Faster Fréchet Distance Approximation Through Truncated Smoothing

Thijs van der Horst
Department of Information and Computing Sciences, Utrecht University, The Netherlands
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Tim Ophelders
Department of Information and Computing Sciences, Utrecht University, The Netherlands
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Abstract

The Fréchet distance is a popular distance measure for curves. Computing the Fréchet distance between two polygonal curves of n vertices takes roughly quadratic time, and conditional lower bounds suggest that even approximating to within a factor 3 cannot be done in strongly-subquadratic time, even in one dimension. The current best approximation algorithms present trade-offs between approximation quality and running time. Recently, van der Horst *et al.* (SODA, 2023) presented an $O((n^2/\alpha)\log^3 n)$ time α-approximate algorithm for curves in arbitrary dimensions, for any $\alpha \in [1, n]$. Our main contribution is an approximation algorithm for curves in one dimension, with a significantly faster running time of $O(n\log^3 n + (n^2/\alpha^3)\log^2 n \log\log n)$. Additionally, we give an algorithm for curves in arbitrary dimensions that improves upon the state-of-the-art running time by a logarithmic factor, to $O((n^2/\alpha)\log^2 n)$. Both of our algorithms rely on a linear-time simplification procedure that in one dimension reduces the complexity of the reachable free space to $O(n^2/\alpha)$ without making sacrifices in the asymptotic approximation factor.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Frécht distance, approximation algorithms, simplification

Digital Object Identifier 10.4230/LIPIcs.SoCG.2024.63

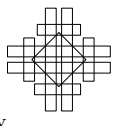
Related Version *Full Version*: `https://arxiv.org/abs/2401.14815`

Funding *Tim Ophelders*: Partially supported by the Dutch Research Council (NWO) under the project number VI.Veni.212.260.

1 Introduction

Comparing curves is an important task in for example trajectory analysis [11], handwriting recognition [19] and matching time series in data bases [18]. To compare curves, one needs a suitable distance measure. The Hausdorff distance is a commonly used distance measure when comparing sets of points. However, although each curve corresponds to a set of points, a point set by itself does not capture the order in which points appear along the curve. This may lead to curves having low Hausdorff distance, even when they are clearly very different. The Fréchet distance is a distance measure that does take the ordering of points along the curves into account, and hence compares curves more accurately.

The first algorithm for computing the Fréchet distance between polygonal curves was given by Godau [16], who presented an $O(n^3 \log n)$ time algorithm for two curves with n vertices in total. Alt and Godau [2] later improved the result to an $O(n^2 \log n)$ time algorithm. The discrete version of the problem was first studied by Eiter and Mannila [15], who gave an $O(n^2)$ time algorithm. While small polylogarithmic improvements have since been achieved (see for example [1, 7]), there is strong evidence that these results cannot be improved significantly, since Bringmann [4] showed that a strongly-subquadratic (i.e., $n^{2-\Omega(1)}$) time algorithm would refute the *Strong Exponential Time Hypothesis* (SETH).




Due to the conditional lower bound, we focus on efficient approximation algorithms. When the curves are from certain families of "realistic" curves, strongly-subquadratic time $(1+\varepsilon)$-approximation algorithms are known to exist. For example, if the curves are either κ-*bounded* or *backbone* curves, the algorithm by Aronov et al. [3] gives a $(1+\varepsilon)$-approximation to the discrete Fréchet distance in $O(n^{4/3}\log n/\varepsilon^2)$ time. Later on, Driemel et al. [13] presented $(1+\varepsilon)$-approximate algorithms for the continuous Fréchet distance that take near-linear time, given that the curves are from one of four realistic curve classes. These four classes include κ-bounded curves, but also *c-packed*, *φ-low density* and *κ-straight* curves. Their result on c-packed curves was improved by Bringmann and Künnemann [5], whose algorithm matches conditional lower bounds.

When approximating the Fréchet distance between arbitrary curves however, SETH again gives conditional lower bounds. The lower bound by Bringmann [4] holds not only for exact algorithms, but for 1.001-approximate algorithms as well. Buchin et al. [8] later improved this bound, showing that under SETH, no strongly-subquadratic $(3-\varepsilon)$-approximation algorithm exists, even for curves in one dimension. For the current strongly-subquadratic algorithms, the best known approximation factor is polynomial (n^{ε}) for both the discrete and the continuous Fréchet distance [6, 10, 21]. Whether a strongly-subquadratic constant factor approximation algorithm exists remains open.

In the discrete setting, Bringmann and Mulzer [6] presented the first strongly-subquadratic time algorithm with polynomial approximation factor. For any $\alpha \in [1, n/\log n]$, their algorithm gives an α-approximation in $O(n^2/\alpha)$ time. This result was later improved by Chan and Rahmati [10], who gave an $O(n^2/\alpha^2)$ time algorithm, for any $\alpha \in [1, \sqrt{n/\log n}]$.

For continuous Fréchet distance, the first polynomial approximation algorithm running in strongly-subquadratic time is due to Colombe and Fox [12]. They gave an α-approximate algorithm running in $O((n^3/\alpha^2)\log n)$ time, for $\alpha \in [\sqrt{n}, n]$. Recently, van der Horst et al. [21] presented the first algorithm that supports arbitrarily small polynomial approximation factors in strongly-subquadratic time. Their achieved running time is $O((n^2/\alpha)\log^3 n)$, for $\alpha \in [1, n]$.

Results. The basis of our results is a curve simplification algorithm. We use the resulting simplified versions of two curves to efficiently approximate the Fréchet distance between the input curves. Doing so, we improve the running time of the algorithm by van der Horst et al. [21] significantly for curves in one dimension, and by a logarithmic factor in higher dimensions. Our running time in one dimension is $O(n\log^3 n + (n^2/\alpha^3)\log^2 n \log\log n)$, in contrast to our $O((n^2/\alpha)\log^2 n)$ time algorithm for higher dimensions. We summarize our algorithms in Section 2, but first we define the Fréchet distance and some useful notation.

Preliminaries

A d-dimensional (polygonal) *curve* is a piecewise-linear function $P\colon [0,1] \to \mathbb{R}^d$, connecting a sequence $p_1, \ldots, p_n$ of d-dimensional points, which we refer to as *vertices*. The linear interpolation between p_i and p_{i+1}, whose image is equal to the directed line segment $\overline{p_i p_{i+1}}$, is called an *edge*. A vertex p_i that lies on the segment $\overline{p_{i-1}p_{i+1}}$ is *degenerate*. We denote by $P[x_1, x_2]$ the subcurve of P over the domain $[x_1, x_2]$. We write $|P|$ to denote the number of vertices of P.

