Partial Curve Matching under the Fréchet Distance*

Sariel Har-Peled† Yusu Wang‡

Abstract

In this paper, we study the fundamental problem of measuring the ***partial*** similarity between curves. In particular, given two curves P and Q, we wish to maximize the total length of portions of them that are *close* to each other, where the *closeness* is measured by the Fréchet distance, a common *global* distance measure for curves. The resulting maximal length is called the ***partial Fréchet similarity*** between P and Q, and we present various algorithms to compute it. Specifically, under the L_1 or L_∞ metrics, we present an algorithm that computes, in $O(m^2)$ time, the partial Fréchet similarity 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 similarity 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 similarity. To the best of our knowledge these are the first results on this problem.

*Work on this paper by Sariel Har-Peled was partially supported by an NSF CAREER award CCR-0132901.

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

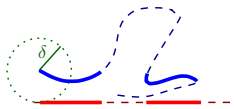
‡Dept. of Comp. Sci. and Engineering, *yusu@cse.ohio-state.edu*.

1 Introduction

Measuring similarity between curves is a fundamental problem that appears in many fields, including computer graphics, pattern recognition, geographic information system, and structural biology.

A natural measure of similarity between two curves is the ***Fréchet distance***. Intuitively, imagine that a dog and its handler are walking on their respective curves with a leash between them. Both can control their speed, but they can only go forward. The Fréchet distance of these two curves is the minimal length of a 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 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]. Furthermore, it is a *continuous* measure, while most other curve similarity measures, such as the widely used root-mean-square-deviation (RMSD) and the dynamic time warping (DTW), are *discrete* measures, considering only vertices of input curves.

The Fréchet distance and its variants have been widely used in many applications [KP99, KKS05, KHM+98, PP90]. Alt and Godau [AG95] presented an algorithms 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 there are outliers and/or when the two curves share only partial similarity. For example, two protein structures (modeled as their backbone curves) may be similar around important functional sites, but are dissimilar in other places. An interesting recent work by Efrat et al. [EFV07] combined time warping to compute an integral (summed) version of the Fréchet distance, which can "smooth out" the impact of some outliers. However, it does not fully resolve the issue of partial similarity, especially when significant parts of the curves are dissimilar. To this end, we investigate the ***partial curve matching problem***, where given two curves and a threshold δ, we wish to find the best matching such that the largest possible fraction of the two curves are matched within (Fréchet) distance δ.

The partial matching problem for point sets has been studied extensively, mainly under the Hausdorff distance [CS98, IMV99]. Currently, in practice, the partial curve matching problem under the discrete Fréchet distance or the RMS distance is usually computed, where only the vertices of the input polygonal curves are considered. However, it is well known that these discrete measures may fail to capture the "real" distance between input curves (see figure on the right). In this paper, we aim to develop a *continuous* partial similarity measure.

Other related work includes minimizing the Fréchet distance under various classes of transformations [AKW01, CM05, Wen02]. However the running times are 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. Computing the partial Fréchet similarity also identifies similar and dissimilar portions between curves, which is perhaps more interesting than simply the similarity score itself. To the best of our knowledge, this is the first paper to study the continuous partial curve matching problem.

More specifically, given a distance threshold δ, let $\mathcal{D}_{\partial,\delta}(P,Q)$ denote the optimal partial Fréchet similarity between two polygonal curves P and Q, which is the total length of the longest subcurves of P and Q that are matched with Fréchet distance at most δ. (Note that the Fréchet distance can be measured under any L_p norm.) In Section 3, for the case where P is a single edge, we compute $\mathcal{D}_{\partial,\delta}(P,Q)$ in $O(m^2)$ time, under the L_1 or L_∞ norms, where Q is a polygonal curve with m edges.

Unfortunately, the exact computation of $\mathcal{D}_{\partial,\delta}(P,Q)$ seems to be challenging in general, and it is unclear whether a polynomial time algorithm exists. We thus focus on efficient approximation algorithms for $\mathcal{D}_{\partial,\delta}(P,Q)$ for general curves. Specifically, in Section 4, given two polygonal curves P and Q of size n and m, respectively, we present two approximation algorithms for $\mathcal{D}_{\partial,\delta}(P,Q)$, that roughly run in $O((n+m)^3)$ time under the L_1 and L_∞ norms, and in $O((n+m)^4)$ time under any other L_p norm. The first algorithm exploits a general framework combined with the exact algorithm for the special case, and the framework is also used later in Section 5.

