

Partial Curve Matching under the Fréchet Distance*

Sariel Har-Peled[†] Yusu Wang[‡]

December 12, 2006

Abstract

Curve matching is a fundamental problem which occurs in many applications. In this paper, we extend the Fréchet distance to measure *partial* curve similarity, study its properties, and develop efficient algorithms to compute it. In particular, under the L_1 or L_∞ metrics, we present an algorithm to compute, in $O(m^2)$ time, the *partial Fréchet distance* between a segment and a polygonal curve of size m . We then develop an approximation algorithm that, in $O((n+m)^3/\varepsilon^2)$ time, computes a $(1-\varepsilon)$ -approximation to the optimal partial Fréchet distance between two polygonal curves of size n and m , respectively. Finally, we propose a third algorithm that, in near quadratic time, computes a (constant) double-sided error approximation to the optimal partial Fréchet distance.

1 Introduction

Measuring similarity between curves is a fundamental problem that occurs in many fields, including computer graphics, pattern recognition, geographic information system, structural biology, and many others. For example, one of the principle challenges in computational biology currently is to classify and cluster the rapidly increasing number of protein structures [BWF⁺00] into similar groups. In this case, proteins are usually modeled as polygonal curves that represent their backbones. As such, curve similarity arises naturally as a measure of resemblance between proteins.

One of the natural measures of similarity between two curves is the *Fréchet distance*. Intuitively, imagine that a dog and its handler are walking on their respective curves. Both can control their speed, but they can only go forward. The Fréchet distance of these two curves is the minimal length of leash necessary for the dog and the handler to move from the starting points of the two curves to their respective endpoints. The Fréchet distance is a *continuous* measure, while most other similarity measures, such as the widely used *root-mean-square-deviation* (RMSD) distance, are *discrete* measures, considering only vertices of

*Work on this paper by Sariel Har-Peled was partially supported by an NSF CAREER award CCR-0132901. The full version of this paper is available online from [HW06].

[†]Dept. of Comp. Sci, University of Illinois; 1304 West Springfield Ave., Urbana, IL 61801; sariel@cs.uiuc.edu.

[‡]Dept. of Comp. Sci. and Engineering, The Ohio State Univ, Columbus, OH 43016; yusu@cse.ohio-state.edu.

input curves. Furthermore, the Fréchet distance takes the inherent order between points along the curves into consideration, making it a better measure of similarity for curves than alternatives such as the Hausdorff distance [AG00, AKW04].

The Fréchet distance and its variants have been widely used in many applications [KP99, KKS05, KHM⁺98, PP90]. Alt and Godau [AG95] presented an algorithm to compute the Fréchet distance between two polygonal curves with n and m vertices, respectively, in $O(nm \log(nm))$ time. Efficient approximation algorithms have been developed for special families of curves [AKW04, AHK⁺06]. However, so far, no algorithm, exact or approximate, with running time $o(nm)$ has been found for this problem for general curves.

Where the Fréchet distance falls short is when two curves share only partial similarity. For example, two protein structures may be similar around important common functional sites, but are dissimilar in other places. To this end, we investigate the *partial curve matching problem*, where given two curves and a threshold δ , we would like to find the best Fréchet matching such that the largest possible fraction of the two curves are matched with distance threshold smaller than δ .

One simpler variant is the partial matching problem under the discrete Fréchet distance, where we consider only the vertices of the curves. The commonly used RMSD distance can also capture partial similarity to some extent. However, it is well known that these discrete measures can fail to capture the “real” distance between input curves (see figure on the right).

The problem of minimizing the Fréchet distance under various classes of transformations has also been studied [AKW01, CM05, Wen02]. However the running times are prohibitively high and practical solutions remain elusive. Fréchet distance has also been extended to graphs [AERW03, BPSW05], to piecewise smooth curves [Rot05], to simple polygons [BBW06], and to surfaces [AB05]. Finally, it has also been used in context of high-dimensional approximate nearest-neighbor search [Ind02], curve simplification [AHMW05], and curve morphing [EHG⁺02].



Our results. As described above, current continuous curve similarity measures do not yet describe partial similarity, while discrete measures may not reflect the real similarity accurately. In this paper, we extend the Fréchet distance to measure *continuous partial* curve similarity in a natural way, and develop several efficient algorithms to compute it. In particular, given a distance threshold δ , let $\mathcal{D}_{\delta}(P, Q)$ denote the optimal partial Fréchet distance between two polygonal curves P and Q , which is the longest subcurves of P and Q that are matched with Fréchet distance at most δ (see Figure 1). The resulting matching also specifies which portions of the two curves are similar and which are dissimilar. This information is probably more useful than just the similarity score computed.

In Section 3, for the case where P is a single edge, we compute, in $O(m^2)$ time, $\mathcal{D}_{\delta}(P, Q)$, under the L_1 or L_∞ norms, where Q is a polygonal curve with m edges. Unfortunately, the computation of $\mathcal{D}_{\delta}(P, Q)$ seems to be challenging in general, and it is unclear whether a polynomial time algorithm exists or not. We thus focus on efficient approximation algorithms for $\mathcal{D}_{\delta}(P, Q)$.

Specifically, given two polygonal curves P and Q of size n and m , respectively, the exact

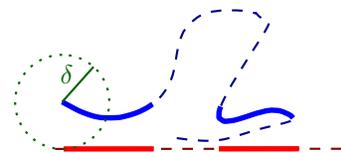


Figure 1: The partial Fréchet distance is the total length of the thick subcurves.

algorithm mentioned above leads to a constant-factor approximation algorithm for $\mathcal{D}_\partial(P, Q)$ under the L_1 or L_∞ norms that runs in $O((n+m)^3)$ time. We next present in Section 4.2 an algorithm that computes a partial matching between P and Q whose score μ is an $(1-\varepsilon)$ -approximation of $\mathcal{D}_\partial(P, Q)$, that is, $(1-\varepsilon)\mathcal{D}_\partial(P, Q) \leq \mu \leq \mathcal{D}_\partial(P, Q)$, in $O((n+m)^3/\varepsilon^2)$ time under the L_1 and L_∞ norms, and in $O((n+m)^4/\varepsilon^2)$ time under any other L_p norm.

Finally, in Section 5, we present a double-sided approximation algorithm for $\mathcal{D}_{\partial, \delta}(P, Q)$, where we also allow the distance threshold δ to be relaxed within a constant factor. The algorithm runs in $O((n+m)^2 \log^2 m)$ time and $O((n+m) \log m)$ space. We remark that even for simpler matching problems such as computing the discrete Fréchet distance or the RMSD distance, currently there is no approximation algorithm that runs in $o(nm)$ time for general curves.

Note, that the $(1-\varepsilon)$ -approximation algorithm of Section 4.2, also implies a constant factor approximation algorithm with similar performance to the one presented in Section 4.1. However, we use the ideas of Section 4.1 in the (faster) double-sided approximation algorithm of Section 5.

Conclusions and open problems are discussed in Section 6.

2 Problem Definition and Preliminaries

A (parameterized) *curve* in \mathbb{R}^d can be represented as a function $f: [0, 1] \rightarrow \mathbb{R}^d$. A (monotone) *reparametrization* α is a continuous non-decreasing function $\alpha: [0, 1] \rightarrow [0, 1]$ with $\alpha(0) = 0$ and $\alpha(1) = 1$. A *matching* between f and g is simply a pair of monotone reparametrizations (α, β) of f and g respectively. Namely, the point $f(\alpha(x))$ is matched to the point $g(\beta(x))$, for any $x \in [0, 1]$. Given two curves $f, g: [0, 1] \rightarrow \mathbb{R}^d$, the *Fréchet distance* between them under the L_p norm, $\mathcal{D}(f, g)$, is defined as

$$\mathcal{D}(f, g) := \inf_{\alpha, \beta} \max_{t \in [0, 1]} d_p(f(\alpha(t)), g(\beta(t))).$$

where $d_p(x, y)$ denotes the distance between points x and y under the L_p norm, and α and β range over all monotone reparametrizations.

Given a distance threshold $\delta > 0$ and a matching (α, β) of curves f and g , the *score* of (α, β) , also referred to as the *partial distance* between f and g for a matching (α, β) , is defined as

$$\mathcal{S}_{\alpha, \beta}(f, g) = \int_{d_p(f(\alpha(t)), g(\beta(t))) \leq \delta} (||f'(\alpha(t))|| + ||g'(\beta(t))||) dt,$$

where $||v||$ is the L_2 norm of a vector v ; namely, the score is the total length of the portions of the two curves f and g that are matched with distance smaller than δ . Naturally, we would like to maximize the length of these portions, and the *partial Fréchet distance* between f and g for a threshold δ is defined as: $\mathcal{D}_{\partial, \delta}(f, g) := \max_{\alpha, \beta} \mathcal{S}_{\alpha, \beta}(f, g)$, that is, the maximum score of any matching of f and g .

For two polygonal curves $P = \langle p_1, \dots, p_n \rangle$ and $Q = \langle q_1, \dots, q_m \rangle$, an alternative way to view the partial Fréchet distance $\mathcal{D}_\delta(P, Q)$ is via the following **free space diagram** $M = M_\delta(P, Q)$ (which is a variant of the one used in [AG95]): M is a n by m map such that its i th column corresponds to the i th edge of P and has width $\|p_i p_{i+1}\|$, while its j th row corresponds to the j th edge of Q and has width $\|q_j q_{j+1}\|$. See figure on the right. For a set of segments X in the plane, let $\text{len}_p(X)$ denote the length of the segments of X under the L_p norm. Let $\mathbf{f} : [0, \text{len}_2(P)] \rightarrow \mathbb{R}^d$ and $\mathbf{g} : [0, \text{len}_2(Q)] \rightarrow \mathbb{R}^d$ represent the arc-length parametrization of P and Q , respectively. Then every point (x, y) in M corresponds to a pair of points $\mathbf{f}(x) \in P$ and $\mathbf{g}(y) \in Q$. We abuse the notation slightly and use $P(x)$ and $Q(y)$ to represent $\mathbf{f}(x)$ and $\mathbf{g}(y)$, respectively, from now on.

