Folding Free-Space Diagrams: Computing the Fréchet Distance between 1-Dimensional Curves*

Kevin Buchin[1], Jinhee Chun[2], Maarten Löffler[3], Aleksandar Markovic[4], Wouter Meulemans[5], Yoshio Okamoto[6], and Taichi Shiitada[7]

1 **Eindhoven University of Technology, Eindhoven, The Netherlands**
k.a.buchin@tue.nl
2 **Tohoku University, Sendai, Japan**
jinhee@dais.is.tohoku.ac.jp
3 **Utrecht University, Utrecht, The Netherlands**
m.loffler@uu.nl
4 **Eindhoven University of Technology, Eindhoven, The Netherlands**
a.markovic@tue.nl
5 **Eindhoven University of Technology, Eindhoven, The Netherlands**
w.meulemans@tue.nl
6 **University of Electro-Communications, Chofu, Japan**
okamotoy@uec.ac.jp
7 **University of Electro-Communications, Chofu, Japan**
shiitada@gmail.com

Abstract

By folding the free-space diagram for efficient preprocessing, we show that the Fréchet distance between 1D curves can be computed in $O(nk \log n)$ time, assuming one curve has ply k.

1998 ACM Subject Classification F.2.2 [Nonnumerical Algorithms and Problems] Geometrical Problems and Computations

Keywords and phrases Fréchet distance, ply, k-packed curves

Digital Object Identifier 10.4230/LIPIcs.SoCG.2017.64

Category Multimedia Contribution

1 Introduction

The Fréchet distance is a popular similarity metric in computational geometry. Computing this distance between curves is mostly well understood: slightly super-quadratic time algorithms for 2D (or higher) are known [1, 5], with a nearly matching conditional lower bound: a $O(n^{2-\delta})$-time algorithm with $\delta > 0$ would imply that the strong exponential time hypothesis (SETH) fails [2]. For the discrete variant in 1D, this same lower bound is also known [4]. However, in the continuous case in 1D, no non-trivial lower bound is known; the fastest known algorithm runs in quadratic time [6]. A near-linear time algorithm is known [3] when the curves are separated: essentially a greedy strategy works. Thus we ask: can we compute the Fréchet distance in 1D in subquadratic time?

* K.B. is supported by NWO (612.001.207); W.M. by NLeSC (027.015.G02) and NWO (639.023.208); and Y.O. by Kayamori Foundation of Informational Science Advancement, JST CREST (JPMJCR1402), and JSPS KAKENHI (JP24106005, JP15K00009).

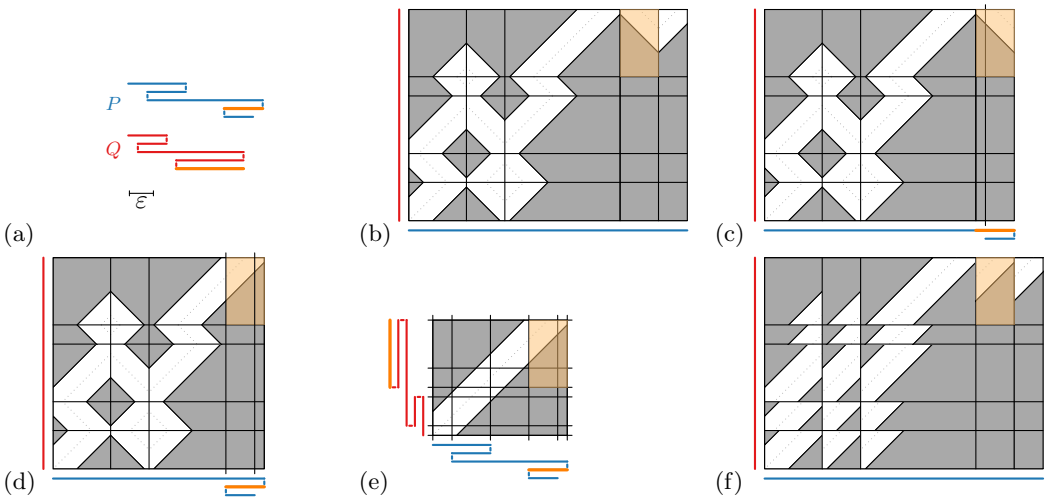

Figure 1 (a) Two 1D curves, lifted to 2D for legibility. (b) The corresponding free-space diagram $\mathcal{F}$. (c–d) Folding up the last two columns. (e) The result of folding all rows and columns. (f) View of each cell in $\mathcal{F}$ from the folded diagram. One orange cell is high-lighted to track through the process.