In Section 5, we present a more efficient 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 for any L_p norm. We remark that even for the much simpler matching problems such as computing the discrete Fréchet distance or the RMS distance, currently there is no approximation algorithm that runs in sub-quadratic time for general curves[①].

We remark that compared to the original Fréchet distance for the *global* curve matching problem, computing the partial Fréchet similarity appears to be significantly harder. Some discussion as well as open problems are presented in Section 6. Finally, details omitted from this extended abstract due to space limitations can be found in the Appendix.

2 Preliminaries

Problem definition. A parameterized curve in $\mathbb{R}^d$ can be represented as a function $\mathsf{f}\colon [0,1] \to \mathbb{R}^d$. A (monotone) ***reparametrization*** α is a continuous non-decreasing function $\alpha\colon [0,1] \to [0,1]$ 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, where the point $\mathsf{f}(\alpha(x))$ is matched to the point $\mathsf{g}(\beta(x))$, for any $x\in[0,1]$. Given two curves $\mathsf{f},\mathsf{g}:[0,1]\to\mathbb{R}^d$, the ***Fréchet distance*** between them under the L_p norm is defined as

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

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

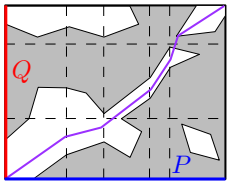
Given a distance threshold $\delta>0$ and a matching (α,β) of curves f and g, the ***score*** of (α,β) under the L_p norm, also referred to as the ***partial Fréchet similarity of*** f ***and*** g ***w.r.t*** (α,β) is defined as

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

[①] Although only single-sided approximation was considered for those problems, it seems that eventually our partial matching problem meets a similar technical difficulty as them (which is some variant of the string matching problem) even when double-sided approximation is allowed.

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 w.r.t. the matching (α, β). Naturally, we would like to maximize the length of these portions, and the ***partial Fréchet similarity between f and g*** for a threshold δ is defined as the maximum score of any matching of f and g; that is, $\mathcal{D}_{\partial,\delta}(\mathsf{f},\mathsf{g}) := \max_{\alpha,\beta} \mathcal{S}_{\alpha,\beta}(\mathsf{f},\mathsf{g})$, where α and β range over all monotone reparametrizations.

Free space diagram. For two polygonal curves $P = \langle p_1, \ldots, p_n \rangle$ and $Q = \langle q_1, \ldots, q_m \rangle$, an alternative way to view $\mathcal{D}_\partial(P,Q)$ is via the following ***free space diagram*** $M = M_\delta(P,Q)$ [AG95]: M is an n by m map where its ith column corresponds to the ith edge of P and has width $||p_i p_{i+1}||$, while its jth row corresponds to the jth 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 $\mathsf{f} : [0, \text{len}_2(P)] \to \mathbb{R}^d$ and $\mathsf{g} : [0, \text{len}_2(Q)] \to \mathbb{R}^d$ represent the arc-length parametrization of P and Q, respectively. Then every point (x, y) in M corresponds to a pair of points $\mathsf{f}(x) \in P$ and $\mathsf{g}(y) \in Q$. We abuse the notation slightly and use $P(x)$ and $Q(y)$ to represent $\mathsf{f}(x)$ and $\mathsf{g}(y)$, respectively, from now on.



We say that a point $(x, y) \in M$ is ***good*** if $\mathrm{d_p(P(x), Q(y))} \leq \delta$. By convexity of the L_p norm, the good points within any cell $M[i][j]$ of M form a closed and connected convex region. In particular, this ***good region*** is the intersection between an ellipse with the cell $M[i][j]$ under the L_2 norm. 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 $O(d)$ (resp. $O(2^d)$) under the L_∞ norm (resp. the L_1 norm).

Monotone paths. There is a one-to-one correspondence between all possible matchings of P and Q, and the set of monotone paths in M from its bottom-left corner to its top-right corner, where a ***monotone path*** in M is a path 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. Then $\mathcal{S}_\pi(P,Q)$, the partial similarity 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. To compute $\mathcal{D}_{\partial,\delta}(P,Q)$, the goal is to find an ***optimal monotone path*** whose score is maximized. This corresponds to an ***optimal partial matching*** between P and Q.