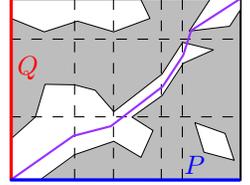


Figure 2: M .

A point $(x, y) \in M$ is **good** if $d_p(P(x), Q(y)) \leq \delta$. By convexity of the L_p norm, the good points within any cell $M[i][j]$ of M form a connected convex region. In particular, under the L_2 norm, this **good region** is the intersection between an ellipse with the cell $M[i][j]$. For curves in the plane, it is the intersection between a parallelogram with the cell $M[i][j]$ under the L_∞ and L_1 norms. For curves in \mathbb{R}^d , the good region is the intersection between the cell $M[i][j]$ with a convex polygon of complexity d under the L_∞ norm, and with a convex polygon of complexity 2^d under the L_1 norm.

There is a one-to-one correspondence between all possible matchings of P and Q , and the set of monotone paths in the diagram M from its bottom-left corner to its top-right corner, where a **monotone path** in M is monotone in both the horizontal and the vertical directions. Given any monotone path π of M , let π_M denote its intersection with the good regions of M . It is easy to verify that $\mathcal{S}_\pi(P, Q)$, the partial distance between P and Q induced by π , also referred to as the **score of π** , is simply the L_1 -norm length of π_M , which is the total length of the projections of the good portions of π onto the x and y axes. To compute $\mathcal{D}_{\delta, \delta}(P, Q)$, the goal is to find an **optimal monotone path** whose score is maximized. This corresponds to an **optimal partial matching** between P and Q .

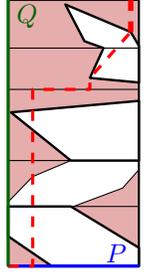
A monotone path π is **conformal** if (i) its intersection with bad regions in M consists of only vertical and horizontal segments, (ii) its intersection with the good region within each cell of M is a single segment, and (iii) the endpoints of horizontal/vertical segments of π lie either on cell boundaries or on boundaries of good regions. The following observation is straightforward.

Observation 2.1 *Any monotone path in M can be modified into a conformal path without decreasing its score.*

Hence, in the remainder of this paper, we consider only conformal paths. Finally, given a distance threshold $\delta \geq 0$, τ is an **α -approximation** of $\mathcal{D}_{\delta, \delta}(P, Q)$ if $\alpha \mathcal{D}_{\delta, \delta}(P, Q) \leq \tau \leq \mathcal{D}_{\delta, \delta}(P, Q)$; and τ is a **double-sided (α, β) -approximation** of $\mathcal{D}_{\delta, \delta}(P, Q)$ if $\alpha \mathcal{D}_{\delta, \delta}(P, Q) \leq \tau \leq \mathcal{D}_{\delta, \beta \delta}(P, Q)$.

3 Partial Fréchet distance of a segment to a curve

In this section, we compute the optimal partial Fréchet distance between a segment s and a polygonal curve Q with m edges under the L_1 or L_∞ norms. In this case, for a fixed threshold δ , the free space diagram $M = M_\delta(s, Q)$ is a single column of size m , and the good region within each cell $M[j]$ is the intersection between $M[j]$ and some convex polygon of constant complexity (depending only on the dimension d of the space). A conformal path π of M is **critical** if every vertical segment passes through at least one vertex of some good region. See figure on the right for an example. The lemma below suggests that we only need to consider critical paths to compute $\mathcal{D}_\delta(s, Q)$.



Lemma 3.1 *Given an edge s and a polygonal curve Q , under the L_1 or L_∞ norms, there always exists an optimal path π^* that is critical.*

Proof: Given any optimal conformal path π of M , we modify it into a critical path as follows. Let $e = (x, y)$ be any vertical edge of π violating the critical condition. Since π is conformal, the endpoints of e , x and y , lie on the boundary of either some cell or some good region (see Figure 3 (a) where x lies on a cell boundary and y lies on the boundary of a good region). Imagine now shifting the edge e either to the left or right, with its endpoints sliding along the corresponding boundaries. Let $\Delta(r)$ be the change in the score of π induced by shifting e horizontally by distance r ; $r < 0$ means shifting e to the left. Since e does not pass through any vertex of the good regions, it intersects a set of boundary edges of the good regions in the interior. Let e' be such an edge with slope ρ . When e shifts horizontally by distance r , the change in the score induced by e' is $\text{sign}(e')(r + \rho r)$, where $\text{sign}(e')$ is either $+1$, -1 or 0 , depending on whether e' is the upper or lower boundary edge of some good region, and whether it is the first/last edge that e intersects. Note that this change in the score is *independent* of r . It follows that $\Delta(r) = -\Delta(-r)$, as long as e intersects the same set of boundary edges. Hence there is always a direction to monotonically shift e to increase (or maintain) the score of the induced partial matching, until it reaches a vertex of a good region, or it merges with another vertical edge (see Figure 3 (b)). If it is the first case, then e is now critical and we are done. For the second case, we continue shifting the new vertical edge (edge wy' in Figure 3 (b)) until it eventually reaches a vertex of a good region. Repeating this for every non-critical vertical edge in π , results in a path π^* which is critical, and $\mathcal{S}_{\pi^*}(s, Q) \geq \mathcal{S}_\pi(s, Q)$.

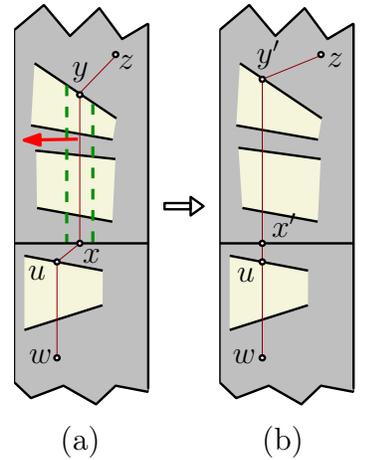


Figure 3: Shifting

Lemma 3.1 suggests a dynamic programming algorithm to compute $\mathcal{D}_\delta(s, Q)$. First, project the vertices of the good regions from all the cells onto each of the horizontal edges of M . There are $N = O(m)$ such **Steiner points** on each horizontal edge. Next, consider a cell $M[j]$. Let $\mathbf{p}_j[i]$ denote the i th Steiner point along the j th horizontal edge of M , and $\mathcal{S}(i, j)$ denote the optimal score for any critical path from $\mathbf{p}_1[1]$ to $\mathbf{p}_j[i]$. It holds that

$$\mathcal{S}(i, j + 1) = \max \left\{ \mathcal{S}(i - 1, j + 1), \max_{1 \leq k \leq i} \{ \mathcal{S}(k, j) + W_{M[j]}(\mathbf{p}_j[k], \mathbf{p}_{j+1}[i]) \} \right\},$$

where $W_C(x, y)$ is the score of the optimal path connecting x to y within a cell C . In other words, the optimal critical path to $\mathbf{p}_{j+1}[i]$ either arrives from the left, or from one of the

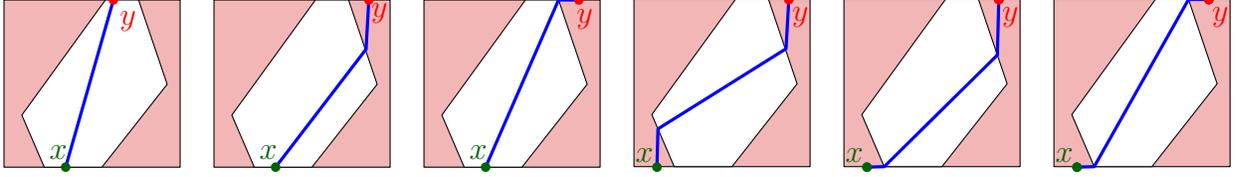


Figure 4: Entry point x and exit point y decide constant number of configurations for optimal critical path within a given cell, some examples are shown.

Steiner points on the lower boundary edge of $M[j]$. We remark that $W_C(x, y)$ is not defined if there is no monotone path to connect x to y .

Computing $\mathcal{S}(i, j)$ s. Given a cell $M[j]$, for simplicity, let U and B denote its upper and lower boundary edges, respectively; $U[i] = \mathbf{p}_{j+1}[i]$ and $B[i] = \mathbf{p}_j[i]$. We now project the vertices of the good region in $M[j]$ onto U and B , and subdivide U and B into constant number of intervals U_1, \dots, U_c and B_1, \dots, B_c , respectively. We refer to such projected vertices as *interval vertices*, to distinguish them from the set of Steiner points on each edge. Note that each interval vertex is also a Steiner vertex. Given an entry point $x \in B$ and an exit point $y \in U$, it can be verified that any optimal path connecting x to y has only constant descriptive complexity within $M[j]$. Some examples are shown in Figure 3. In fact, since we only consider critical paths, we have the following claim, which can be shown by straightforward but tedious case analysis.

Claim 3.2 *Given any pair of entry/exit points (x, y) with $x \in B$ and $y \in U$, the optimal score $W(x, y)$ can be computed in constant time. Furthermore, given any $x, u \in B_i$ and $y, v \in U_j$, for $1 \leq i, j \leq c$, we have that*

$$W(x, y) - W(u, y) = W(x, v) - W(u, v),$$

if all four terms are defined.