Contributions. While we leave the question open in general, we answer positively for the case that the number k of times one of the curves revisits the same point on $\mathbb{R}$ is bounded; no restrictions are posed on the other curve. This gives us a running time of $O(nk \log n)$. The crucial ingredient is a structural insight in the free-space diagram for 1D curves: we can fold it to find a simpler representation and solve the decision version in $O(n(k + \log n))$ time.

In 1D, a k-packed curve has a ply of $\Theta(k)$, and vice versa. Near-linear approximation algorithms for k-packed curves exist [3, 7], with matching conditional lower bounds in higher dimensions [3]. Our results show that efficient exact algorithms exist for 1D k-packed curves.

Preliminaries. A 1D curve P is described by $n + 1$ vertices $\langle p_0, \ldots, p_n \rangle$, with $p_i \in \mathbb{R}$ and n edges $p_{i-1}p_i$. We assume that the direction of the curve changes at every vertex. $|P|$ denotes the geometric length of P; we use $|p_i| = \sum_{j=1}^{i} |p_{j-1} - p_j|$ to denote the length up to vertex p_i. Note that $|p_0| = 0$ and $|p_n| = |P|$. We also consider P as a continuous function $P\colon [0, |P|] \to \mathbb{R}$ with $P(|p_i|) = p_i$ and linearly interpolated between vertices. The *ply* of a curve P is $\max_{r \in \mathbb{R}} |P^{-1}(r)| = \max_{r \in \mathbb{R}} |\{t \mid t \in [0, |P|] \wedge P(t) = r\}|$.

For a definition of the Fréchet distance d_{F}, see [1]. The free-space diagram $\mathcal{F}$ represents the parameter space of two curves [1]. It is a $[0, |P|] \times [0, |Q|]$ diagram and a point $(s, t) \in \mathcal{F}$ represents a pair of curve points: $P(s)$ and $Q(t)$. $(s, t) \in \mathcal{F}$ is *free* if $|P(s) - Q(t)| \leq \varepsilon$; the union of free points is the *free space* of $\mathcal{F}$ (Fig. 1(b)). If a monotone path from $(0, 0)$ to (s, t) exists in the free space, $(s, t) \in \mathcal{F}$ is *reachable*. Then, $d_{\mathrm{F}}(P, Q) \leq \varepsilon$ holds if and only if $(|P|, |Q|)$ is reachable in $\mathcal{F}$ [1]. Each edge $p_{i-1}p_i$ of P is a column of width $|p_{i-1} - p_i|$ in the free-space diagram; edges of Q define rows. A combination of two edges defines a cell.

2 Folding free-space diagrams

The central idea for our results is that we can *fold* the free-space diagram, see Fig. 1. This works for the following reason. First, pick a vertex p_i of P and a point q on Q. Now, consider points p on $p_{i-1}p_i$ and p' on p_ip_{i+1} that are equidistant to p_i. Due to minimality, $p = p'$, and thus $|p - q| \leq \varepsilon$ if and only if $|p' - q| \leq \varepsilon$. Considering the vertical line in $\mathcal{F}$ represents p_i, the free space before and after this line are thus a reflection of each other.

Folding up all columns and rows (Fig. 1(c–d)), we find one basic "cell" with lines specifying certain views that the edges of the curves give on this space (Fig. 1(e)). We can compose all these views to recover $\mathcal{F}$ (Fig. 1(f)): assuming both curves start in a rightward direction, we need to only reflect cells in even rows vertically and in even columns horizontally.