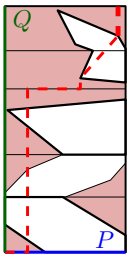
Observe that any two pieces of monotone paths sharing the same starting and ending points have the same L_1-norm length. Hence in general, the set of entry and exit points of a monotone path into the good regions completely decides its score. Now a monotone path π is ***conformal*** if (i) its intersection with bad regions in M consists of only vertical and horizontal segments, and (ii) 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 $\mathcal{D}_{\partial,\delta}(P,Q)$ if $\alpha\mathcal{D}_{\partial,\delta}(P,Q) \leq \tau \leq \mathcal{D}_{\partial,\delta}(P,Q)$; and τ is a ***double-sided (α,β)-approximation*** of $\mathcal{D}_{\partial,\delta}(P,Q)$ if $\alpha\mathcal{D}_{\partial,\delta}(P,Q) \leq \tau \leq \mathcal{D}_{\partial,\beta\delta}(P,Q)$.

3 Partial Fréchet Similarity of A Segment and 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. The lemma below suggests that we only need to consider critical paths to compute $\mathcal{D}_\partial(s, Q)$. The proof is in Appendix A.1.



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.*

Lemma 3.1 suggests a dynamic programming algorithm to compute $\mathcal{D}_\partial(s, Q)$. First, project the vertices of the good regions from all the cells onto each of the horizontal grid edges of M. This produces $N = O(m)$ number of ***Steiner points*** on every horizontal edge. Next, consider the jth cell $M[j]$ of M. Let $\mathsf{p}_j[i]$ denote the ith Steiner point along the jth horizontal edge of M, and $\mathcal{S}(i, j)$ denote the optimal score for any critical path from $\mathsf{p}_1[1]$ to $\mathsf{p}_j[i]$. Assume without loss of generality that $\mathsf{p}_j[i]$ is not in the good region (the other case is similar). It holds that

$$\mathcal{S}(i, j+1) = \max\{\ \mathcal{S}(i-1, j+1), \max_{1 \le k \le i} \{\mathcal{S}(k, j) + \mathrm{W}_{\mathrm{M[j]}}(\mathsf{p}_{\mathrm{j}}[\mathrm{k}], \mathsf{p}_{\mathrm{j+1}}[\mathrm{i}])\}\ \},$$

where $\mathrm{W_C(x, y)}$ is the score of the optimal path connecting x to y within a cell C and it is not defined it there is no monotone path from x to y. In other words, the optimal critical path to $\mathsf{p}_{j+1}[i]$ either arrives from the left, or from one of the Steiner points on the lower boundary edge of $M[j]$. Observe that $\mathrm{W_C(x, y)}$ is in fact the largest L_1-norm length of any segment within the intersection of the good region in C and the slab between l_x and l_y, where l_a denotes the vertical line passing through point a; $\mathrm{W_C(x, y)}$ is not defined if there there is no monotone path connecting x to y.

Consider a cell $M[j]$, since we only consider conformal paths, it is not hard to show that each $\mathcal{S}(i, j)$, for $1 \le i \le N$, can be computed in $O(N)$ time. In fact, by using an amortized analysis, we have the following result (proof in Appendix A.2).

Lemma 3.2 *Given $\mathcal{S}(1, j), \ldots, \mathcal{S}(N, j)$ for a fixed j, one can compute, in $O(N)$ time, the values of $\mathcal{S}(1, j+1), \ldots, \mathcal{S}(N, j+1)$, using $O(1)$ space.*

Since $N = O(m)$, by using dynamic programming together with Lemma 3.2, we can show the following.

Theorem 3.3 *Given $\delta > 0$, the partial Fréchet similarity 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 present two results on approximating $\mathcal{D}_\partial(P, Q)$ for two general polygonal curves P and Q. The details are relative standard, and can be found in Appendix B. The first one uses a general framework that converts some algorithm for the special case (where one input is a segment), exact or approximate, to an approximate algorithm for the general case. When combined with the

algorithm in Section 3, this framework produces the following result. The framework will also be used later in Section 5 to produce the faster double-sided approximation algorithm.

Theorem 4.1 *Given two polygonal curves P and Q of size n and m, respectively, and a distance threshold $\delta > 0$, 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.*

The second result is a $(1-\varepsilon)$-approximate algorithm for $\mathcal{D}_\partial(P,Q)$ under any L_p norm, by adding $O((n+m)^3/\varepsilon^2)$ number of *Steiner points* to the good regions in the free space diagram.