Fréchet distance. A *reparameterization* of $[0,1]$ is a non-decreasing, continuous surjection $f\colon [0,1] \to [0,1]$. Two reparameterizations f, g describe a *matching* (f, g) between two curves P and Q, where any point $P(f(t))$ is matched to $Q(g(t))$. A matching (f, g) between P and Q is said to have *cost*

$$\max_t \|P(f(t)) - Q(g(t))\|.$$

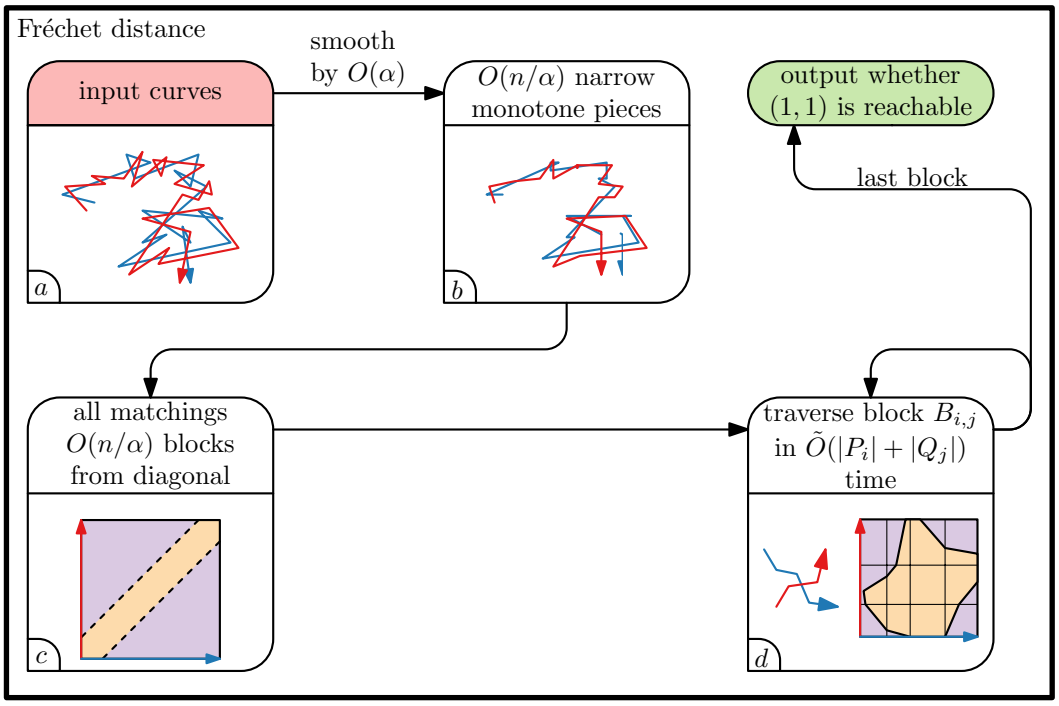


Figure 1 An illustration of the approximate decision algorithm.

It is common to use the Euclidean norm $\|P(f(t))-Q(g(t))\|_2$ to measure the cost of a matching. For our purposes however, it is more convenient to use the L_∞ norm $\|P(f(t))-Q(g(t))\|_\infty$. Since we aim for at least polynomial approximation factors, and the norms differ by at most a factor $\sqrt{d}$, approximations using the L_∞ norm implies the same asymptotic approximation factor for the Euclidean norm, as long as d is constant. A matching with cost at most δ is called a *δ-matching*. The (continuous) *Fréchet distance* $d_F(P,Q)$ between P and Q is the minimum cost over all matchings.

Free space diagram and matchings. The *free space diagram* of P and Q is the parameter space $[0,1]^2$ of $P \times Q$, denoted $\mathcal{D}(P,Q)$. Any point $(x,y) \in \mathcal{D}(P,Q)$ corresponds to the pair of points $P(x)$ and $Q(y)$ on the two curves. Any pair of edges $(P[x_1,x_2],Q[y_1,y_2])$ corresponds to a *cell* $[x_1,x_2]\times[y_1,y_2]$ of $\mathcal{D}(P,Q)$.

For $\delta \geq 0$, a point $(x,y) \in \mathcal{D}(P,Q)$ is *δ-close* if $\|P(x)-Q(y)\|_\infty \leq \delta$. The *$\delta$-free space* $\mathcal{F}_{\leq\delta}(P,Q)$ of P and Q is the subset of $\mathcal{D}(P,Q)$ containing all δ-close points. A point $z_2=(x_2,y_2) \in \mathcal{F}_{\leq\delta}(P,Q)$ is *δ-reachable* from a point $z_1=(x_1,y_1)$ if there exists a bimonotone path in $\mathcal{F}_{\leq\delta}(P,Q)$ from z_1 to z_2. Points that are δ-reachable from $(0,0)$ are simply called δ-reachable points. Alt and Godau [2] observe that the Fréchet distance between $P[x_1,x_2]$ and $Q[y_1,y_2]$ is at most δ if and only if there is a bimonotone path in $\mathcal{F}_{\leq\delta}(P,Q)$ from z_1 to z_2. We therefore abuse terminology slightly and refer to a bimonotone path from z_1 to z_2 as a δ-matching between $P[x_1,x_2]$ and $Q[y_1,y_2]$.

2 Algorithmic outline

Let P and Q be our two d-dimensional input curves with a total of n vertices. Given a parameter $\alpha \geq 1$, we describe an *α-approximate decision algorithm* for the continuous Fréchet distance. Such an algorithm takes as input an additional parameter $\delta \geq 0$, and

must correctly report that $d_F(P,Q) \leq \alpha\delta$ or that $d_F(P,Q) > \delta$. If $d_F(P,Q) \in (\delta, \alpha\delta]$, the algorithm may report either. We thus either confirm that an $\alpha\delta$-matching exists, or assert that no δ-matching exists. Refer to Figure 1 for a diagram illustrating our algorithm. We turn our decision algorithms into approximation algorithms for the Fréchet distance with the procedure of Colombe and Fox [12] (with logarithmic overhead in the running time and arbitrarily small increase in approximation ratio).

Recall that a δ-matching between P and Q represents a bimonotone path from $(0,0)$ to $(1,1)$ in the δ-free space $\mathcal{F}_{\leq\delta}(P,Q)$. Our algorithms search for such a path. However, exploring all of the free space, which may have $\Theta(n^2)$ complexity, does not result in a subquadratic time algorithm. Still, the worst-case complexity of the *reachable* free space, the part of free space containing all δ-reachable points, is smaller for certain types of curves. We explore this in Section 3, where we investigate the relation between the complexity of the reachable free space and the number of *narrow pieces* on the curves. If the number of such pieces is k, then the reachable free space complexity is only $O(kn)$ *blocks*. Here, a block is a generalization of cells, that instead of edges considers monotone pieces. Specifically, a block is the rectangular region of the free space diagram corresponding to two monotone pieces, one of P and one of Q.

Given that a sublinear number of narrow pieces implies a subquadratic complexity of the reachable free space, we present a simplification procedure in Section 4 that reduces the number of narrow pieces to at most dn/α, at an additive 2α cost in the approximation ratio, see Figure 1 (a–b). The simplification takes linear time, and results in a reachable free space complexity of only $O(n^2/\alpha)$ blocks, see Figure 1 (c). Intuitively, the proportion of the free space diagram that we need to explore is inversely proportional to the approximation factor.

The complexity of the free space inside a block $B_{i,j}$ corresponding to monotone pieces P_i, Q_j is $O(|P_i| \cdot |Q_j|)$. This is too large even when considering only the $O(n^2/\alpha)$ blocks containing the reachable free space, as these blocks may still have a combined complexity of $\Theta(n^2)$. However, the free space inside a block is ortho-convex,[1] see Figure 1 (d). We use this fact in Section 6 to construct a data structure on P and Q for traversing $B_{i,j}$ in just $O((|P_i| + |Q_j|)\log n)$ time, after $O(n)$ time preprocessing. This gives an $O(n + (n^2/\alpha)\log n)$ time algorithm for traversing all $O(n^2/\alpha)$ blocks, and hence gives an $O(n + (n^2/\alpha)\log n)$ time $(2\alpha+1)$-approximate decision algorithm.

We use the technique of Colombe and Fox [12] to turn the decision algorithm into an approximation algorithm for the Fréchet distance, while increasing the running time by only a logarithmic factor.