Lemma 3.3 *For a fixed j , given $\mathcal{S}(1, j), \dots, \mathcal{S}(N, j)$, one can compute, in $O(N)$ time, the values of $\mathcal{S}(1, j+1), \dots, \mathcal{S}(N, j+1)$, using $O(c)$ space.*

Proof: We compute $\mathcal{S}(i, j+1)$ for increasing value of i as follows. Let $T(k, i)$ be the score by extending the optimal critical path ending at $B[k]$ to $U[i]$: i.e, $T(k, i) = \mathcal{S}(k, j) + W(B[k], U[i])$ for $k \leq i$, and $T(k, i) = 0$ otherwise. Thus $\mathcal{S}(i, j+1) = \max\{\mathcal{S}(i-1, j), \max_{1 \leq k \leq N} T(k, i)\}$. For each i , we maintain c IDs $I_i[1] \in B_1, \dots, I_i[c] \in B_c$, such that, for any $l \in [1, c]$, it holds $I_i[l] = \operatorname{argmax}_{k \in B_l} T(k, i)$. This implies that

$$\mathcal{S}(i, j+1) = \max\{\mathcal{S}(i-1, j+1), \max_{1 \leq l \leq c} T(I_i[l], i)\}.$$

Given $\mathcal{S}(i, j+1)$ and $I_i[l]$ s, if $U[i+1]$ and $U[i]$ lie in the same interval of U , then we can compute all $I_{i+1}[l]$ s (thus $\mathcal{S}(i+1, j+1)$) in constant time. Indeed, by Claim 3.2, the difference $T(k, i+1) - T(k, i)$ remains the same for all $k \in B_l$ and $k \leq i$. Hence either $I_{i+1}[l] = I_i[l]$ or $I_{i+1}[l] = i+1$, which can be checked in constant time for every $1 \leq l \leq c$. If $U[i+1]$ moves to a new interval, then we recompute $I_{i+1}[l]$ s from scratch, which takes $O(N)$ time. Overall,

the time complexity to compute $\mathcal{S}(i, j)$ s, for all $1 \leq i \leq N$ is $O(cN) = O(N)$, and the space complexity is $O(c)$. \blacksquare

Since $N = O(m)$, putting everything together, we conclude that

Theorem 3.4 *Given $\delta > 0$, the partial Fréchet distance between a segment s and a polygonal curve Q of size m under the L_1 or L_∞ norms, $\mathcal{D}_{\partial, \delta}(s, Q)$, can be computed in $O(m^2)$ time and $O(m)$ space.*

4 Approximation Algorithms for Two Curves

In this section, we outline two algorithms to approximate $\mathcal{D}_\partial(P, Q)$ for two polygonal curves P and Q of size n and m , respectively. The first algorithm computes, in $O((n+m)^3)$ time, a $\frac{1}{3}$ -approximation to the optimal partial matching, for the L_1 and L_∞ norms. The second algorithm computes, in $O((n+m)^3/\varepsilon^2)$ time, a $(1-\varepsilon)$ -approximation to the optimal partial matching, for the L_1 and L_∞ norms. This algorithm works for any L_p norm, but its running time deteriorates to $O((n+m)^4/\varepsilon^2)$.

4.1 A Constant-Factor Approximation

Given a monotone path π in the free space diagram $M = M_\delta(P, Q)$, the set of cells of M that π passes through forms a *trail*. See Figure 5. We subdivide the trail into three portions: (1) right-turning cells $R(\pi)$, where the trail makes a right turn; (2) left-turning cells $L(\pi)$, where the trail makes a left turn; and (3) remaining vertical/horizontal segments $HV(\pi)$. In other words, for every vertical / horizontal segment of the trail, remove the cells at both ends, and add the remaining segment (if exists) into $HV(\pi)$. The removed end-cells, i.e, the turning cells (darker cells in Figure 5), will be added into sets $R(\pi)$ and $L(\pi)$ appropriately. Let $\mathcal{S}(X, \pi)$ denote the score (i.e, partial distance) induced by the intersection of a path π and a collection of cells X , and let $\mathcal{S}(\pi) = \mathcal{S}(M, \pi)$. Now consider any optimal conformal path π^* that realizes $\mathcal{D}_\partial(P, Q)$. By the pigeon hole principle, we have that

$$\max\{\mathcal{S}(R(\pi^*), \pi^*), \mathcal{S}(L(\pi^*), \pi^*), \mathcal{S}(HV(\pi^*), \pi^*)\} \geq \frac{1}{3}\mathcal{D}_\partial(P, Q).$$

In what follows, we will compute three monotone paths in M , π_1, π_2 and π_3 , such that $\mathcal{S}(\pi_1) \geq \mathcal{S}(R(\pi^*), \pi^*)$, $\mathcal{S}(\pi_2) \geq \mathcal{S}(L(\pi^*), \pi^*)$, and $\mathcal{S}(\pi_3) \geq \mathcal{S}(HV(\pi^*), \pi^*)$. Clearly, $\max\{\mathcal{S}(\pi_1), \mathcal{S}(\pi_2), \mathcal{S}(\pi_3)\}$ is a $\frac{1}{3}$ -approximation of $\mathcal{D}_\partial(P, Q)$.

4.1.1 Computation of π_1 and π_2

The computation of π_2 is the same as that of π_1 . Hence here we only consider π_1 . The following observation is straightforward.

Observation 4.1 *For any monotone path π in M , no two cells in set $R(\pi)$ shares the same column nor the same row of M .*

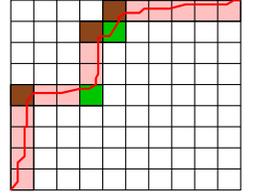


Figure 5: Trail.

For any path π in M , set $S_R(\pi) = \mathcal{S}(R(\pi), \pi)$ and let $S_R(i, j)$ denote the best S_R -score for any monotone path from $M[1][1]$ to $M[i][j]$ (but excluding $M[i][j]$). It is easy to see that:

$$S_R(i, j) = \max\{S_R(i, j - 1), S_R(i - 1, j), S_R(i - 1, j - 1) + W_{M[i-1][j]}^*\},$$

where W_C^* denote the best score for any path within a cell C . The third term corresponds to the case where cell $M[i - 1][j]$ is a right-turning cell. Note that W_C^* is the longest L_1 -norm length of any segment contained within the good region Ω in cell C . Since Ω is convex and has constant complexity, this optimal segment s^* can be computed in constant time. (In fact, it is easy to verify that both endpoints of s^* are vertices of Ω .) It then follows that $S_R(n, m)$, as well as a path π_1 to realize it, can be computed in $O(nm)$ time and space. Thus we have that $\mathcal{S}(\pi_1) \geq S_R(\pi_1) = S_R(n, m) \geq \mathcal{S}(R(\pi^*), \pi^*)$.

4.1.2 Computation of π_3

For any monotone path π in M , set $S_{HV}(\pi) = \mathcal{S}(HV(\pi), \pi)$. Let $S_{HV}(i, j)$ be the best S_{HV} -score for any monotone path from $M[1][1]$ to $M[i][j]$. We have that:

$$S_{HV}(i, j) = \max\left\{\max_{1 \leq k \leq i} \{h(k, i, j) + S_{HV}(k - 1, j - 1)\}, \max_{1 \leq k \leq j} \{v(i, k, j), S_{HV}(i - 1, k - 1)\}\right\},$$

where $h(k, i, j)$ (resp., $v(i, k, j)$) represents the best score of any path contained within the subrow from $M[k][j]$ to $M[i][j]$ (resp., subcolumn from $M[i][k]$ to $M[i][j]$). The first term corresponds to the case where the best path coming from left, with cell $M[k - 1][j]$ being a right-turning cell, while the second term corresponds to the case that it comes from bottom.

We can precompute $h(k, i, j)$ and $v(i, k, j)$ for all i, j, k s. Consider the case for $v(i, k, j)$; that for $h(k, i, j)$ is similar. Observe that each computation of $v(i, k, j)$ is exactly one instance of the special case that we considered in Section 3 — $v(i, k, j)$ is the best partial Fréchet distance between the i th edge of P and the subchain of Q between its k th and j th vertices. Hence it can be computed in $O((j - k)^2)$ time. In fact, for fixed i and k , the computation for $v(i, k, m)$ produces all $v(i, k, j)$ s, for $j \in [k, m]$. This implies that we can compute $S_{HV}(n, m)$, as well as a path π_3 to realize it, in $O((n + m)^4)$ time and $O((n + m)^3)$ space. Note that $\mathcal{S}(\pi_3) \geq S_{HV}(n, m) \geq \mathcal{S}(HV(\pi^*), \pi^*)$.

The time and space complexity for computing π_3 can be further improved.

Theorem 4.2 *Given two polygonal curves P and Q of size n and m respectively, and an error threshold δ , a $\frac{1}{3}$ -approximation of $\mathcal{D}_\partial(P, Q)$ under the L_1 or L_∞ norm can be computed in $O((n + m)^3)$ time and $O((n + m)^2)$ space.*

Proof: We first project the vertices of all good regions in each column to every horizontal cell boundaries in this column, and perform similar projection for each row of M as well. Afterwards, there are $O(m)$ Steiner vertices on each horizontal cell boundary, and $O(n)$ vertices on each vertical cell boundary. We now refine $S_{HV}(i, j)$ as defined before to $S_h(k, i, j)$ and $S_v(k, i, j)$, where $S_h(k, i, j)$ (resp., $S_v(k, i, j)$) is the best S_{HV} -score of any monotone path from $M[1][1]$ to the k th point on the (i, j) th horizontal edge (resp., the k th point on the (i, j) th vertical edge).

Now given a path π , $HV(\pi)$ has at most $O(n + m)$ components (as separated by turning cells) such that each component either lies within the same column or within the same row.

We say that π is critical if each component of $HV(\pi)$ is critical (recall the definition of critical paths in Section 3 for the special case). By applying Lemma 3.1 to each such component, any conformal path can be modified into a critical path without decreasing its S_{HV} -score. Hence we now only need to consider critical paths.