Theorem 4.2 *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 $\mathcal{D}_\partial(P,Q)$, 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.*

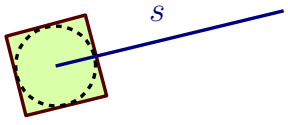
5 Double-sided Approximation of $\mathcal{D}_\partial(P,Q)$

In this section, we present a double-sided approximation algorithm for $\mathcal{D}_{\partial,\delta}(P,Q)$. The output of the algorithm is a number τ which is a $(\frac{1}{3+3\sqrt{d}}, \sqrt{d})$-approximation to $\mathcal{D}_{\partial,\delta}(P,Q)$ for two polygonal curves P and Q in $\mathbb{R}^d$ under the L_2 norm; namely, $\mathcal{D}_{\partial,\delta}(P,Q)/(3+3\sqrt{d}) \le \tau \le \mathcal{D}_{\partial,\sqrt{d}\delta}(P,Q)$. Using the same framework one can also compute a $(1/(3+3\sqrt{d}), d)$-approximation to $\mathcal{D}_\partial(P,Q)$ under any L_p norm. In what follows, for simplicity of exposition, we assume that P and Q are polygonal curves in the plane. 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. It turns out that combined with a similar framework as in the proof of Theorem 4.1, the $(\frac{1}{1+\sqrt{2}}, \sqrt{2})$-approximation algorithm for the special case leads to a $(\frac{1}{3+3\sqrt{2}}, \sqrt{2})$-approximation algorithm for the general case. Details omitted in this section due to lack of space can be found in Appendix C.

5.1 Algorithm outline

Given a segment s and a polygonal curve $Q = \langle q_1, q_2, \ldots, q_m \rangle$ in the plane, the goal is to compute a $(\frac{1}{1+\sqrt{2}}, \sqrt{2})$-approximation of $\mathcal{D}_{\partial,\delta}(s,Q)$ under the L_2 norm. Let $F_{\delta,k} : s \to \mathbb{R}$ be the function defined as $F_{\delta,k}(x) = \mathcal{D}_{\partial,\delta}(s[0,x], Q_k)$, where $s[0,x]$ is the subsegment of s from $s(0)$ to $s(x)$ and $Q_k = \langle q_1, \ldots, q_k \rangle$ 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 $F_m(1) = \mathcal{D}_\partial(s[0,1], Q_m) = \mathcal{D}_\partial(s,Q)$.

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 (see the solid square in the right figure). Note that if we rotate s so that it is parallel to the x-axis, then ψ coincides with the unit ball 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, as for a general segment, we are using

a ***rotated*** unit square to approximate the unit disk. Furthermore, for any $x \in s = [0,1]$, we now redefine $F_k(x)$ to be the optimal partial Fréchet similarity between the segment $[0,x]$ and the chain Q_k under the L_∞ norm. Since the unit square always contains a unit disk and is always contained inside a disk of radius $\sqrt{2}$ (and $\sqrt{d}$ in $\mathbb{R}^d$), we have the following.

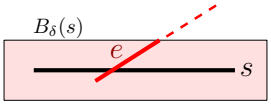
Claim 5.1 *Given an arbitrary segment $\hat{s}$ and a polyline $\hat{Q}$ of size m in $\mathbb{R}^d$, let μ be the affine transformation that transform $\hat{s}$ to $s = [0,1]$, and let $Q = \mu(\hat{Q})$. Then, $F_m(1) = \mathcal{D}_{\partial,\delta}(s,Q)$ under the L_∞ norm is a $(1,\sqrt{d})$-approximation for $\mathcal{D}_{\partial,\delta}(\hat{s},\hat{Q})$ under the L_2 norm. Furthermore, an α-approximation for $F_m(1)$ induces an $(\alpha,\sqrt{d})$-approximation of $\mathcal{D}_{\partial,\delta}(\hat{s},\hat{Q})$.*

$F_m(1)$ is also a $(1,d)$-approximation for $\mathcal{D}_{\partial,\delta}(\hat{s},\hat{Q})$ under any other L_p norm.