In Section 5 we give a faster algorithm for when P and Q are one-dimensional curves. Figure 2 illustrates this algorithm. The core of the algorithm is a subroutine for constructing *approximate exit sets* (see the bottom diagram of Figure 2). Given a set of points $S \subseteq \{0\} \times [0,1]$ on the left side of the free space diagram, an (α,δ)-exit set for S is a set of points $E_\alpha(S) \subseteq \{1\} \times [0,1]$ on the right side of the diagram that contains all points that are δ-reachable from S, and only points that are $\alpha\delta$-reachable from S. If $(1,1) \in E_\alpha(\{(0,0)\})$, then $d_F(P,Q) \leq \alpha\delta$, and otherwise $d_F(P,Q) > \delta$. Computing such exit sets thus generalizes the approximate decision problem.

We construct approximate exit sets in the full paper [20]. For this we use the ideas of Chan and Rahmati [10] for the current state-of-the-art discrete approximate decision algorithm. They construct a graph approximately representing the free space, which can be used to construct approximate exit sets (in the discrete setting). These exit sets take only $O(n\log n + n^2/\alpha^2)$ time to construct.

[1] A region S is ortho-convex if every line parallel to a coordinate axis intersects S in at most one connected component.

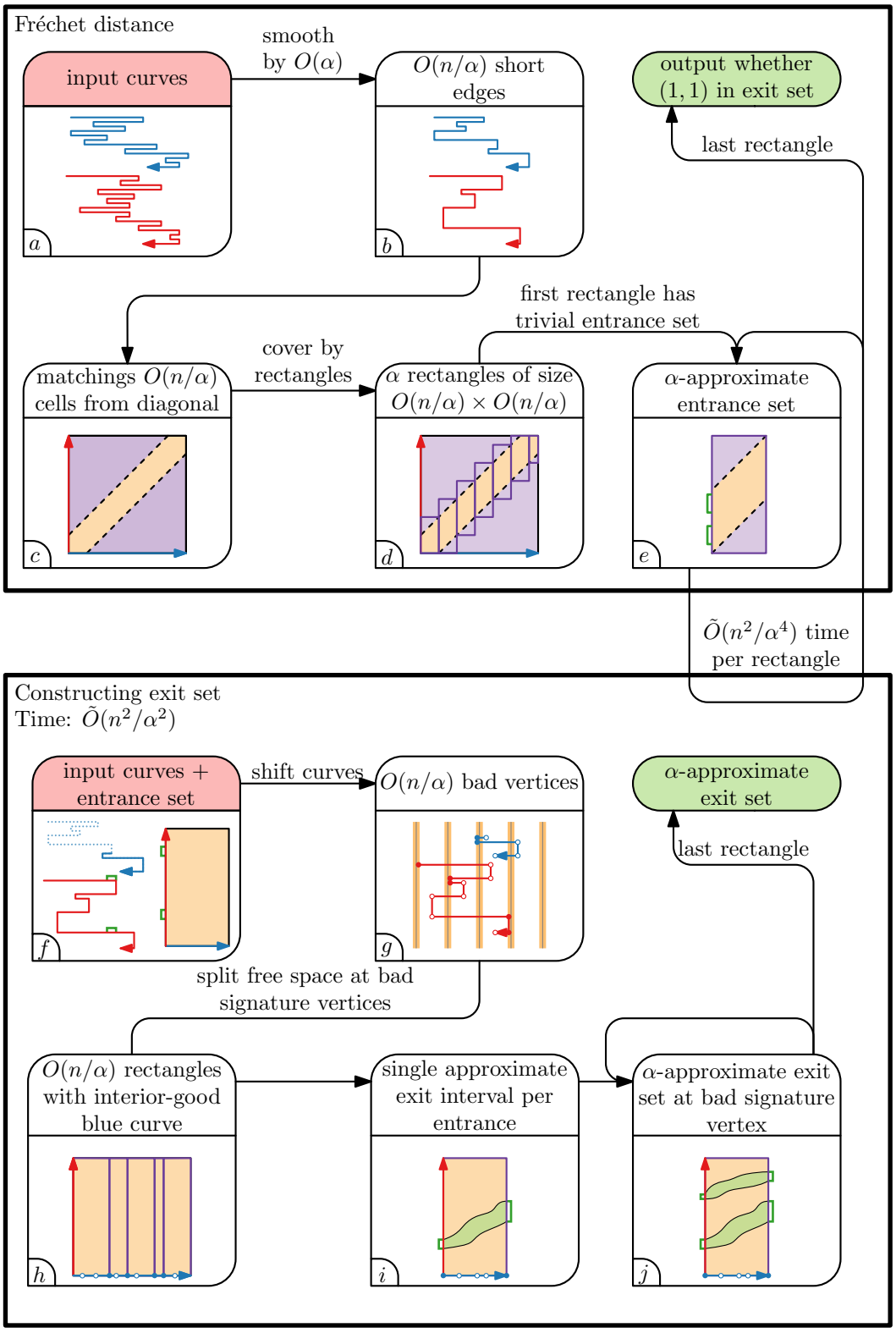


Figure 2 An illustration of the approximate decision algorithm for one-dimensional curves.

To achieve a similar running time in the continuous setting, we first note that continuous matchings in one dimension are relatively discrete. In particular, *signature vertices*, special vertices introduced by Driemel et al. [14], must match in an almost discrete manner, matching to points close to vertices of the other curve. With this in mind, we apply the techniques of Chan and Rahmati [10] to the signature vertices of P.

We construct an infinite grid $\mathbb{G}$ with few *bad* vertices of both P and Q. See Figure 2 (f–g). This grid has cellwidth $\alpha\delta$, and we classify a point as bad if it is within distance 7δ of the boundary of $\mathbb{G}$. Chan and Rahmati [10] show that by shifting P and Q, the number of bad vertices can be made as low as $O(n/\alpha)$. We say that a signature vertex of P is bad if it is within distance 6δ of the boundary of $\mathbb{G}$, rather than within distance 7δ. These vertices must match to points close to bad vertices of Q, and hence have essentially only $O(n/\alpha)$ possible ways to match to points.

Between two bad signature vertices of P, the signature vertices are all sufficiently far from the boundary of $\mathbb{G}$ that we can represent them by the gridcells containing them, after which matchings become effectively diagonal. We can detect such matchings with the exact string matching data structure by Chan and Rahmati [10], and use an additional data structure to handle the matchings around bad signature vertices. For a single entrance, we can then efficiently compute an (α, δ)-exit sets for any subcurve between two subsequent bad signature vertices, see Figure 2 (h–i). The data structure constructs such a set in only $O(\log n \log\log n)$ time. Applied to all $O(n^2/\alpha^2)$ possible matchings with a bad signature vertex of P, we get an $O((n^2/\alpha^2)\log n \log\log n)$ time algorithm for constructing (α, δ)-exit sets of general sets of points, after $O(n \log^2 n)$ time preprocessing.

The above algorithm is already an improvement over the higher-dimensional case, but we can improve the algorithm by taking advantage of the lower-complexity reachable free space. Given that the reachable free space stays within $O(n/\alpha)$ cells of the diagonal, we cover this region by α rectangles of size $O(n/\alpha) \times O(n/\alpha)$ cells. See Figure 2 (c–d). In each rectangle we construct an (α, δ)-exit set for a given set of entrance points, which depend on the exit set of the rectangle to the left of the current one. These exit sets take only $O((n/\alpha)\log^2 n + (n^2/\alpha^4)\log n \log\log n)$ time to construct for a rectangle, totalling $O(n \log^2 n + (n^2/\alpha^3)\log n \log\log n)$ time. This is a factor α improvement, which we would expect given the lower complexity of the reachable free space.