Consider the cell $M[i][j]$. Assume that the S_v - and S_h -scores have been computed for the Steiner points on its left and bottom edges already. We now wish to compute $S_h(k, i, j + 1)$ for all $1 \leq k \leq N$, where $N = O(m)$ is the number of Steiner points. (The computation for $S_v(k, i + 1, j)$ s is symmetric.) There are two possible cases: (i) cell $M[i][j]$ is a turning cell, and (ii) optimal path enters from bottom and leaves from top. For case (i), ignoring the score that can be produced in this turning cell, we can take the largest score from left boundary. For case (ii), we inspect all potential entry points on the bottom cell boundary of $M[i][j]$, and return the one with largest score. Overall, we have that

$$S_h(k, i, j + 1) = \max\left\{\max_1 S_v(l, i, j), \max_{1 \leq l \leq k} \{S_h(l, i, j) + W_{M[i][j]}(l, k)\}\right\},$$

where $W_{M[i][j]}(l, k)$ is the score of an optimal critical path within the cell $M[i][j]$ and connecting the l th point on the bottom boundary of cell $M[i][j]$, to the k th point on the upper boundary. For the first term, we maintain the maximum score value for each vertical boundary edge. For the second term, we can apply Lemma 3.3 to compute $S_h(k, i, j + 1)$ s, for all $1 \leq k \leq N$, in $O(N) = O(m)$ time.

Putting everything together yields the result. ■

4.2 A $(1 - \varepsilon)$ -approximation Algorithm

We now describe a more general $(1 - \varepsilon)$ -approximation algorithm for $\mathcal{D}_\partial(P, Q)$ that runs in polynomial time for any L_p norm (although its performance is better under the L_1 and L_∞ norms). For simplicity, we use L_∞ in what follows.

Given an error threshold $\delta > 0$ and the corresponding free space diagram $M = M_\delta(P, Q)$, consider a cell $M[i][j]$ and the good region within it, denoted by Ω . We subdivide $\partial\Omega$, the boundary of Ω , into intervals of size $\varepsilon\omega/(c(n + m))$ each, where ω is the length of $\partial\Omega$. Let $\Delta^* = \mathcal{D}_\partial(P, Q)$, and observe that since $\omega = O(\Delta^*)$, one can choose the constant c appropriately, so that the L_1 -norm length of each interval is at most $\varepsilon\Delta^*/(4(n + m))$. Next, project these subdividing points onto the four cell boundaries. If we perform this for every cell in M , then every grid edge in M , say the (i, j) th horizontal edge, has two sets of projections, once as the lower boundary edge of the cell $M[i][j]$, and once as the upper boundary edge of $M[i][j - 1]$. Hence the total number of *Steiner points* on each edge is $N = O((n + m)/\varepsilon)$. The diagram M together with these Steiner points is an **augmented diagram** \widehat{M} . The following observation is straightforward, where $W_C(x, y)$, as defined earlier, is the best score connecting x to y within cell C .

Observation 4.3 *Given any cell $C = M[i][j]$ of M , let x be an entry point from the lower or left boundary of C and y an exit point from the upper or right boundary edge of C . Let w be the length of the boundary of the good region in cell C , and x' (resp. y') the closest Steiner points to the left of or below x (resp. y). Then,*

$$W_C(x, y) - \varepsilon\omega/(n + m) \leq W_C(x', y') \leq W_C(x, y) + \varepsilon\omega/(n + m).$$

A **constraint conformal path** of \widehat{M} is a conformal path of M such that its intersections with the cell boundaries are either grid points of M , or Steiner points of \widehat{M} . The following lemma follows from Observation 4.3.

Lemma 4.4 *Any conformal path π in the free space diagram M can be modified into a constraint conformal path π' such that $(1 - \varepsilon)\mathcal{S}(\pi) \leq \mathcal{S}(\pi') \leq \mathcal{S}(\pi)$.*

The optimal constraint conformal path in M can be computed, in $O((n + m)^4/\varepsilon^2)$ time, under any L_p norm using dynamic programming. The running time can be improved to $O((n + m)^3/\varepsilon^2)$ under the L_1 or L_∞ norms using the same technique as in Lemma 3.3.

Theorem 4.5 *Given a parameter $\delta > 0$ and two polygonal curves P and Q of size n and m , respectively, one can compute, in $O((n + m)^4/\varepsilon^2)$ time, a $(1 - \varepsilon)$ -approximation to $\mathcal{D}_\delta(P, Q)$, using $O((n + m)^2/\varepsilon)$ space, for any L_p norm. For the L_1 or L_∞ norms, the running time of the algorithm is $O((n + m)^3/\varepsilon^2)$, and it uses $O((n + m)^2/\varepsilon)$ space.*

5 Double-sided Approximation of $\mathcal{D}_\delta(P, Q)$

In this section, we present a double-sided approximation algorithm for $\mathcal{D}_{\delta,\delta}(P, Q)$. The output of the algorithm is a number τ which is a $(\frac{1}{3+3\sqrt{d}}, \sqrt{d})$ -approximation to $\mathcal{D}_{\delta,\delta}(P, Q)$ for two polygonal curves P and Q in \mathbb{R}^d under the L_2 norm; namely, $\mathcal{D}_{\delta,\delta}(P, Q)/(3+3\sqrt{d}) \leq \tau \leq \mathcal{D}_{\delta,\sqrt{d}\delta}(P, Q)$. Using the same framework one can also compute a $(1/(3 + 3\sqrt{d}), d)$ -approximation to $\mathcal{D}_\delta(P, Q)$ under any L_p norm. In what follows, for simplicity of exposition, we assume that P and Q are polygonal curves in the plane — the same algorithm and analysis can be extended to higher dimensions. Furthermore, We will only describe a $(\frac{1}{1+\sqrt{2}}, \sqrt{2})$ -approximation algorithm for the special case where P has only one edge. Using the same framework as in Section 4.1, this leads to a $(\frac{1}{3+3\sqrt{2}}, \sqrt{2})$ -approximation algorithm for the general case.

5.1 Algorithm outline

Given a segment s and a polygonal curve $Q = \langle q_1, q_2, \dots, q_m \rangle$ in the plane, the goal is to compute a $(\frac{1}{1+\sqrt{2}}, \sqrt{2})$ -approximation of $\mathcal{D}_{\delta,\delta}(s, Q)$ under the L_2 norm. Let $F_{\delta,k} : s \rightarrow \mathbb{R}$ be the function defined as $F_{\delta,k}(x) = \mathcal{D}_{\delta,\delta}(s[0, x], Q_k)$, where $s[0, x]$ is the subsegment of s from $s(0)$ to $s(x)$ and $Q_k = \langle q_1, \dots, q_k \rangle$ is the subchain of Q spanned by its first k vertices. The idea is to maintain F_k in a bottom-up manner: start with $k = 1$, and in each round, update F_k to obtain F_{k+1} . In the end, we have that $F_m(1) = \mathcal{D}_\delta(s, Q)$.

Unfortunately, the structure of F_k seems to be quite complicated. We conjecture that its descriptive complexity is polynomial but currently we have no proof — we leave this as an open problem for further research. Hence we consider approximating these functions. First, we *approximate the distance metric* by replacing the unit ball with a unit square ψ such that one of its sides is parallel to the segment s . Note that if we rotate s so that it is parallel to the x -axis, then ψ coincides with the unit ball (i.e., square) for the L_∞ norm. Thus, from now on, we assume, without loss of generality, that s is the interval $[0, 1]$ on the x -axis, and our algorithm uses the L_∞ norm to approximate the L_2 norm. (Note that this is different

from using the L_∞ norm for an arbitrary segment s .) Furthermore, for any $x \in s = [0, 1]$, we now redefine $F_k(x)$ to be the optimal partial matching between the segment $[0, x]$ and the chain Q_k under the L_∞ norm (while $\mathcal{D}_{\partial, \delta}(P, Q)$ still denotes the optimal Fréchet distance under the L_2 norm). Since the unit square is contained inside a disk of radius $\sqrt{2}$ (and \sqrt{d} in \mathbb{R}^d), we have the following.

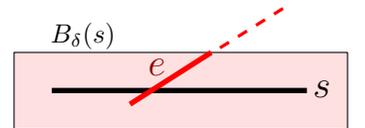
Observation 5.1 *The quantity $F_m(1)$ is a $(1, \sqrt{2})$ -approximation for $\mathcal{D}_{\partial, \delta}(s, Q)$. Furthermore, an α -approximation for $F_m(1)$ induces an $(\alpha, \sqrt{2})$ -approximation of $\mathcal{D}_{\partial, \delta}(s, Q)$.*

For any $k \in \{1, \dots, m\}$, since $F_k(x)$ is the optimal partial matching between the segment $[0, x]$ and the chain Q_k under the L_∞ norm, it is easy to verify that F_k is a *monotone* (non-decreasing) piece-wise linear (PL) function (its detailed structure is described below). To further improve the efficiency of our algorithm, we introduce the *second level of approximation*: in each round, instead of F_k , we maintain another monotone PL function G_k that approximates F_k . In particular, the complexity of G_k is bounded by $O(m \log m)$, and each update of G_k into G_{k+1} takes $O(\log^2 m)$ amortized time. Hence overall, the resulting algorithm runs in $O(m \log^2 m)$ time. In the end, $G_m(1)$ provides a $\frac{1}{1+\sqrt{2}}$ -approximation for $F_m(1)$, which, combined with Observation 5.1, implies that it is a $(\frac{1}{1+\sqrt{2}}, \sqrt{2})$ -approximation to the optimal partial Fréchet distance between the segment s and polygonal curve Q under the L_2 norm, i.e., $\mathcal{D}_{\partial, \delta}(s, Q)/(1 + \sqrt{2}) \leq G_m(1) \leq \mathcal{D}_{\partial, \sqrt{2}\delta}(s, Q)$.

In the remainder of this section, given a segment $s = [0, 1]$ and a polygonal curve Q of size m , we first describe the structure of F_k for any $k \in \{1, \dots, m\}$. We then show how to approximate the F_k s by G_k s to obtain a double-sided approximation algorithm for $\mathcal{D}_{\partial}(s, Q)$ in this special case, which then results in a double-sided approximation algorithm in the general case.