Let $\hat{s}, \hat{Q}, s$, and Q be as defined in the claim above. The detailed structure of F_k for s and Q will be described in Section 5.2. To further improve the efficiency of our algorithm, we introduce a *second level of approximation* in Section 5.3: in each round, instead of F_k, we maintain a simpler function G_k that approximates F_k that is faster to maintain. In the end, $G_m(1)$ provides a $\frac{1}{1+\sqrt{d}}$-approximation for $F_m(1)$, which, combined with Claim 5.1, implies that it is a $(\frac{1}{1+\sqrt{d}},\sqrt{d})$-approximation to the optimal partial Fréchet similarity between the original arbitrarily oriented segment $\hat{s}$ and the polygonal curve $\hat{Q}$ under the L_2 norm, i.e, $\mathcal{D}_{\partial,\delta}(\hat{s},\hat{Q})/(1+\sqrt{d}) \le G_m(1) \le \mathcal{D}_{\partial,\sqrt{d}\delta}(\hat{s},\hat{Q})$. From now on, we will only focus on the segment $s = [0,1]$ and the polygonal curve Q.

5.2 The structure and computation of the F_ks

We describe the structure of F_ks 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 $Q_k = \langle q_1, \ldots, q_{k-1}\rangle$.



Let $B_\delta(X) = \{y \in \mathbb{R}^d \mid \exists x \in X \text{ s.t. } d_\infty(x,y) \le \delta\}$ denote the ***δ-neighborhood*** of an 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.

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 1, and the slope of the diagonal sides of this parallelogram is $\sqrt{1+\rho(e)^2}$, where $\rho(e)$ is the slope of e. Now let $\mathcal{I}_a(x)$ denote the score of the best matching between the edge e and the subsegment $[a,x] \subseteq s$. Geometrically, the function $\mathcal{I}_a(x)$ is the maximum L_1-norm length of any 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 $\mathcal{I}_a(x)$ may change its behavior (the values x_1, x_2, x_3 and x_4 as depicted in Figure 1). For simplicity of exposition, we assume from now on that $x_2 < x_3$, unless otherwise specified.

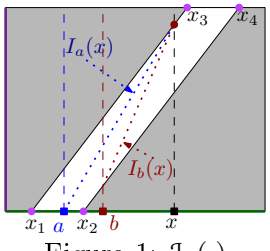


Figure 1: $\mathcal{I}_a(\cdot)$

Let $\mathcal{V}(a) = \mathcal{I}_a(a)$ be 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 (see Figure 2): $\mathcal{V}(a) = \sqrt{1+\rho(e)^2}(a - x_1)$ for $a \in [x_1, x_2]$, $\mathcal{V}(a) = \mathcal{V}(x_2)$ for $a \in (x_2, x_3]$, and $\mathcal{V}(a) = \mathcal{V}(x_2) - \sqrt{1+\rho(e)^2}(a - x_3)$ if $a \in (x_3, x_4]$.

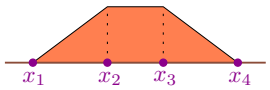


Figure 2: $\mathcal{V}(\cdot)$

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 1). As $\mathcal{I}_a(x)$ is the L_1-norm length of this segment, we have that for $a \in [x_1, x_3]$,

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

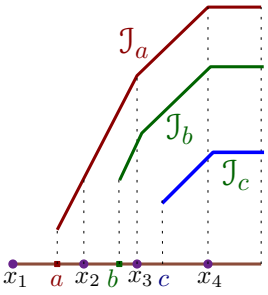


Figure 3: $\mathcal{I}_a$, $\mathcal{I}_b$, $\mathcal{I}_c$.

where $\rho_I = 1 + \sqrt{1 + \rho(e)^2}$ is a constant that only depends on e. The case for $a \in [x_3, x_4]$ is similar and simpler. See Figure 3. Now we rewrite $\mathcal{I}_a(x)$ as $\mathcal{I}_a(x) = \mathcal{V}(a) + \mathcal{R}_a(x)$ to simplify later expositions; geometrically, $\mathcal{R}_a(x)$ is the L_1-norm length of the segment connecting the ***top*** feasible point on ℓ_a to the ***top*** feasible point on ℓ_x.

Observation 5.2 *(i) For any a, $\mathcal{R}_a(a) = 0$, and for any $x \geq b \geq a$, 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 $\mathcal{R}_a(x) \geq x - a$.*

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

$$F_{k+1}(x) = \begin{cases} F_k(x) & \text{if } x \in [0, x_1) \\ \max_{a \in [x_1, x]}\{F_k(a) + \mathcal{I}_a(x)\} & \text{if } x \in [x_1, x_4] \\ \max\{F_{k+1}(x_4), F_k(x)\} & \text{if } x \in (x_4, 1]. \end{cases} \quad (2)$$

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}. It is easy to verify that F_k is a monotone piecewise linear (PL) function. Let N be the descriptive complexity of F_k. Based on Eqn (2), below we show that F_{k+1} has complexity $N + O(1)$, and can be computed in $O(N)$ time.