The technique by Colombe and Fox [12] turns our algorithm for constructing a (α, δ)-exit set into an $O(n \log^3 n + (n^2/\alpha^3)\log^2 n \log\log n)$ time α-approximate algorithm for the Fréchet distance.

3 Bounding the reachable free space

The complexity of the δ-free space can be as high as $\Theta(n^2)$, meaning that explicitly traversing the free space does not give a strongly-subquadratic time algorithm. As an improvement, we aim to bound the complexity of the *reachable δ-free space*, the subset of δ-free space that is reachable by a bimonotone path from $(0, 0)$. This subset contains all bimonotone paths to $(1, 1)$, so it suffices to consider only this subset.

Like the complexity of free space, the complexity of the reachable free space can be quadratic. Still, there are special cases of curves for which we can check if the top-right point $(1, 1)$ is reachable in as little as linear time. One example is when the edges of one of the curves are long, meaning their lengths are all strictly greater than 4δ (see the work of Gudmundsson et al. [17]). Under the L_∞ norm, which is the chosen norm in this work, van der Horst et al. [21] generalize the result to the case where one curve is the concatenation of long monotone curves.

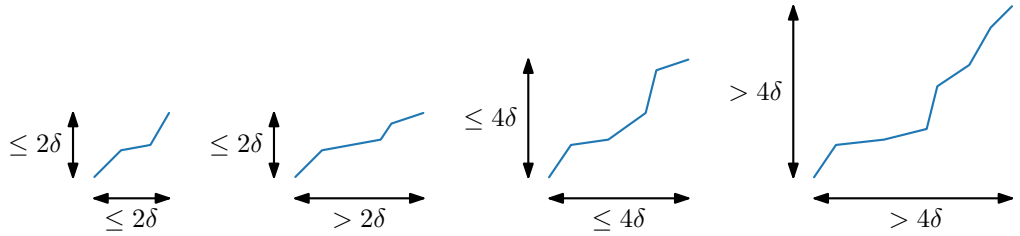


Figure 3 From left to right: two 2δ-narrow curves, a short curve (which is 4δ-narrow) and a curve that is not 4δ-narrow (which is long).

A curve is monotone if in every coordinate it is either non-increasing or non-decreasing. A curve P is the concatenation of maximal monotone pieces $P_1, \ldots, P_k$. We call the curves $P_1, \ldots, P_k$ the *monotone decomposition* of P, and call the individual monotone curves P_i *(monotone) pieces* of P. A monotone curve is long if the L_∞ distance between its endpoints is strictly greater than 4δ, and is short otherwise. In the next section, we investigate a class of monotone curves that we call η-*narrow* curves. A monotone curve is η-narrow if its bounding box has at least one side of length at most η. As an example, short monotone curves are 4δ-narrow, although 4δ-narrow curves are not necessarily short. See Figure 3 for concrete examples. We show that there is a relation between the number of 2δ-narrow monotone pieces of P and Q and the complexity of the reachable free space.

Under the L_∞ norm, the monotone pieces of a curve behave much like line segments. Most importantly, any ball under the L_∞ norm intersects a piece in at most one connected component. For the free space, this implies that the subset of $\mathcal{F}_{\leq\delta}(P,Q)$ that corresponds to a monotone piece of P and a monotone piece of Q is ortho-convex. This somewhat generalizes the convexity of the free space within a cell (defined by two line segments) to unions of cells that together are defined by two monotone pieces.

We define a *block* $B_{i,j} \coloneqq [x_i, x_{i+1}] \times [y_j, y_{j+1}] \subseteq \mathcal{D}(P,Q)$ to be the subset of $\mathcal{D}(P,Q)$ corresponding to a monotone piece $P_i = P[x_i, x_{i+1}]$ and a monotone piece $Q_j = Q[y_j, y_{j+1}]$. This block is the union of the cells defined by the edges of P_i and Q_j. We associate $\mathcal{D}(P,Q)$ with its partitioning into blocks. The *block diagonal* consists of the blocks $B_{i,i}$ and we say that block $B_{i,j}$ is $|i-j|$ blocks away from the (block) diagonal.

For our algorithm, we analyse the complexity of the reachable free space in terms of blocks, rather than cells. In particular, we bound the number of blocks that reachable points can be away from the diagonal in terms of the number of narrow pieces. In the following theorem we show that the number of blocks a reachable point can be away from the block diagonal depends linearly on the number of 2δ-narrow pieces of P and Q.

▶ **Theorem 1.** *Let P and Q be two d-dimensional curves, each with at most k monotone pieces that are 2δ-narrow. Any δ-reachable point in $\mathcal{F}_{\leq\delta}(P,Q)$ lies within $2k+1$ blocks of the block diagonal.*

Proof. Assume for ease of exposition that P and Q contain no degenerate vertices. Suppose for sake of contradiction that a δ-reachable point z lies more than $2k+1$ blocks right of the diagonal. Let π be a bimonotone path to this point. Because z lies more than $2k+1$ blocks right of the diagonal, there must be more than $2k$ pairs of blocks $B_{i,j}, B_{i+1,j}$ that π traverses from the left side of $B_{i,j}$ to the right side of $B_{i+1,j}$. We argue that the number of such pairs is at most $2k$, which gives a contradiction.

Consider two adjacent blocks $B_{i,j}$ and $B_{i+1,j}$ that π traverses from the left side of $B_{i,j}$ to the right side of $B_{i+1,j}$. We show that either P_i or P_{i+1} is 2δ-narrow. To this end, suppose that both P_i and P_{i+1} are not 2δ-narrow. Because π traverses $B_{i,j}$ from left to right, there is

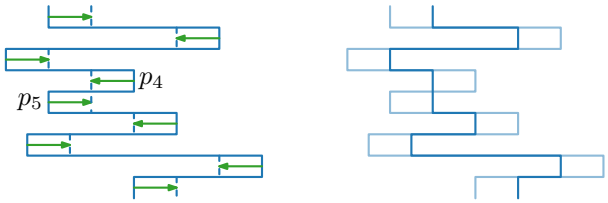


Figure 4 An illustration of truncated smoothings in one dimension. (left) The vertices of curve P (non-dashed) are drawn as vertical segments for clarity. The minimum edge length of P is realized by $\overline{p_i p_{i+1}}$ (right) The result of the smoothing procedure (non-shaded).

a subcurve $Q'_j = Q_j[y, y']$ of Q_j with $\|P_i(0) - Q'_j(0)\|_\infty \leq \delta$ and $\|P_i(1) - Q'_j(1)\|_\infty \leq \delta$. This means that Q'_j is more than a single point, and that it has the same direction with respect to the coordinate axes as P_i. However, P_{i+1} has the opposite direction with respect to at least one coordinate axis, and since it is not 2δ-narrow, its last endpoint $P_{i+1}(1)$ is more than distance δ away from $Q'_j(1)$. Therefore it is more than distance δ away from all of $Q_j[y', 1]$. Hence π cannot traverse $B_{i+1,j}$ from left to right if both P_i and P_{i+1} are not 2δ-narrow.

With the above, we charge each pair of blocks $B_{i,j}, B_{i+1,j}$ that π traverses from the left side of $B_{i,j}$ to the right side of $B_{i+1,j}$ to one of the 2δ-narrow pieces of P corresponding to these blocks. As each narrow piece corresponds to at most two such pairs, pieces are charged at most twice. It follows that there are at most $2k$ such pairs of blocks, which gives a contradiction. This shows that z is at most $2k + 1$ blocks right of the diagonal. That z is at most $2k + 1$ blocks above the diagonal follows from a symmetric argument. ◀

4 Reducing the number of narrow pieces

We present a family of simplifications for curves that we use to reduce the number of narrow pieces on a curve. The simplifications are based on truncated smoothings for Reeb graphs [9], and we hence call them truncated smoothings.