5.2 The structure and computation of the F_k s

We describe the structure of F_k s by studying the relation between F_k and F_{k+1} . In particular, given F_k , we wish to compute F_{k+1} when a new edge $e = (q_k, q_{k+1})$ is added to subchain Q_k . Let



$$B_\delta(X) = \{y \in \mathbb{R}^d \mid \exists x \in X \text{ s.t. } d_\infty(x, y) \leq \delta\}$$

denote the δ -*neighborhood* of the object X under the L_∞ norm. Since only the intersection between e and $B_\delta(s)$ may contribute to the partial matching between e and s , we assume from now on that e lies completely inside $B_\delta(s)$ (i.e., the dashed piece of e in the figure above is ignored). We also assume that the angle between e and s is at most $\pi/2$ (s and e are oriented by their ordering along the two curves); the case where the angle is obtuse is simpler and it is thus omitted.

Let $J_a(x)$ denote the score of the best matching between the edge e and the subsegment $[a, x] \subseteq s$. Since we are using the L_∞ norm and s is horizontal, the good region in the free space diagram of s and e is a parallelogram as shown in Figure 6, and the slope of the diagonal sides of this parallelogram is $\sqrt{1 + \rho(e)^2}$, where $\rho(e)$ is the slope of e . Clearly, the function $J_a(x)$ is the maximum length (under the L_1 metric) segment inside the good region that lies between the two vertical lines ℓ_a and ℓ_x passing through a and x , respectively. There are four critical values of x where the function $J_a(x)$ may change its behavior (the values x_1, x_2, x_3 and x_4 as depicted in Figure 6). For simplicity of exposition, we assume from now on that $x_2 < x_3$, unless otherwise specified.

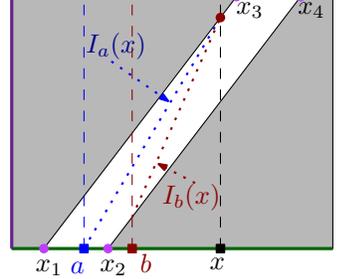


Figure 6: good region

Let $\mathcal{V}(a) = J_a(a)$ denote the length of the intersection of the vertical line ℓ_a with the good region. Clearly, $\mathcal{V}(a) = 0$ for $a \notin [x_1, x_4]$. It is easy to verify that within the range $[x_1, x_4]$, \mathcal{V} is a piecewise linear function of three pieces: $\mathcal{V}(a) = \sqrt{1 + \rho(e)^2}(a - x_1)$ for $a \in [x_1, x_2]$, $\mathcal{V}(a) = \mathcal{V}(x_2)$ for $a \in [x_2, x_3]$, and $\mathcal{V}(a) = \mathcal{V}(x_2) - \sqrt{1 + \rho(e)^2}(a - x_3)$ for $a \in [x_3, x_4]$. See Figure 7. If $x_2 \geq x_3$, then the graph of \mathcal{V} has the same symmetric shape, but with the roles of x_2 and x_3 switched.

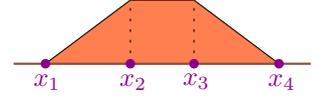


Figure 7: $\mathcal{V}(\cdot)$

If the vertical lines ℓ_a and ℓ_x both intersect the good region, then the longest segment inside the good region is simply the segment connecting the bottom feasible point on ℓ_a to the top feasible point on ℓ_x (dotted segments in Figure 6). As $J_a(x)$ is the L_1 -norm length of this segment, we have that for $a \in [x_1, x_3]$,

$$J_a(x) = \begin{cases} \mathcal{V}(a) + \rho_I(x - a) & \text{if } x \in [x_1, x_3] \\ \mathcal{V}(a) + \rho_I(x_3 - a) + (x - x_3) & \text{if } x \in (x_3, x_4] \\ J_a(x_4) & \text{if } x \in (x_4, 1], \end{cases}$$

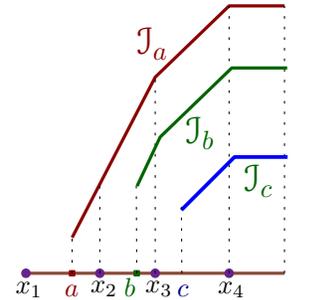


Figure 8: J_a, J_b, J_c .

where $\rho_I = 1 + \sqrt{1 + \rho(e)^2}$ is a constant that only depends on e . For the case where $a \in [x_3, x_4]$, $J_a(x) = \mathcal{V}(a) + (x - x_3)$ for $x \in [a, x_4]$, and $J_a(x) = J_a(x_4)$ for $x \in (x_4, 1]$. See Figure 8. Note that the above formula holds for both the case $x_2 < x_3$ and $x_2 \geq x_3$, although we have assumed that $x_2 < x_3$ for simplicity of exposition.

Computing F_{k+1} from F_k . It is easy to verify that the following holds.

$$F_{k+1}(x) = \begin{cases} F_k(x) & \text{if } x \in [0, x_1) \\ \max_{a \in [x_1, x]} \{F_k(a) + J_a(x)\} & \text{if } x \in [x_1, x_4] \\ \max\{F_{k+1}(x_4), F_k(x)\} & \text{if } x \in (x_4, 1]. \end{cases} \quad (1)$$

Indeed, the best way to match s up to the point x with Q_{k+1} , is either the best way to match $[0, x]$ with Q_k , or by matching some fraction $[a, x] \subseteq s$ with the $(k + 1)$ th segment e of Q_{k+1} . Now decompose $J_a(x)$ into two functions, $J_a(x) = \mathcal{V}(a) + \mathcal{R}_a(x)$. Since $\mathcal{V}(a)$ is the length of the intersection of ℓ_a with the good region, we can interpret $\mathcal{R}_a(x)$ as the L_1 -norm length of the segment in the good region connecting the top intersection point of ℓ_a to the top intersection point of ℓ_x .

Claim 5.2 (i) For any a , $\mathcal{R}_a(a) = 0$, and for any $a \leq b \leq x$, it holds that $\mathcal{R}_a(x) \geq \mathcal{R}_b(x)$.
(ii) For any $x \geq b \geq a$ it holds that $\mathcal{R}_a(x) - \mathcal{R}_b(x)$ is a constant that depends only on a and b .
(iii) For any $a, x \in [x_1, x_4]$ with $a \leq x$, it holds that $\mathcal{R}_a(x) \geq x - a$.

Proof: Claim (i) and (iii) follows directly from the above interpretation of $\mathcal{R}_a(\cdot)$. As for (ii), observe that $\mathcal{R}_a(x)$ and $\mathcal{R}_b(x)$ have the same derivative, and they are continuous functions; namely, the difference between them is a constant (recall Figure 8). ■

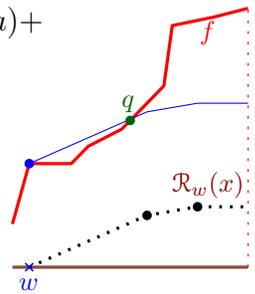
We now describe how to compute $F_{k+1}(x)$ based on Eq. (1). Let N be the descriptive complexity of F_k . In the following, we will show that F_{k+1} has complexity $N + O(1)$, and can be computed in $O(N)$ time. First, observe that for $x \in [x_4, 1]$, once $F_{k+1}(x_4)$ is known, we simply scan $[x_4, 1]$ and set $F_{k+1}(x) = F_{k+1}(x_4)$ if $F_k(x) \leq F_{k+1}(x_4)$. As F_k is a monotone piecewise linear function, it is easy to see that this will increase the complexity by at most one, and the process takes $O(N)$ time. The other case is when $x \in [x_1, x_4]$. Here we wish to compute functions $F_k(a) + \mathcal{J}_a(x)$ for every a , and then for a given x , take the largest value that such a function can generate. We implement this goal in two steps. First, we compute the function $f(a) = F_k(a) + \mathcal{V}(a)$; note that $F_k(a) + \mathcal{J}_a(x) = F_k(a) + \mathcal{V}(a) + \mathcal{R}_a(x) = f(a) + \mathcal{R}_a(x)$. As $\mathcal{V}(a)$ is a piecewise linear function consisting of three pieces, and $F_k(a)$ is also a piecewise-linear function, the function $f(\cdot)$ has complexity $N + O(1)$ — we introduce four new vertices (i.e., x_1, \dots, x_4) to the graph of the function f . The second step is to compute $F_{k+1}(x) = \max_a(f(a) + \mathcal{R}_a(x))$, for any x . By Claim 5.2 we have the following observation.

Claim 5.3 Consider $a \leq b$ and x such that $f(a) + \mathcal{R}_a(x) = f(b) + \mathcal{R}_b(x)$. Then $f(a) + \mathcal{R}_a(b) = f(b)$. Furthermore, if $f(a) + \mathcal{R}_a(b) \geq f(b)$, then $f(a) + \mathcal{R}_a(x) \geq f(b) + \mathcal{R}_b(x)$.

Proof: By Claim 5.2, $\mathcal{R}_a(x) - \mathcal{R}_b(x) = \mathcal{R}_a(b) - \mathcal{R}_b(b) = \mathcal{R}_a(b)$. Since $\mathcal{R}_b(b) = 0$, $f(b) = f(a) + \mathcal{R}_a(x) - \mathcal{R}_b(x) = f(a) + \mathcal{R}_a(b)$. The second claims follows from a similar argument. ■

Now for a point $(a, f(a))$ on the graph of f , consider the function $h_a(x) = f(a) + \mathcal{R}_a(x)$. Visually, this function is the result of “attaching” the graph of $\mathcal{R}_a(x)$ to the graph of f at the point $(a, f(a))$. Let y be the smallest value (larger than a) such that $h_a(y) = f(y)$. Claim 5.3 implies that no point $b \in [a, y]$ can generate a larger value for any $x \geq b$ than $h_a(x)$; that is, $f(b) + \mathcal{R}_b(x) \leq f(a) + \mathcal{R}_a(x)$. Thus there is no need to consider $f(b) + \mathcal{R}_b(x)$ for such b s when computing F_{k+1} .