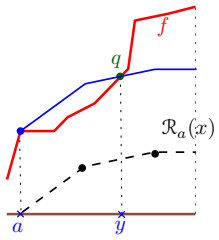
The cases for $x \in [0, x_1)$ and $x \in (x_4, 1]$ are straightforward. We consider only when $x \in [x_1, x_4]$. Here for every x, we wish to compute the set of function values $F_k(a) + \mathcal{I}_a(x)$ for all $x_1 \leq a \leq x$, and then take the largest value. This is equivalent to that for every $a \in [x_1, x_4]$, we compute the function $F_k(a) + \mathcal{I}_a(y)$ for $y \geq a$. Then for a fixed x, we simply take the largest value that such a function can generate — hence the graph of the function $F_{k+1}(\cdot)$ is the upper envelope of the graphs of functions $F_k(a) + \mathcal{I}_a(\cdot)$ for all $a \in [x_1, x_4]$.

We implement this goal in two steps. First, we compute the function $f : [x_1, x_4] \to \mathbb{R}$ defined as $f(a) = F_k(a) + \mathcal{V}(a)$; note that $F_k(a) + \mathcal{I}_a(x) = F_k(a) + \mathcal{V}(a) + \mathcal{R}_a(x) = f(a) + \mathcal{R}_a(x)$. 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 at most four new vertices (i.e., $x_1, \ldots, x_4$) to the graph of the function f. We call this step the ***lifting stage***.

The second step is called the ***enveloping stage***, aiming at computing $F_{k+1}(x) = \max_a(f(a) + \mathcal{R}_a(x))$ for any x. By Observation 5.2 we have the following claim.

Claim 5.3 *If $f(a) + \mathcal{R}_a(x) = f(b) + \mathcal{R}_b(x)$ for some $a \leq b$ and x, then $f(a) + \mathcal{R}_a(b) = f(b)$. Furthermore, if $f(a) + \mathcal{R}_a(b) \geq f(b)$, then $f(a) + \mathcal{R}_a(x) \geq f(b) + \mathcal{R}_b(x)$.*

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 $\mathcal{R}_a$ to the graph of f at the point $(a, f(a))$. See the right figure. 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, $f(b) + \mathcal{R}_b(x) \leq f(a) + \mathcal{R}_a(x)$. Since $F_{k+1}(x) = \max_{a \in [x_1, x_4]}\{F_k(a) + \mathcal{I}_a(x)\} = \max_{a \in [x_1, x_4]}\{f(a) + \mathcal{R}_a(x)\}$, there is no need to consider $f(b) + \mathcal{R}_b(x)$ for such $b \in (a, y)$ when computing $F_{k+1}(x)$ (as their graph will not appear on the upper envelope). Thus once we have attached the graph $\mathcal{R}_a$ at $(a, f(a))$, we only need to start from $(y, f(y))$ to find the next position to attach the $\mathcal{R}$ graph. This leads to an algorithm to compute $F_{k+1}(x)$ by sweeping the graph of $f(x)$ from left to right only ***once*** in $O(N)$ time, and the resulting function $F_{k+1}(x)$ has descriptive complexity $N + O(1)$. See Appendix C.1 for details.



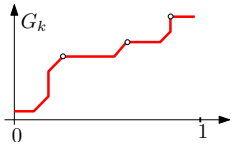
In summary, our algorithm starts with $F_0 = 0$ (i.e, $k = 0$). It then runs in m rounds, updating F_k to F_{k+1} in $O(N_k)$ time in each round, where N_k is the complexity of F_k. Since in each round, the complexity of F_{k+1} increases by at most $O(1)$, F_m is a monotone PL-function of complexity $O(m)$, and can be computed in $O(m^2)$ time. Combined with Claim 5.1, we conclude that:

Lemma 5.4 *Given an arbitrary segment $\hat{s}$ and a polygonal curve $\hat{Q}$ of size m, 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 $\hat{s}$ and $\hat{Q}$ under the L_2 norm, that is, $\mathcal{D}_{\partial,\delta}(\hat{s}, \hat{Q}) \leq \tau \leq \mathcal{D}_{\partial,\sqrt{2}\delta}(\hat{s}, \hat{Q})$.*

5.3 Approximation algorithm for $F_m(1)$

In this section, we describe an efficient near-linear time algorithm to $1/(1+\sqrt{2})$-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 a simpler *monotone PL* function G_k, so that for any $x \in s = [0, 1]$, we have $F_k(x)/\gamma - k\xi\Delta^*/m \leq G_k(x) \leq F_k(x)$, where $\Delta^* = F_m(1)$ and γ is a constant that will be specified shortly. In the end, $G_m(1)$ provides the required approximation for $F_m(1)$.