4.1 Truncated smoothings

First consider a one-dimensional curve P. We assume for ease of exposition that P has no degenerate vertices.[2] Let $\varepsilon \geq 0$ be at most half the minimum edge length of P. The *truncated ε-smoothing* P^ε of P is the curve obtained by truncating every edge of P by ε on either side. See Figure 4 for an example. We extend the truncated smoothing definition to all non-negative values $\varepsilon' \geq 0$ by recursively defining the truncated ε'-smoothing of P for $\varepsilon' > \varepsilon$ to be the truncated $(\varepsilon' - \varepsilon)$-smoothing of P^ε.

In the full paper [20], we show that the identity matching between P and its ε-smoothing P^ε has cost at most ε. Intuitively, this means that points on P are moved over distance at most ε during the smoothing process. Furthermore, we show that when smoothing two one-dimensional curves P and Q by the same amount ε, the Fréchet distance is decreased by at most 2ε, and never increases.

▶ **Lemma 2.** *Let P and Q be two one-dimensional curves. For all $\varepsilon \geq 0$ we have $d_F(P^\varepsilon, Q^\varepsilon) \leq d_F(P, Q) \leq d_F(P^\varepsilon, Q^\varepsilon) + 2\varepsilon$.*

[2] Note that if P has degenerate vertices, then the curve obtained by deleting these vertices has Fréchet distance 0 to P.

Truncated smoothings are particularly useful with respect to reducing the complexity of the reachable free space. Recall from Theorem 1 that the number of narrow monotone pieces is proportional to the number of blocks a matching can stray away from the block diagonal. In one dimension, a narrow monotone piece is simply a short edge, and a block is merely a cell of the free space diagram. Thus, a low number of short edges on the curves implies a small complexity bound for the reachable free space. In the following lemma (refer to [20] for the proof), we show that there is a linear trade-off in (additive) approximation quality and number of short edges of the simplified curves.

▶ **Lemma 3.** *Let P and Q be two one-dimensional curves with n vertices. For all $\alpha \in [1, n]$ and $\delta \geq 0$, there is an $\varepsilon \leq \alpha\delta$ for which P^ε and Q^ε together have at most n/α edges of length at most 2δ.*

We extend the truncated smoothing definition to higher-dimensional curves by defining the truncated smoothing of a d-dimensional curve using a parameter vector $\vec{\varepsilon} \in [0, \infty)^d$. The truncated $\vec{\varepsilon}$-smoothing $P^{\vec{\varepsilon}}$ of a curve P is the result of coordinate-wise truncated smoothing with the corresponding elements of $\vec{\varepsilon}$ as parameters. Under the L_∞-norm, this coordinate-wise procedure yields an error that is at most the maximum of the errors obtained from the one-dimensional truncated smoothings. Note that every monotone piece on a higher-dimensional curve P projects onto (part of) a single edge in every dimension. Thus, if the number of edges of a certain length c in any projection is at most k, then there are at most dk monotone pieces on P that are c-narrow. We obtain the following theorem as consequences of Lemmas 2 and 3:

▶ **Theorem 4.** *Let P and Q be two d-dimensional curves with n vertices. For all $\alpha \in [1, n]$ and $\delta \geq 0$, there is a vector $\vec{\varepsilon} \in [0, \alpha\delta]^d$ for which $P^{\vec{\varepsilon}}$ and $Q^{\vec{\varepsilon}}$ together have at most dn/α monotone pieces that are 2δ-narrow. Furthermore, we have $d_F(P^{\vec{\varepsilon}}, Q^{\vec{\varepsilon}}) \leq d_F(P, Q) \leq d_F(P^{\vec{\varepsilon}}, Q^{\vec{\varepsilon}}) + 2\alpha\delta$.*

4.2 Constructing truncated smoothings

We present a linear time algorithm for constructing the truncated ε-smoothing P^ε of a one-dimensional curve P. This immediately gives a linear time algorithm for constructing truncated smoothings for higher-dimensional curves. Note that the parameterization of P^ε is piecewise-linear and its vertices correspond to the vertices of P. Hence it suffices to compute the images of the vertices of P after simplification.

The algorithm relies on computing the *death times* of the vertices $p_1, \ldots, p_n$ of P. We define the death time of a vertex p_i to be the smallest value $\varepsilon \geq 0$ for which p_i is degenerate in P^ε (and thus is removed for higher parameter truncated smoothings). The death times of p_1 and p_n are infinite, and the death time of degenerate vertices of P are 0.

We show in the full paper [20] that the death time of a vertex can be expressed in terms of the left and right minima (or maxima) in its sublevel (or superlevel) set component. The sublevel set of a point p on P is the set of points on P with value at most p. We define the *sublevel set component* of p to be the connected component of its sublevel set that contains p, see Figure 5. The *superlevel set component* of p is defined symmetrically.

For a local maximum p_i of P, let P^- be its sublevel set component. We define the points ℓ_i and r_i as (global) minima on the prefix and suffix curves of P^- that end and start at p_i, respectively. We let $m_i := \min\{|p_i - \ell_i|, |p_i - r_i|\}$, see Figure 5. We symmetrically define P^+ to be the superlevel set component of a local minimum p_i of P, and symmetrically define ℓ_i and r_i in terms of P^+. The definition of m_i is the same as for local maxima. For a degenerate vertex p_i we set $m_i := 0$. In the full version [20] we show the expression for death times given in Lemma 5.

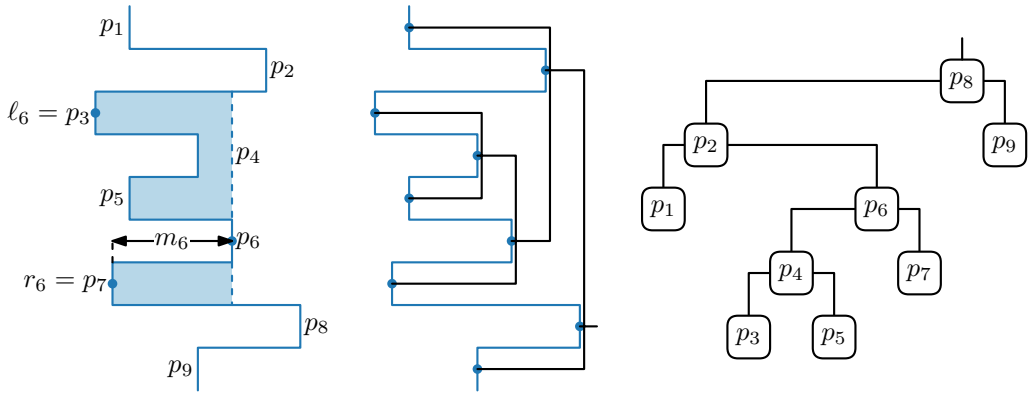


Figure 5 (left) The sublevel set component of p_i, below the dashed line segment. Points ℓ_i and r_i are the minima of the left and right parts of this component. (middle and right) The max-Cartesian tree built on the vertex sequence of P.