This leads to the following algorithm to compute $F_{k+1}(x) = \max_{a \in [x_1, x_4]} (F_k(a) + \mathcal{J}_a(x))$ for $x \in [x_1, x_4]$. The algorithm sweeps the graph of $f(x)$ from left to right. At each point x , we check whether attaching the graph of $\mathcal{R}_x(\cdot)$ to f at x , results in a larger value immediately to the right of x or not. This decision can be made by considering the slope of the two graphs at this point. In fact, since the two functions are piecewise linear, we need to perform this attachment decision only at vertices of the two graphs. If the slope of f is larger, say at a point w , we attach the graph of $\mathcal{R}_w(\cdot)$ to $f(w)$, and compute the first intersection point, say $q = (y, f(y))$, of this attached polygonal line with the graph of $f(\cdot)$ to the right of w ; we call q a *hitting point*. Now, for any point $x \in [w, y]$, we update $F_{k+1}(x) = f(w) + \mathcal{R}_w(x)$. We then continue this sweeping process starting from y , until we reach x_4 .



Lemma 5.4 *The above algorithm computes $F_{k+1}(\cdot)$ in $O(N)$ time and the resulting function has complexity $N + O(1)$, where N is the complexity of F_k .*

Proof: It is easy to verify that the resulting function is indeed the required $F_{k+1}(x)$. We now bound its descriptive complexity. Recall that N is the complexity of $F_k(\cdot)$ and $f(\cdot)$ has complexity $N + O(1)$. First, observe that at most two new vertices of $F_{k+1}(x)$ are created due to the vertices of $\mathcal{R}_a(\cdot)$ for all $a \in [x_1, x_4]$, as this family of functions have vertices always at same coordinates (i.e, x_1, x_3 and x_4), and the number of vertices of a graph in this family is at most 3. The only other new vertices are those hitting points created by attaching the graph of some $\mathcal{R}_w(\cdot)$ to $f(w)$ during the above process (note that w itself is an original vertex of f). Consider the hitting vertex $v = (z, f(z))$ corresponding to \mathcal{R}_w , and recall that the portion of f between $(w, f(w))$ and $(z, f(z))$ was replaced by the new polygonal chain of $\mathcal{R}_w(\cdot)$. If \mathcal{R}_w contains vertices between $[w, z]$, then we can charge the complexity of inserting the hitting vertex v to vertices of \mathcal{R} , which is $O(1)$ for the entire interval $[x_1, x_4]$. Otherwise, the graph of \mathcal{R}_w over $[w, z]$ is a single segment, and as such it hides at least one old vertex of $f(x)$ which was removed, and we charge the new vertex to this old vertex. It then follows that $F_{k+1}(\cdot)$ has complexity $N + O(1)$ and can be computed in $O(N)$ time. ■

Computation of all F_k s. To recap, our algorithm starts with F_0 (i.e, $k = 0$), which is a constant function of value zero. It then performs m rounds, updating F_k to obtain F_{k+1} in $O(N_k)$ time in each round, where N_k is the complexity of F_k . Since in each round, the complexity of F_{k+1} can increase by at most $O(1)$, overall, F_m is a monotone PL function of complexity $O(m)$, and can be computed in $O(m^2)$ time and $O(m)$ space. This means that $F_m(1)$, the best partial matching between an edge s and a polyline Q of size m under L_∞ metric, can be computed in $O(m^2)$ time.

Lemma 5.5 *Given a segment s and a polygonal curve Q with m segments, one can compute, in $O(m^2)$ time and $O(m)$ space, a number τ , which is a $(1, \sqrt{2})$ -approximation to the optimal partial Fréchet distance between s and Q under the L_2 norm, that is, $\mathcal{D}_{\partial, \delta}(s, Q) \leq \tau \leq \mathcal{D}_{\partial, \sqrt{2}\delta}(s, Q)$.*

5.3 Approximation algorithm for $F_m(1)$

In this section, we describe an efficient near-linear time algorithm to $1/3$ -approximate $F_m(1)$. It follows the ideas used in the previous section, and runs in m rounds. However, in each round, instead of F_k , it maintains another **monotone** PL function G_k , so that for any $x \in s = [0, 1]$, we have $F_k(x)/\gamma - k\varepsilon\Delta^*/m \leq G_k(x) \leq F_k(x)$, where $\Delta^* = \mathcal{D}_{\partial}(s, Q)$ and γ is a constant that will be specified shortly. In the end, $G_m(1)$ provides the required approximation for $F_m(1)$.

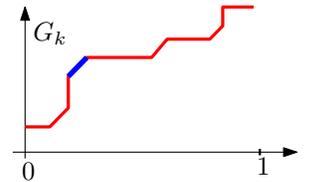


Figure 9: G_k .

5.3.1 Basic setup

Each G_k has the **tridirectional property**; that is, each segment in its graph is either horizontal, vertical or with slope 1. See Figure 9. We refer to vertices of the graph of such a

PL-function connecting a diagonal or a vertical edge with a horizontal one as **base vertices**. Given a PL-function, a **base query** for a given real number x , asks for the first base vertex on the graph of the function to the right of x . Our goal is to compute G_{k+1} from G_k , in $O(\log^2 m)$ amortized time (instead of in $O(m)$ time, as in the previous section). To achieve this goal, we first need a better data structure for representing a given PL-function.

Lemma 5.6 *Suppose $H : [b_1, b_{l+1}] \rightarrow \mathbb{R}$ is a PL-function of $l + 1$ break points b_1, \dots, b_{l+1} , associated with a sequence of linear functions $H_i : [b_i, b_{i+1}] \rightarrow \mathbb{R}$, for $1 \leq i \leq l$. One can build a data-structure to represent H so that the following operations are supported: (i) insert/delete a consecutive sequence of k break points, (ii) evaluate $H(x)$, or return H_i for any $1 \leq i \leq l$, (iii) base query if H has the tridirectional property, and (iv) lift the curve by adding a constant to an interval the function is defined over. The first operation can be performed in $O(k + \log N)$ time, and the remaining operations can be performed in $O(\log N)$ time.*

Proof: We use a balanced binary tree $T = T(H)$ for representing H . More specifically, T is the following augmented balanced interval tree: From left to right, the i th leaf of T , denoted by l_i , corresponds to interval $[b_i, b_{i+1}]$. Every leaf node l_i stores a linear function $f_i : [b_i, b_{i+1}] \rightarrow \mathbb{R}$, called the **base function** at leaf l_i . Each internal node $v \in T$ is associated with an interval $I_v = [b_{v_l}, b_{v_r+1}]$ where v_l and v_r are the indices of the left-most and right-most leaves of the subtree rooted at v , respectively. It also stores an **additive constant** c_v for the interval I_v . To obtain H_i , we first find the path from the root of T to l_i ; let A_i denote the set of internal nodes along this path. We then have that $H_i = f_i + \sum_{v \in A_i} c_v$.

Clearly, for a PL function of complexity l , such a balanced binary tree representation has size $O(l)$, and supports each operation such as insertion/deletion, or computing H_i , in $O(\log l)$ time. Given any interval $[b_i, b_j]$, one can also identify $O(\log l)$ number of canonical and disjoint intervals whose union forms $[b_i, b_j]$, and each such interval corresponds to a leaf or an internal node of the tree T . We call this set of intervals the **canonical decomposition of $[b_i, b_j]$** .

Now suppose we wish to add a constant c over an interval $[a, b]$. First, we insert a and b if they are not already break points of H . Next, we identify the set of nodes corresponding to the canonical decomposition of $[a, b]$, and increase the additive constant c_v associated with each such node $v \in T$ by c . Obviously, the entire process takes $O(\log l)$ time.

If the input function H also has the tridirectional property, then we can further modify T to answer each base query in $O(\log l)$ time. The required modifications are straightforward and are thus omitted. ■

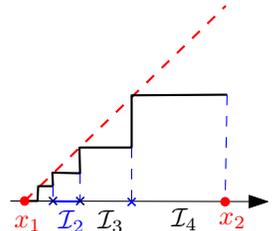
5.3.2 Updating the G_k s

Given the function G_k , of complexity N , represented by an instance T of the data-structure of Lemma 5.6, we wish to update T into representing G_{k+1} . Again, let x_1, \dots, x_4 be the projection of the vertices of the white region onto s (recall Figure 6). We describe only the case when $x \in [x_1, x_4]$, since the other cases are similar and easier. For $x \in [x_1, x_4]$, recall that $F_{k+1}(x) = \max_{a \in [x_1, x]} \{F_k(a) + \mathcal{J}_a(x)\}$, and we have separated the update of F_k into two stages, lifting F_k by \mathcal{V} to function f (the *lifting stage*), and then computing $\max_a (f(a) + \mathcal{R}_a(x))$ for

each x (the *enveloping stage*). We follow the same two-stage framework here to update G_k , but introduce approximation at each stage.

The Lifting stage. Previously, we first lift $F_k(x)$ to $F_k + \mathcal{V}(x)$ for every $x \in [x_1, x_4]$. Here, we approximate \mathcal{V} (recall Figure 7) by the function $\widehat{\mathcal{V}}$ specified below.