In particular, each G_k has the ***tridirectional property***; that is, each segment in its graph is either horizontal, vertical or with slope 1. Vertices of the graph of such a PL-function are called ***break points***, and those connecting a diagonal or a vertical edge with a horizontal one are ***base vertices*** (empty dots in the right figure). Given a PL function g, the ***base query*** for a real number x asks for the first base vertex on the graph of g 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 case of F_ks). To achieve this goal, we first need a better data structure for representing a PL-function (details in Appendix C.2).



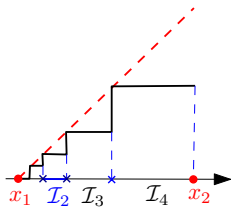
Lemma 5.5 *Given a PL-function $H : [l, r] :\to \mathbb{R}$ of complexity N, one can build a data-structure to represent H so that the following operations are supported: (i) insert/delete a consecutive sequence of k break points, (ii) evaluate $H(x)$ for any $x \in [l, r]$, (iii) base query if H is tridirectional, and (iv) add H with a constant over an interval $[a, a'] \subset [l, r]$. The first operation can be performed in $O(k + \log N)$ time, and the remaining operations can be performed in $O(\log N)$ time.*

5.3.1 Updating the G_ks

Given the function G_k, of complexity N, represented by an instance T of the data-structure of Lemma 5.5, we wish to update T into representing G_{k+1}. Again, let $x_1, \ldots, x_4$ be the projection

of the vertices of the white region onto s (recall Figure 1), and we describe only the case when $x \in [x_1, x_4]$. Recall that $F_{k+1}(x) = \max_{a \in [x_1, x]}\{F_k(a) + \mathcal{I}_a(x)\}$ for $x \in [x_1, x_4]$, 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.

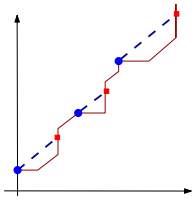
The Lifting stage. Previously, we first lift $F_k(x)$ to $F_k + \mathcal{V}(x)$ for every $x \in [x_1, x_4]$. Here, we approximate $\mathcal{V}$ (recall Figure 2) by the function $\widehat{\mathcal{V}}$ specified below.



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, where we use the solid stair to approximate the dashed linear function): First, let $L = x_2 - x_1$, and ξ a constant to be specified later. Subdivide $[x_1, x_2]$ into $\beta = O(\log(m/\xi))$ intervals, with the ith interval $\mathcal{I}_i = [x_1 + L/2^{\beta-i}, x_1 + L/2^{\beta-i-1}]$, for $i = 1, \ldots, \beta-1$. Also, set $\mathcal{I}_0 = [x_1, x_1 + L/2^{\beta-1}]$. For $x \in \mathcal{I}_i$, we set $\widehat{\mathcal{V}}(x) = \min_{y \in \mathcal{I}_i} \mathcal{V}(y) = \mathcal{V}(x_1 + L/2^{\beta-i})$ for $i > 0$ and $\widehat{\mathcal{V}}(x) = \mathcal{V}(x_1) = 0$ for $i = 0$. 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 a small *additive* error when $x \in \mathcal{I}_0$. In particular, $\mathcal{V}(x)/2 - \xi\Delta^*/m \leq \widehat{\mathcal{V}}(x) \leq \mathcal{V}(x)$.

The lifting stage now consists of $O(\log(m/\xi))$ modifications of G_k by adding to it the constants defined above on the intervals $\mathcal{I}_0, \ldots, \mathcal{I}_\beta$, respectively. Using the data-structure of Lemma 5.5, each such update takes $O(\log N)$ time. Overall, the lifting stage takes $O(\log(N)\log(m/\xi))$ time, and introduces $O(\beta) = O(\log(m/\xi))$ new break points. The resulting PL-function $H(\cdot)$ still satisfies the tridirectional property, and has complexity $N + O(\log(m/\xi))$.

