.*, 7(1):Art. 9, 9, 2010.

[11] M. de Berg and M. J. van Kreveld. Trekking in the Alps Without Freezing or Getting Tired. *Algorithmica*, 18(3):306–323, 1997.

[12] A. Driemel, S. Har-Peled, and C. Wenk. Approximating the Fréchet distance for realistic curves in near linear time. In *Proc. 26th SoCG*, pages 365–374, 2010.

[13] S. Har-Peled and B. Raichel. The Fréchet distance revisited and extended. In *Proc. 27th SoCG*, pages 448–457, 2011.

[14] M. Overmars and J. van Leeuwen. Maintenance of configurations in the plane. *J. Comput. System Sci.*, 23(2):166–204, 1981.

[15] C. Wenk, R. Salas, and D. Pfoser. Addressing the need for map-matching speed: Localizing global curve-matching algorithms. In *Proc. 18th Int. Conf. on Sci. and Stat. Database Management*, pages 379–388, 2006.