With a physical sheet of paper, the size of an unfolded free-space diagram, we only need to fold each row and column, and cut along the two remaining boundaries of the free space. If we unfold it, we have the free space of $\mathcal{F}$. Computing this quadratically sized structure can thus be done in a linear number of operations, using the "parallelism" of the cut.

3 One curve with bounded ply

Here, we sketch a proof for the decision algorithm, which uses folding, in the theorem below.

▶ **Theorem 1.** *Let P and Q be two 1D curves of complexity n; let the ply of Q be k. There is an algorithm to decide in $O(n(k+\log n))$ time whether $d_{\mathrm{F}}(P,Q) \leq \varepsilon$ and an algorithm to compute $d_{\mathrm{F}}(P,Q)$ in $O(nk \log n)$ time.*

We fold all columns (but not the rows) to obtain the *free-space structure* between the infinite line and Q. Consider the points q on Q for which $\|q - r\| \leq \varepsilon$ for some $r \in \mathbb{R}$: the free space on the vertical line at r in the free-space structure. We focus on the *maximal intervals* of free space along such a line. Each interval is bounded by two edges $q_{i-1}q_i$ and $q_{j-1}q_j$ such that $i < j$, $|q_{i-1} - r| > \varepsilon$ and $|q_j - r| > \varepsilon$. Our algorithm consists of the following three steps.

1. Decompose the free-space structure into vertical slabs. Each slab has the same structure of free space, that is, defined by the same edges of Q. (Lemma 3)
2. Precompute reachability information for the free-space structure. (Lemma 4)
3. Walk through the free-space diagram on a column-by-column basis, keeping track of intervals on Q that are reachable at vertex p_i. (Lemma 6)

▶ **Lemma 2.** *There are at most $k+1$ maximal intervals in Q for a given $r \in \mathbb{R}$.*

▶ **Lemma 3** (Decomposition structure)**.** *We can compute in $O(nk)$ time and space, a data structure $\mathcal{S}$ such that: (1) each slab stores the $O(k)$ edges of Q that bound the maximal intervals in order; (2) we can find the slab that contains a point $r \in \mathbb{R}$ in $O(\log n)$ time.*

Proof Sketch. We sort events (vertices of Q, $\pm\varepsilon$) in $O(nk)$ time using a linked list, starting at the previous event. Then, a sweepline decomposes the structure into slabs; only the maximal interval(s) at the event change. Query (2) is a binary search on the sorted events. ◀

▶ **Lemma 4** (Reachability structure)**.** *In $O(nk)$ time, we can determine for each edge $q_{i-1}q_i$ in Q the highest reachable point $\mathcal{H}(q_{i-1}q_i)$ in rightward direction in the free-space structure, starting from any point $(r,q) \in \mathbb{R} \times [0,|Q|]$ with $|r-q| \leq \varepsilon$, $|r-q_i| > \varepsilon$ and $q \in q_{i-1}q_i$.*

Proof Sketch. We do this by walking down the free-space structure, computing or using $\mathcal{H}$ for the boundary that we hit when we shoot a vertical ray up from (r,q). Using $\mathcal{S}$ (Lemma 3), we can answer such queries in constant time: we obtain a running time of $O(nk)$. ◀

We need the reachability structure $\mathcal{H}$ (Lemma 4) also in leftward direction, for leftward edges of P. For simplicity, we describe the algorithm only for rightward edges. For each vertex p_i of P, we compute the (maximal) reachable intervals: intervals reachable from $(0,0)$ in $\mathcal{F}$. As a maximal interval contains at most one reachable interval, a vertex has $O(k)$ reachable intervals (Lemma 2). Lemma 5 captures what is reachable at p_{i+1} from a reachable interval at p_i. We process $\mathcal{F}$ column by column, as formalized by Lemma 6 below. Afterwards, $d_{\mathrm{F}}(P,Q) \leq \varepsilon$ if and only if q_n is in a reachable interval of p_n.