▶ **Lemma 5.** *For all $2 \leq i \leq n-1$ the death time of vertex p_i is equal to $m_i/2$.*

With the expression for the death times of interior vertices, we are able to compute the death times of these vertices in linear total time. To this end we use *Cartesian trees*, introduced by Vuillemin [22]. A Cartesian tree is a binary tree with the heap property. We call a Cartesian tree a *max-Cartesian tree* if it has the max-heap property and a *min-Cartesian tree* if it has the min-heap property. A max-Cartesian tree T for a sequence of values $x_1, \ldots, x_n$ is recursively defined as follows. The root of T contains the maximum value x_j in the sequence. The subtree left of the root node is a max-Cartesian tree for the sequence $x_1, \ldots, x_{j-1}$, and the right subtree is a max-Cartesian tree for the sequence $x_{j+1}, \ldots, x_n$ (see Figure 5). Min-Cartesian trees are defined symmetrically.

Note that in a max-Cartesian tree, the left and right subtrees of a local maximum p_i contain precisely the vertices before and after p_i that are in its sublevel set component. Thus, the minimum value stored in its left subtree is ℓ_i and the minimum value stored in its right subtree is r_i. This allows us to compute the death times of all local maxima of P with a bottom-up traversal of the max-Cartesian tree built on the vertex sequence $p_1, \ldots, p_n$. We symmetrically compute the death times of all local minima using the min-Cartesian tree.

▶ **Lemma 6.** *We can compute the death time of every interior vertex in $O(n)$ time.*

We construct the truncated ε-smoothing by using another max-Cartesian tree, this time built on the sequence of death times (see Figure 6). This tree contains all non-degenerate vertices in a single connected component containing the root, which can be computed in $O(n)$ time. This gives the truncated ε-smoothing of a one-dimensional curve. For the truncated $\vec{\varepsilon}$-smoothing of a higher-dimensional curve, we simply perform the simplification procedure coordinate-wise, with a changing parameter depending on $\vec{\varepsilon}$. This gives the following result:

▶ **Theorem 7.** *We can construct the truncated $\vec{\varepsilon}$-smoothing of a d-dimensional curve with n vertices in $O(n)$ time for any vector $\vec{\varepsilon} \in [0, \infty)^d$.*

We additionally use the death times to compute a suitable parameter vector $\vec{\varepsilon}$ for reducing the number of narrow pieces. We search for an interval $(\varepsilon, \varepsilon + \delta]$ that contains at most n/α death times, as each short edge in the truncated ε-smoothing has a vertex that becomes degenerate when upping ε by at most δ, and thus a low number of death times indicates a low number of short edges. For details, refer to the full paper [20].

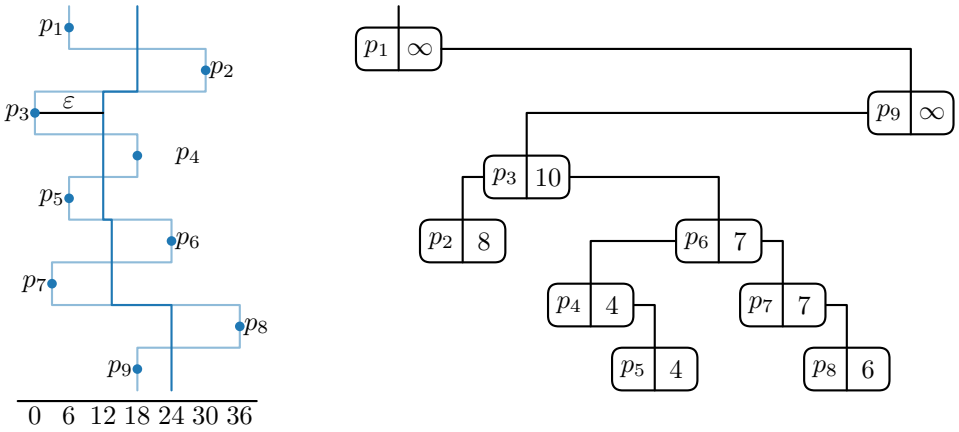


Figure 6 (left) The truncated ε-smoothing for $\varepsilon = 8$. The curve has three non-degenerate vertices, corresponding to the three vertices p_1, p_3 and p_9 with death times larger than ε. The curve also indicates the location of degenerate vertices. (right) The death times of the vertices, stored in a max-Cartesian tree.

▶ **Theorem 8.** *Let P and Q be two d-dimensional curves with n vertices. Let $\alpha \in [1, n]$ and $\delta \geq 0$. In $O(n)$ time, we can compute a vector $\vec{\varepsilon} \in [0, \alpha\delta]^d$ for which $P^{\vec{\varepsilon}}$ and $Q^{\vec{\varepsilon}}$ together have at most dn/α monotone pieces that are 2δ-narrow.*

5 Approximating the Fréchet distance for one-dimensional curves

Next we present an $O(n \log^3 n + (n^2/\alpha^3) \log^2 n \log\log n)$ time algorithm for computing an α-approximation to the Fréchet distance between P and Q, for $\alpha \in [1, n]$. We first present an approximate decision algorithm, which is then turned into an approximation algorithm for computing the Fréchet distance with the black box technique of Colombe and Fox [12].

We use the simplification of Section 4 to reduce the number of 2δ-narrow monotone pieces on both P and Q to at most n/α, which by Theorem 1 makes the reachable free space constrained to within $O(n/\alpha)$ blocks of the diagonal. This gives an additive 2α term in the approximation factor. Note that a monotone piece in one dimension is simply an edge of a curve, and hence a block is merely a cell of the free space diagram. Thus matchings are constrained to within $O(n/\alpha)$ cells of the diagonal.

A crucial ingredient in our algorithm is the concept of *δ-signature vertices*, introduced by Driemel et al. [14]. One interesting property of δ-signature vertices is that any δ-matching matches them close to endpoints of edges of the other curve (see also [21]). This makes matchings behave somewhat discrete near these vertices. We use this property to apply the techniques of Chan and Rahmati [10] for approximate discrete Fréchet distance to our continuous setting. Specifically, we use them to construct exit sets.

The *δ-exit set* for a set of points $S \subseteq \{0\} \times [0, 1]$ with respect to P and Q is the set of all points $E(S) \subseteq \{1\} \times [0, 1]$ that are δ-reachable from points in S. We allow for approximations, where instead of constructing an (exact) δ-exit set for S, we construct an *(α, δ)-exit set* $E_\alpha(S)$. Such a set must contain all of $E(S)$, and may only contain points that are $\alpha\delta$-reachable from points in S. In the full paper [20] we give a construction algorithm for exit sets, proving Theorem 9.

▶ **Theorem 9.** *Let P and Q be two one-dimensional curves with n vertices. For any $\alpha \in [1, n]$ and $\delta \geq 0$, we can construct an (α, δ)-exit set $E_\alpha(S)$ for a given set $S \subseteq \{0\} \times [0, 1]$ consisting of $O(n)$ connected components in $O(n \log^2 n + (n^2/\alpha^2) \log n \log\log n)$ time.*

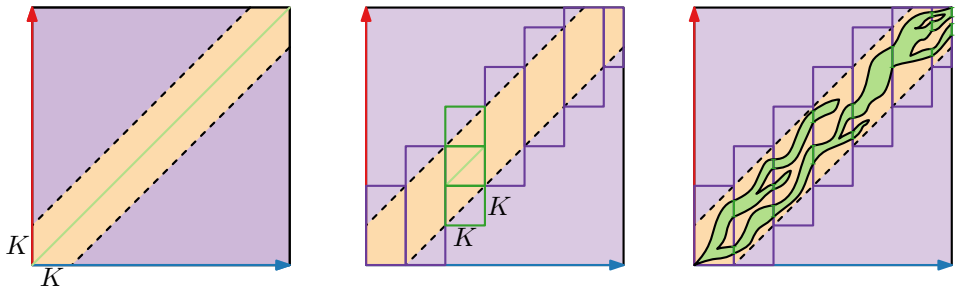


Figure 7 (left) Matchings are constrained to within K cells of the diagonal. (middle) The α rectangles covering all possible matchings. Each rectangle spans K cells in width and $3K$ cells in height. (right) Constructing exit sets in each rectangle, given the exit set of the previous rectangle.