We remind the reader that for simplicity of exposition, we have assumed that $x_2 < x_3$. If this is not the case, then the roles of x_2 and x_3 below in the construction of $\widehat{\mathcal{V}}$ should switch. Let $\varepsilon > 0$ be some constant to be specified shortly. For $x \in [x_2, x_3]$, we set $\widehat{\mathcal{V}}(x) = \mathcal{V}(x)$, which is a constant. For the interval $[x_1, x_2]$, we replace \mathcal{V} by an exponential staircase function (see figure on the right): First, subdivide $[x_1, x_2]$ into $\beta = O(\log(m/\varepsilon))$ intervals, with the i th interval $\mathcal{I}_i = [x_1 + L/2^{\beta-i}, x_1 + L/2^{\beta-i-1}]$, for $i = 1, \dots, \beta - 1$, where $L = x_2 - x_1$. Also, set $\mathcal{I}_0 = [x_1, x_1 + L/2^{\beta-1}]$. For $x \in \mathcal{I}_i$, we set $\widehat{\mathcal{V}}(x) = \min_{y \in \mathcal{I}_i} \mathcal{V}(y) = \mathcal{V}(x_1 + L/2^{\beta-i})$. The interval $[x_3, x_4]$ is handled in a symmetric manner. Since \mathcal{V} is a linear function over the interval $[x_1, x_2]$, it is easy to verify that $\widehat{\mathcal{V}}(\cdot)$ is a $\frac{1}{2}$ -approximation to $\mathcal{V}(\cdot)$ on $[x_1, x_2]$, except for $x \in \mathcal{I}_0$. For $x \in \mathcal{I}_0$, we have that



$$\mathcal{V}(x) - (\varepsilon\mathcal{V}(x_2)/m) \leq \mathcal{V}(x) - \mathcal{V}(x_1 + L/2^{\beta-1}) \leq \widehat{\mathcal{V}}(x) \leq \mathcal{V}(x),$$

that is, there is an *additive* approximation error of at most $\varepsilon\mathcal{V}(x_2)/m$. Furthermore, as $\mathcal{V}(x_2) \leq \Delta^* = \mathcal{D}_\partial(P, Q)$, we have that, for any $x \in [x_1, x_2]$, it holds that

$$\mathcal{V}(x)/2 - \varepsilon\Delta^*/m \leq \widehat{\mathcal{V}}(x) \leq \mathcal{V}(x). \quad (2)$$

The lifting stage now consists of $O(\log(m/\varepsilon))$ modifications of G_k by adding to it the functions defined above on the intervals $\mathcal{I}_0, \dots, \mathcal{I}_\beta$, respectively. Using the data-structure of Lemma 5.6, each such update can be done in $O(\log N)$ time. Overall, the lifting stage takes $O(\log(N) \log(m/\varepsilon))$ time, and creates at most $\beta + 1$ new break points. Let $H(\cdot)$ denote the resulting piecewise linear function. Clearly, $H(\cdot)$ satisfies the tridirectional property, and has complexity at most $N + O(\log(m/\varepsilon))$.

Enveloping stage. In this stage, we use $\widehat{\mathcal{R}}_a(x) = x - a$ for $x \in [x_1, x_4]$ to replace \mathcal{R}_a and approximate $F_{k+1}(x)$ by $G_{k+1}(x) = \max_{a \leq x} (H(a) + \widehat{\mathcal{R}}_a(x))$. In particular, attaching the graph of $\widehat{\mathcal{R}}_a$ at the point $(a, H(a))$ looks like shooting a ray of slope 1 from the point $(a, H(a))$. As such, G_{k+1} is the upper envelop of all such rays. It follows from Claim 5.2 (iii) that $\widehat{\mathcal{R}}_a(x) \leq \mathcal{R}_a(x)$. Observe that Claim 5.3 also holds for $\widehat{\mathcal{R}}_a$.

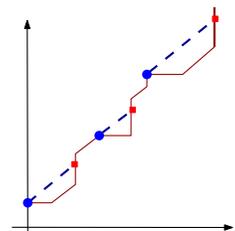


Figure 10: From H to G_{k+1}

To compute G_{k+1} , we start with the first base vertex of H , say $(w_1, H(w_1))$, and shoot a diagonal ray from it. The first intersection point $(y_1, H(y_1))$ between this diagonal ray and the graph of H is a *hitting vertex*. By Claim 5.3, for any point $(x, H(x))$ with $x \in (w_1, y_1)$, its ray will not appear on the upper envelope. Hence the next ray on the upper envelope will be originated from the first base point, say w_2 , after y_1 . We then repeat the same process starting from w_2 until we reach x_4 . See Figure 10.

It is easy to verify that the resulting function G_{k+1} satisfies the tridirectional property. Furthermore, each diagonal ray can only hit a vertical segment in the graph of H , as H is tridirectional. Hence, each hitting vertex corresponds to some break point of H . Since the starting point of each attached ray is also a vertex of the graph of H (thus corresponding to a break point of H), the enveloping stage will **not** create any new break points. This fact leads to a global analysis of the time complexity for the enveloping stage, which in turn results in the following lemma.

Lemma 5.7 *Computing G_m takes $O(m \log(m) \log(m/\varepsilon))$ time.*

Proof: For the k th iteration, let r_k be the number of new rays appearing in the upper envelop, and w_k the number of break points of H covered by such rays (which are then deleted). Since each base query can be answered in $O(\log N_k)$ time, the above procedure updates H into G_{k+1} , as well as adjusting the data structure T , in $O(w_k + r_k \log N_k)$ time, where N_k is the complexity of G_{k+1} .

Observe that w_k and r_k can be $\Omega(N_k)$. However, each of the w_k vertices covered by the rays will be deleted from the list of break points forever. Since only the lifting stage can create new break points, and in each iteration, at most $O(\log(m/\varepsilon))$ new break points are being created, overall there are most $O(m \log(m/\varepsilon))$ break points ever created. Hence we have that $\sum_{k=1}^m w_k = O(m \log(m/\varepsilon))$. To bound $\sum r_k$, observe that once an edge becomes a diagonal edge, it will never becomes horizontal again until it is deleted. (New break points may be inserted into the corresponding interval later.) Hence for each endpoint of this interval, it will never becomes a base vertex again. Since there are $O(m \log(m/\varepsilon))$ break points ever created, we have that $\sum_{k=1}^m r_k = O(m \log(m/\varepsilon))$ as well. Putting everything together, the total cost for m rounds of updates to compute G_m from G_0 is

$$O(m \log(m/\varepsilon) + \sum_{k=1}^m (w_k + r_k \log m)) = O(m \log m \log(m/\varepsilon)).$$

■

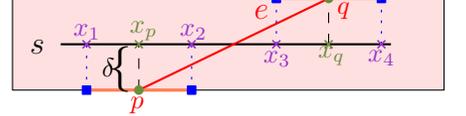
5.3.3 Approximation quality

Lemma 5.8 *For any $0 \leq k \leq m$, we have that $F_k(x)/\gamma - k\varepsilon\Delta^*/m \leq G_k(x) \leq F_k(x)$, where $\gamma = 2 + \sqrt{2}$ and $\Delta^* = F_m(1)$ is the optimal partial matching between segment s and curve Q under the L_∞ metric.*

Proof: The proof is by induction. Indeed, $G_0(x) = 0$ and as such $G_0(x) = F_0(x)$ for $x \in [0, 1]$. Thus the lemma holds for $k = 0$. Next, assume that, for any $x \in [0, 1]$, $F_k(x)/\gamma - k\varepsilon\Delta^*/m \leq G_k(x) \leq F_k(x)$. We wish to show that $F_{k+1}(x)/\gamma - (k+1)\varepsilon\Delta^*/m \leq G_{k+1}(x) \leq F_{k+1}(x)$. The right inequality is straightforward. We now focus on the left inequality for $x \in [x_1, x_3]$; the cases for $x < x_1$ or $x > x_3$ are simpler and as such they are omitted.

For $x \in [x_1, x_3]$, our algorithm approximates $\mathcal{J}_a(x) = \mathcal{V}(a) + \rho_I(x - a)$ by $\widehat{\mathcal{J}}_a(x) = \widehat{\mathcal{V}}(a) + (x - a)$, where $\rho_I = 1 + \sqrt{1 + \rho(e)^2}$. Intuitively, as long as ρ_I is relatively small, this is a reasonable approximation, as $\widehat{\mathcal{V}}(\cdot)$ is a “decent” approximation to $\mathcal{V}(\cdot)$. Indeed, consider the error induced by $\widehat{\mathcal{V}}$. If $a \in [x_2, x_3]$, then $\widehat{\mathcal{V}}(a) = \mathcal{V}(a)$. For interval $[x_1, x_2]$, as described in Eq. (2), $\widehat{\mathcal{V}}(x) \geq \mathcal{V}(x)/2 - \varepsilon\Delta^*/m$.

For the error induced by the enveloping stage, if $|\rho(e)| \leq 1$, then we have that $\rho_I(x-a) \leq (1+\sqrt{2})(x-a)$, implying that, for any $a < x$, we have that $\widehat{\mathcal{J}}_a(x) \geq \mathcal{J}_a(x)/(1+\sqrt{2})$ for an ε small enough (the ε we choose later will satisfy its requirement here).



The case for $|\rho(e)| > 1$ is more involved. In this case, the angle between e and s is larger than 45 degrees. Observe that if x_p and x_q are the projections of the endpoints of e into s , then $x_1 = x_p - \delta$, $x_2 = x_p + \delta$, $x_3 = x_q - \delta$, and $x_4 = x_q + \delta$, see figure on the right. Now, if the angle between e and s is more than 45 degrees, then the projection of e on the x -axis is shorter than its projection on the y axis (which is 2δ). Hence in this case $x_3 < x_2$.

For $x_3 < x_2$, the shape of the graph of $\mathcal{V}(\cdot)$ remains the same (as in Figure 7), but with the role of x_2 and x_3 switched. Hence, for any $x_1 < a < x_3$, we have that $\mathcal{V}(x) = (\rho_I - 1)(x - x_1)$. Together with the fact $\rho_I - 1 \geq \sqrt{2}$, this implies that

$$\begin{aligned} \mathcal{J}_a(x) &= \mathcal{V}(a) + \rho_I(x-a) = (\rho_I - 1)(a - x_1) + \rho_I(x-a) = (\rho_I - 1)(x - x_1) + (x-a) \\ &\leq (\rho_I - 1)(x - x_1) + (x - x_1) \leq (1 + 1/\sqrt{2})\mathcal{V}(x). \end{aligned}$$

On the other hand, recall that we have subdivided $[x_1, x_2]$ (or $[x_1, x_3]$ if $x_3 < x_2$) into $\beta + 1$ intervals $\mathcal{I}_1, \dots, \mathcal{I}_\beta$ when approximating \mathcal{V} . Now given $x \in \mathcal{I}_i = [\xi_i, \xi_{i+1}]$, for $i > 0$, we have that ,for any $a \in [x_1, x]$, it holds

$$\widehat{\mathcal{J}}_x(x) = \widehat{\mathcal{V}}(x) + \widehat{\mathcal{R}}_x(x) = \widehat{\mathcal{V}}(x) = \mathcal{V}(\xi_i) \geq \frac{\mathcal{V}(\xi_{i+1})}{2} \geq \frac{\mathcal{V}(x)}{2} \geq \frac{\mathcal{J}_a(x)}{2 + \sqrt{2}}.$$

If $x \in \mathcal{I}_0$ then a similar argument as the one for the lifting stage shows that $I_a(x) \leq \varepsilon\Delta^*/m$.