▶ **Lemma 5.** *Let (r, q) be a point in $\mathcal{S}$; $r'' \geq r$; $q_{i-1}q_i$ the edge corresponding to the boundary of the free-space structure that we hit when shooting a ray up from (r, q); and (r', q') the highest reachable point in the free-space structure from (r, q) with $r' \leq r''$. Either $(r', q') = \mathcal{H}(q_{i-1}q_i)$ or the boundary slope at (r', q') is positive and $r' = r''$.*

Proof Sketch. We distinguish three cases for (r', q'): (1) If (r', q') is not on the boundary of the free space, this immediately contradicts the definition of (r', q'); (2) if (r', q') is on an upward-sloped boundary, either $r'' = r'$ or we can find a higher reachable point; (3) if (r', q') is on a downward-sloped boundary and monotonicity prevents us from going higher to the left, (r', q') must actually be equal to $\mathcal{H}(q_{i-1}q_i)$. ◀

▶ **Lemma 6.** *Given the decomposition structure $\mathcal{S}$, the reachability structure $\mathcal{H}$ and the sorted reachable intervals at p_i, we can compute the sorted reachable intervals at p_{i+1} in $O(\log n + k)$ time.*

Proof Sketch. We use the decomposition structure $\mathcal{S}$ to find the slab of p_{i+1} in $O(\log n)$ time (Lemma 3). Let A denote the maximal intervals at p_{i+1}, derived from the slab. Let E be the reachable intervals at p_i. We use $\mathcal{H}(E[j])$ as the highest reachable point in the free-space structure, stored in $\mathcal{H}$ (Lemma 4) with the edge defining the upper end of $E[j]$.

We walk over E and A simultaneously, with indices j and x respectively, to find all reachable intervals for p_{i+1}. We repeatedly do one of the following, checking them in order. (1) If the lower bound of $E[j]$ is above the upper bound of $A[x]$, we increment x by one. (2) If $\mathcal{H}(E[j])$ is below the lower bound of $A[x]$, we increment j by one. (3) If $|q - p_{j-1}| \leq \varepsilon$ where q is the point on Q represented by the lower end of $E[j]$, then we can cut across straight from $E[j]$ at q into interval $A[x]$ and we record a reachable interval starting at q and ending at $A[x]$. (4) Finally, if none of the previous cases apply, we can reach the lower end of $A[x]$ and $A[x]$ is completely reachable. At the end of (3) and (4), we increment as follows: if the upper bound of $A[x]$ is upward sloped, we increment j until we find a value such that $E[j]$ is above $A[x]$ (or run out of values in E). Then, we increment x by one.

Each case takes $O(1)$ time, thus this runs in $O(|A| + |E|) = O(k)$ time (Lemma 2). Including the query in $\mathcal{S}$, this results in a total time of $O(\log n + k)$. Correctness follows from the invariant that we collected all reachable intervals before $A[x]$, that intervals before $E[j]$ cannot reach up to $A[x]$ or later intervals, and that intervals before $A[x]$ do not limit how high $E[j]$ can reach (via Lemma 5). ◀

Acknowledgments. This research was initiated at the Dutch-Japanese Bilateral Workshop on Kinetic Geometric Networks, supported under the Japan-Netherlands Research Cooperative Program by NWO (grant 040.05.033) and JSPS.

References

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

2 K. Bringmann. Why walking the dog takes time: Fréchet distance has no strongly subquadratic algorithms unless SETH fails. In *Proc. 55th FOCS*, pages 661–670, 2014.

3 K. Bringmann and M. Künnemann. Improved approximation for Fréchet distance on c-packed curves matching conditional lower bounds. In *Proc. 26th ISAAC*, pages 517–528, 2015.

4 K. Bringmann and W. Mulzer. Approximability of the discrete fréchet distance. *JoCG*, 7(2):46–76, 2016.

5 K. Buchin, M. Buchin, W. Meulemans, and W. Mulzer. Four Soviets walk the dog: improved bounds for computing the Fréchet distance. *DCG*, 2017.

6 K. Buchin, M. Buchin, R. van Leusden, W. Meulemans, and W. Mulzer. Computing the Fréchet distance with a retractable leash. *DCG*, 56(2):315–336, 2016.

7 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.