Since matchings are restricted to lie within $K = 2n/\alpha + 1$ cells of the free space diagonal, the total complexity of the reachable free space is only $O(n^2/\alpha)$, rather than potentially quadratic. We wish to translate this lower complexity into lower input complexities for subroutines. For this we cover the reachable free space with α interior-disjoint rectangles, each $K \times 3K$ cells in dimension, such that rectangles do not share a common x-coordinate on their interiors. That is, the rectangles are laid out from left to right over the reachable free space. See Figure 7 for an illustration.

We iteratively go over the rectangles from left to right, constructing exit sets inside each rectangle for given sets on their left boundary. Let $R_1, \ldots, R_\alpha$ be the rectangles in left to right order. Let $P_1, \ldots, P_\alpha$ be the subcurves of P corresponding to these rectangles, and let $Q_1, \ldots, Q_\alpha$ be the subcurves of Q corresponding to these rectangles. For each rectangle R_i we construct an (α, δ)-exit set $E_i = E_\alpha(S_i)$ for the set $S_i = E_{i-1} \cap \mathcal{F}_{\leq\delta}(P, Q)$, with respect to P_i and Q_i.

Initially, $S_1 = \{(0, 0)\}$. Given a set S_i, we construct E_i using Theorem 9 to construct an (α, δ)-exit set $E_\alpha(S_i)$ with respect to P_i and Q. Because P_i and Q_i have only $O(K)$ vertices each, we construct E_i in $O(K \log^2 K + (K^2/\alpha^2) \log K \log\log K)$ time. We then construct the set S_{i+1} as the intersection between E_i and $\mathcal{F}_{\leq\delta}(P_i, Q_i)$, which takes $O(K \log K)$ time by sorting the sets and scanning over them.

Performing the above for all α rectangles, with $K = O(n/\alpha)$, we obtain an $O(n \log^2 n + (n^2/\alpha^3) \log n \log\log n)$ time α-approximate decision algorithm, after using the simplification procedure of Section 4 to limit the complexity of the reachable free space. This incurs an additive error of $2\alpha\delta$, resulting in an overall 3α-approximate decider.

▶ **Theorem 10.** *Let P and Q be two one-dimensional curves with $O(n)$ vertices. Let $\alpha \in [1, n]$ and $\delta \geq 0$. We can decide whether $d_F(P, Q) \leq 3\alpha\delta$ or $d_F(P, Q) > \delta$ in $O(n \log^2 n + (n^2/\alpha^3) \log n \log\log n)$ time.*

To turn this decision algorithm into an approximation algorithm for the Fréchet distance, we apply the black box technique of Colombe and Fox [12]. For any $\varepsilon \in (0, 1]$, this increases the running time by a factor $\log(n/\varepsilon)$ and the approximation factor by a factor $(1+\varepsilon)$. We set $\varepsilon = 1$ for concreteness, giving an $O(n \log^3 n + (n^2/\alpha^3) \log^2 n \log\log n)$ time 6α-approximation algorithm for the Fréchet distance. To turn this algorithm into an α-approximation algorithm running in the same time bound, we set $\alpha \leftarrow \alpha/6$ for $\alpha \geq 6$, and run the exact quadratic time decision algorithm of Alt and Godau [2] for $\alpha \in [1, 6)$. This gives the following result:

▶ **Theorem 11.** *Let P and Q be one-dimensional curves of $O(n)$ vertices. Let $\alpha \in [1, n]$. We can compute an α-approximation of $d_F(P, Q)$ in $O(n \log^3 n + (n^2/\alpha^3) \log^2 n \log\log n)$ time.*

6 An improved algorithm for curves in arbitrary dimensions

Finally, we give a simpler algorithm for approximate continuous Fréchet distance in arbitrary dimensions, which also improves upon the state-of-the-art algorithm by van der Horst et al. [21] by a logarithmic factor. We again first present an approximate decision algorithm, which is then turned into an approximation algorithm for computing the Fréchet distance with the black box technique of Colombe and Fox [12].

We use the simplification of Section 4 to reduce the number of 2δ-narrow monotone pieces on both P and Q to at most dn/α, which by Theorem 1 makes the reachable free space constrained to within $O(n/\alpha)$ blocks of the diagonal. This gives an additive 2α term in the approximation factor. In Corollary 13, we show how to efficiently traverse the free space inside a block, after preprocessing P and Q.

▶ **Lemma 12.** *We can preprocess a d-dimensional curve P with n vertices in $O(n)$ time, such that given a monotone piece P_i of P and line segment e we can compute the δ-close points on the bottom and top sides of $\mathcal{D}(P_i, e)$ in $O(\log n)$ time.*

Proof. We store the vertices of each monotone piece P_i in a binary search tree T_i. Together with T_i we store for each coordinate whether P_i moves in the positive or negative direction, cutting ties arbitrarily. Preprocessing P takes $O(n)$ time, since the vertices are sorted after scanning over P.

Given a monotone piece P_i and query line segment e, we compute the δ-close points on the bottom and top sides of $\mathcal{D}(P_i, e)$ by computing the at most two maximal subcurves of P_i that are within distance δ of the endpoints of e. The subcurve close to the first endpoint of e corresponds to the δ-close points on the bottom side of $\mathcal{D}(P_i, e)$, while the other subcurve corresponds to the δ-close points on the top side.

Let q be an endpoint of e and let R be the axis-parallel hypercube centered at q with diameter δ. Since we know the direction of P_i, we can apply binary search over the vertices of P_i to find the first and last vertices of P_i inside R. With T_i this takes $O(\log |P_i|)$ time. Computing the maximal subcurve inside R takes constant additional time, as we merely have to compare the edges incident to the reported vertices with the boundary of R. Thus after $O(\log |P_i|) = O(\log n)$ time, we have computed the subcurve of P_i that is within distance δ of q, which corresponds to the δ-close points on the bottom and top sides of $\mathcal{D}(P_i, e)$. ◀

By preprocessing P and Q separately as above, we can efficiently traverse the free space within a block $B_{i,j}$ from bottom to top by iteratively considering the edges of Q_j together with all of P_i. We can traverse $B_{i,j}$ from left to right in a symmetric manner. This leads to the following result.

▶ **Corollary 13.** *We can preprocess P and Q in $O(n)$ time, such that given a block $B_{i,j}$ and all δ-reachable points on the bottom and left sides of $B_{i,j}$, represented by two horizontal and vertical line segments, we can compute all δ-reachable points on the top and right sides of $B_{i,j}$ in $O((|P_i| + |Q_j|) \log n)$ time.*

By Theorem 1 each monotone piece of P and Q corresponds to only $O(n/\alpha)$ blocks containing parts of the reachable free space. Using the data structure of Corollary 13 to traverse these blocks, we get a free space traversal algorithm with running time

$$O\Big((n/\alpha) \cdot \Big[\sum_i |P_i| + \sum_j |Q_j|\Big] \log n\Big) = O((n^2/\alpha) \log n).$$

This is summarized in Theorem 14.