Now assume that $F_{k+1}(x)$ is achieved by some $a^* \leq x$ via $F_{k+1}(x) = F_k(a^*) + I_{a^*}(x)$. Since G_k is a monotone function, we have that $G_k(x) + \widehat{I}_x(x) \geq G_k(a^*) + \widehat{I}_x(x)$. Overall, combined the above results with the facts that $G_{k+1}(x) \geq \max\{G_k(a^*) + \widehat{I}_{a^*}(x), G_k(x) + \widehat{I}_x(x)\}$ and $G_k(x) \geq F_k(x)/(2 + \sqrt{2}) - k\varepsilon\Delta^*/m$, we conclude that $G_{k+1}(x) \geq F_{k+1}(x)/(2 + \sqrt{2}) - (k+1)\varepsilon\Delta^*/m$. Therefore by induction, we have that $F_i(x)/(2 + \sqrt{2}) - (i+1)\varepsilon\Delta^*/m \leq G_i(x) \leq F_i(x)$, for any $i \in [1, m]$. This implies that $G_m(1) \geq F_m(1)/(2 + \sqrt{2}) - \varepsilon\Delta^* = (1/(2 + \sqrt{2}) - \varepsilon)F_m(1)$. ■

By setting $\varepsilon = (\sqrt{2} - 1)^2/4$, the above lemma implies that $G_m(1)$ is a $1/4$ -approximation to $F_m(1)$, thus a $(1/4, \sqrt{2})$ -approximation to $\mathcal{D}_{\partial, \delta}(s, Q)$. In fact, the approximation factor can be improved to $(\frac{1}{1+\sqrt{2}}, \sqrt{2})$ -approximation without changing the asymptotic time complexity by refining our approximation scheme. For curves in \mathbb{R}^d , the same algorithm can be used to $(\frac{1}{1+\sqrt{d}}, \sqrt{d})$ -approximate $\mathcal{D}_{\partial, \delta}(s, Q)$ under the L_2 norm, as well as $(\frac{1}{1+\sqrt{d}}, d)$ -approximate $\mathcal{D}_{\partial, \delta}(s, Q)$ under any other L_p norm in the same asymptotic time and space complexity. Combining with Observation 5.1, we obtain the following result.

Theorem 5.9 *Given an edge s and a polygonal curve Q of size m in \mathbb{R}^d , one can compute, in $O(m \log^2 m)$ time, using $O(m \log m)$ space, a $(\frac{1}{1+\sqrt{d}}, \sqrt{d})$ -approximation to the optimal partial matching between s and Q under the L_2 norm (i.e., $\mathcal{D}_\partial(s, Q)$). Similarly, one can compute a $(\frac{1}{1+\sqrt{d}}, d)$ -approximation to $\mathcal{D}_\partial(s, Q)$ under any other L_p norm in the same time and space complexity.*

5.3.4 Approximation algorithm for the general case

Finally, we combine the above double-sided approximation algorithm with the same framework as in Section 4.1 and get the following result.

Theorem 5.10 *Given two polygonal curves P and Q in \mathbb{R}^d , of size n and m respectively, and an error threshold δ , a $(\frac{1}{3+3\sqrt{d}}, \sqrt{d})$ -approximation of $\mathcal{D}_\delta(P, Q)$ under the L_2 norm can be computed in $O((n+m)^2 \log^2(n+m))$ time and $O((n+m) \log(n+m))$ space. The same time and space complexity holds for an $(\frac{1}{3+3\sqrt{d}}, d)$ -approximation algorithm of $\mathcal{D}_\delta(P, Q)$ under any other L_p norm.*

6 Conclusions

In this paper, we studied the problem of computing partial Fréchet distance for two polygonal curves. We presented several approximation algorithms for this problem. We believe that our work provides an important first step in understanding the partial curve matching problem and its structure.

Compared to computing the Fréchet distance for the global curve matching problem, our algorithms are more involved. This may be because that the structure of the partial Fréchet distance problem appears to be inherently more complex. Indeed, after appropriate transformation, the Fréchet distance computation boils down to the decision problem of whether two points on a terrain can be connected by a curve that lies completely below a certain threshold (i.e, reachability test). The partial Fréchet distance problem, however, is not memoryless as reachability. Informally, to compute the optimal solution, it seems necessary to remember the score of the best path to arrive at each possible point. Thus, the dynamic programming required is more complicated.

Although our work has revealed several properties of the structure of an optimal partial curve matching, there is still a lot of ground for further research. In particular, can one find an exact algorithm to compute $\mathcal{D}_\delta(P, Q)$ for general curves that runs in polynomial time? Can we develop better approximation algorithms, such as near-quadratic time algorithm that approximates $\mathcal{D}_\delta(P, Q)$ in a single-sided manner? It seems that developing a sub-quadratic time approximation algorithm may be hard, as currently, no such algorithm exists for the easier problem of computing the (global) Fréchet distance.

References

- [AB05] H. Alt and M. Buchin. Semi-computability of the Fréchet distance between surfaces. In *Proc. 21st Euro. Workshop on Comput. Geom.*, 2005.
- [AERW03] H. Alt, A. Efrat, G. Rote, and C. Wenk. Matching planar maps. *J. Algorithms*, 49:262–283, 2003.
- [AG95] H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *Internat. J. Comput. Geom. Appl.*, 5:75–91, 1995.

- [AG00] H. Alt and L. J. Guibas. Discrete geometric shapes: Matching, interpolation, and approximation. In Jörg-Rüdiger Sack and Jorge Urrutia, editors, *Handbook of Computational Geometry*, pages 121–153. Elsevier Science Publishers B. V. North-Holland, Amsterdam, 2000.
- [AHK⁺06] B. Aronov, S. Har-Peled, C. Knauer, Y. Wang, and C. Wenk. Fréchet distance for curves, Revisited. In *Proc. 14th Annu. European Sympos. Algorithms*, pages 52–63, 2006.
- [AHMW05] P. K. Agarwal, S. Har-Peled, N. Mustafa, and Y. Wang. Near-linear time approximation algorithms for curve simplification in two and three dimensions. *Algorithmica*, 42:203–219, 2005.
- [AKW01] H. Alt, C. Knauer, and C. Wenk. Matching polygonal curves with respect to the Fréchet distance. In *Proc. 18th Internat. Sympos. Theoret. Asp. Comp. Sci.*, pages 63–74, 2001.
- [AKW04] H. Alt, C. Knauer, and C. Wenk. Comparison of distance measures for planar curves. *Algorithmica*, 38(1):45–58, 2004.
- [BBW06] K. Buchin, M. Buchin, and C. Wenk. Computing the Fréchet distance between simple polygons in polynomial time. In *Proc. 22nd Annu. ACM Sympos. Comput. Geom.*, pages 80–87, 2006.
- [BPSW05] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On map-matching vehicle tracking data. In *Proc. 31st VLDB Conference*, pages 853–864, 2005.
- [BWF⁺00] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. H. Shinkdyalov, and P. E. Bourne. The protein data bank. *Nucleic Acid Res.*, 28:235–242, 2000.
- [CM05] M. Clausen and A. Mosig. Approximately matching polygonal curves with respect to the Fréchet distance. *Comput. Geom. Theory Appl.*, 30:113–127, 2005.
- [EHG⁺02] A. Efrat, S. Har-Peled, L. J. Guibas, J. S.B. Mitchell, and T.M. Murali. New similarity measures between polylines with applications to morphing and polygon sweeping. *Discrete Comput. Geom.*, 28:535–569, 2002.
- [HW06] S. Har-Peled and Y. Wang. Partial curve matching under the Fréchet distance. Manuscript. Available from http://www.uiuc.edu/~sariel/papers/06/part_f, 2006.
- [Ind02] P. Indyk. Approximate nearest neighbor algorithms for Fréchet distance via product metrics. In *Proc. 18th Annu. ACM Sympos. Comput. Geom.*, pages 102–106, 2002.

- [KHM⁺98] S. Kwong, Q. H. He, K. F. Man, K. S. Tang, and C. W. Chau. Parallel genetic-based hybrid pattern matching algorithm for isolated word recognition. *Int. J. Pattern Recognition & Artificial Intelligence*, 12(5):573–594, August 1998.
- [KKS05] M.S. Kim, S.W. Kim, and M. Shin. Optimization of subsequence matching under time warping in time-series databases. In *Proc. ACM symp. Applied comput.*, pages 581–586, New York, NY, USA, 2005.
- [KP99] E. J. Keogh and M. J. Pazzani. Scaling up dynamic time warping to massive dataset. In *Proc. of the Third Euro. Conf. Princip. Data Mining and Know. Disc.*, pages 1–11, 1999.
- [PP90] M. Parizeau and R. Plamondon. A comparative analysis of regional correlation, dynamic time warping, and skeletal tree matching for signature verification. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(7):710–717, 1990.
- [Rot05] G. Rote. Computing the Fréchet distance between piecewise smooth curves. Technical Report ECG-TR-241108-01, Freie Universität, Berlin, May 2005. To appear in *Comput. Geom. Theory Appl.*
- [Wen02] C. Wenk. *Shape Matching in Higher Dimensions*. PhD thesis, Dept. of Comput. Sci., Freie Universität, Berlin, 2002.