▶ **Theorem 14.** *Let P and Q be two d-dimensional curves with n vertices. For any $\alpha \in [1, n]$ and $\delta \geq 0$, we can decide whether $d_F(P, Q) \leq (1 + 2\alpha)\delta$ or $d_F(P, Q) > \delta$ in $O((n^2/\alpha) \log n)$ time.*

To turn this decision algorithm into an approximation algorithm for the Fréchet distance, we apply the black box technique of Colombe and Fox [12]. For any $\varepsilon \in (0, 1]$, this increases the running time by a factor $\log(n/\varepsilon)$ and the approximation factor by a factor $(1 + \varepsilon)$. We set $\varepsilon = 1$ for concreteness, giving an $O((n^2/\alpha) \log^2 n)$ time $(2 + 4\alpha)$-approximation algorithm for the Fréchet distance. To turn this algorithm into an α-approximation algorithm running in the same time bound, we set $\alpha \leftarrow (\alpha - 2)/4$ for $\alpha \geq 6$, and run the exact quadratic time decision algorithm of Alt and Godau [2] for $\alpha \in [1, 6)$. This gives the following result:

▶ **Theorem 15.** *Let P and Q be two d-dimensional curves with n vertices. For any $\alpha \in [1, n]$ we can compute an α-approximation to $d_F(P, Q)$ in $O((n^2/\alpha) \log^2 n)$ time.*

7 Concluding remarks

We presented faster approximation algorithms for computing the continuous Fréchet distance between curves. For the one-dimensional case in particular, our algorithm significantly improves upon previous results. Our curve simplification procedure proves to be a valuable tool in speeding up the one-dimensional algorithm, and we are confident that future approximation algorithms can make use of the simplification as well.

While we used the simplification for curves in general dimensions, lowering the complexity of the reachable free space by a factor α (in terms of blocks), we have not been able to take advantage of this lower complexity with the existing result of van der Horst et al. [21]. Instead, we currently traverse this lower-complexity space somewhat naively. We expect that our one-dimensional algorithm can be adapted to work in higher dimensions as well, taking advantage of the lower complexity to yield a faster algorithm. The main hurdle is the time required to compare two "good" subcurves. In one dimension this takes merely constant time using string matching. In higher dimensions however, where we define a curve as good if the endpoints of all its monotone pieces are good, we currently see no sublinear time algorithm for this, even after preprocessing. The problem here is that a monotone piece may increase the complexity of the label curve beyond linear, and it is unclear how to obtain a linear bound even when approximations are allowed.

References

1. Pankaj K. Agarwal, Rinat Ben Avraham, Haim Kaplan, and Micha Sharir. Computing the discrete Fréchet distance in subquadratic time. *SIAM Journal on Computing*, 43(2):429–449, 2014. doi:10.1137/130920526.
2. Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 5:75–91, 1995. doi:10.1142/S0218195995000064.
3. Boris Aronov, Sariel Har-Peled, Christian Knauer, Yusu Wang, and Carola Wenk. Fréchet distance for curves, revisited. In *proc. 14th Annual European Symposium on Algorithms (ESA)*, pages 52–63, 2006. doi:10.1007/11841036_8.
4. Karl Bringmann. Why walking the dog takes time: Fréchet distance has no strongly subquadratic algorithms unless SETH fails. In *proc. 55th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 661–670, 2014. doi:10.1109/FOCS.2014.76.
5. Karl Bringmann and Marvin Künnemann. Improved approximation for Fréchet distance on c-packed curves matching conditional lower bounds. *International Journal of Computational Geometry & Applications*, 27(1-2):85–120, 2017. doi:10.1142/S0218195917600056.

6 Karl Bringmann and Wolfgang Mulzer. Approximability of the discrete Fréchet distance. *Journal of Computational Geometry*, 7(2):46–76, 2016. doi:10.20382/jocg.v7i2a4.

7 Kevin Buchin, Maike Buchin, Wouter Meulemans, and Wolfgang Mulzer. Four soviets walk the dog: Improved bounds for computing the Fréchet distance. *Discrete & Computational Geometry*, 58(1):180–216, 2017. doi:10.1007/s00454-017-9878-7.

8 Kevin Buchin, Tim Ophelders, and Bettina Speckmann. SETH says: Weak Fréchet distance is faster, but only if it is continuous and in one dimension. In *proc. 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2887–2901, 2019. doi:10.1137/1.9781611975482.179.

9 Erin Wolf Chambers, Elizabeth Munch, and Tim Ophelders. A family of metrics from the truncated smoothing of Reeb graphs. In *proc. 37th International Symposium on Computational Geometry (SoCG)*, pages 22:1–22:17, 2021. doi:10.4230/LIPIcs.SoCG.2021.22.

10 Timothy M. Chan and Zahed Rahmati. An improved approximation algorithm for the discrete Fréchet distance. *Information Processing Letters*, 138:72–74, 2018. doi:10.1016/j.ipl.2018.06.011.

11 Lei Chen, M. Tamer Özsu, and Vincent Oria. Robust and fast similarity search for moving object trajectories. In *proc. 31st ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 491–502, 2005. doi:10.1145/1066157.1066213.

12 Connor Colombe and Kyle Fox. Approximating the (continuous) Fréchet distance. In *proc. 37th International Symposium on Computational Geometry (SoCG)*, pages 26:1–26:14, 2021. doi:10.4230/LIPIcs.SoCG.2021.26.

13 Anne Driemel, Sariel Har-Peled, and Carola Wenk. Approximating the Fréchet distance for realistic curves in near linear time. *Discrete & Computational Geometry*, 48(1):94–127, 2012. doi:10.1007/s00454-012-9402-z.

14 Anne Driemel, Amer Krivosija, and Christian Sohler. Clustering time series under the Fréchet distance. In *proc. 27th Annual Symposium on Discrete Algorithms (SODA)*, pages 766–785, 2016. doi:10.1137/1.9781611974331.ch55.

15 Thomas Eiter and Heikki Mannila. Computing discrete Fréchet distance. Technical report, Vienna University of Technology, 1994.

16 Michael Godau. A natural metric for curves – Computing the distance for polygonal chains and approximation algorithms. In *proc. 8th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 127–136, 1991. doi:10.1007/BFb0020793.

17 Joachim Gudmundsson, Majid Mirzanezhad, Ali Mohades, and Carola Wenk. Fast Fréchet distance between curves with long edges. *International Journal of Computational Geometry & Applications*, 29(2):161–187, 2019. doi:10.1142/S0218195919500043.

18 Xiao-Ying Liu and Chuan-Lun Ren. Fast subsequence matching under time warping in time-series databases. In *proc. 12th International Conference on Machine Learning and Cybernetics (ICML)*, pages 1584–1590, 2013. doi:10.1109/ICMLC.2013.6890855.

19 Mario E. Munich and Pietro Perona. Continuous dynamic time warping for translation-invariant curve alignment with applications to signature verification. In *proc. 7th International Conference on Computer Vision (ICCV)*, pages 108–115, 1999. doi:10.1109/ICCV.1999.791205.

20 Thijs van der Horst and Tim Ophelders. Faster Fréchet distance approximation through truncated smoothing, 2024. arXiv:2401.14815.

21 Thijs van der Horst, Marc J. van Kreveld, Tim Ophelders, and Bettina Speckmann. A sub-quadratic n^{ε}-approximation for the continuous Fréchet distance. In *proc. ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1759–1776, 2023. doi:10.1137/1.9781611977554.ch67.

22 Jean Vuillemin. A unifying look at data structures. *Communications of the ACM*, 23(4):229–239, 1980. doi:10.1145/358841.358852.