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 $(a, H(a))$ now looks like shooting a ray of slope 1, referred to as a *diagonal ray*, from the point $(a, H(a))$; and G_{k+1} is the upper envelope of all such rays. It follows from Observation 5.2 (iii) that $\widehat{\mathcal{R}}_a(x) \leq \mathcal{R}_a(x)$. Furthermore, Claim 5.3 still holds for $\widehat{\mathcal{R}}_a$.



Observe that only diagonal rays attached at base vertices may potentially appear on the upper envelope. Now, 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 ray and the graph of H is necessarily on a vertical edge in the graph of H. By Claim 5.3, for any point $(x, H(x))$ with $w_1 < x < y_1$, 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 the right figure.

Easy to verify that the resulting function G_{k+1} is still tridirectional, and this procedure does not create any new break point. This fact, together with a global analysis, implies that the enveloping stage takes $O(\log m \log \frac{m}{\xi})$ *amortized* time, and overall, we have: (proof in Appendix C.3).

Claim 5.6 *It takes $O(m \log m \log \frac{m}{\xi})$ time to compute G_m.*

The approximation quality of G_k is summarized in the following lemma, and details can be found in Appendix C.4.

Lemma 5.7 *For any $0 \leq k \leq m$, we have that $F_k(x)/\gamma - k\xi\Delta^*/m \leq G_k(x) \leq F_k(x)$, where $\gamma = 2+\sqrt{2}$ and $\Delta^* = F_m(1)$ is the optimal partial matching between segment $s = [0,1]$ and curve Q under the L_∞ metric.*

By setting $\xi = (\sqrt{2}-1)^2/4$, the above lemma implies that $G_m(1)$ is a 1/4-approximation to $F_m(1)$. In fact, the approximation factor can be improved to $\frac{1}{1+\sqrt{2}}$ without changing the asymptotic time complexity by refining our approximation scheme. The same algorithm can be extended to higher dimensions. Combining with Claim 5.1, we have the following result.

Theorem 5.8 *Given an arbitrary segment $\hat{s}$ and a polygonal curve $\hat{Q}$ 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 $\hat{s}$ and $\hat{Q}$ under the L_2 norm (i.e., $\mathcal{D}_\partial(\hat{s},\hat{Q})$). Similarly, one can compute a $(\frac{1}{1+\sqrt{d}}, d)$-approximation to $\mathcal{D}_\partial(\hat{s},\hat{Q})$ under any other L_p norm in the same time and space complexity.*

Finally, we combine the above double-sided approximation algorithm with a similar framework as the one used for the constant factor approximation algorithm in Theorem 4.1, and conclude with the following result.

Theorem 5.9 *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 $\mathcal{D}_\partial(P,Q)$ 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 $\mathcal{D}_\partial(P,Q)$ 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, the partial Fréchet similarity problem appears to be considerably harder. In some sense, the Fréchet distance can eventually be modeled as a reachability problem (with some constraints) in a planar region, while the partial Fréchet similarity problem asks for the path (under constraints) that maximize its intersection with this region. Finding the longest path is not memoryless as in a reachability test. Informally, to compute the optimal solution, it seems necessary to remember the score of the best path to arrive at each possible point. The partial Fréchet similarity problem bears similarity to the shortest path problem with obstacles. The difference is that instead of shortest path, it looks for longest path with some constraints, and the characterization of such paths is not well understood.

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 $\mathcal{D}_\partial(P,Q)$ for general curves that runs in polynomial time? Given any two points in the free space diagram, consider the sequence of boundary edges that the best monotone path π (i.e, with largest score) connecting them will pass through. Once this sequence is known, π can be computed using linear programming. We recently observed that only polynomial number of such sequences exist for all pairs of points. Is it possible to compute these polynomial number of distinct sequences in polynomial time? A positive answer to this question will imply that we

can compute the optimal partial Fréchet similarity by making a polynomial number of LP-queries. Finally, Can we develop better approximation algorithms, such as near-quadratic time algorithm that approximates $\mathcal{D}_\partial(P, Q)$ in a single-sided manner?

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.

[CM05] M. Clausen and A. Mosig. Approximately matching polygonal curves with respect to the Fréchet distance. *Comput. Geom. Theory Appl.*, 30:113–127, 2005.

[CS98] D. Cardoze and L. Schulman. Pattern matching for spatial point sets. In *Proc. 39th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 156–165, 1998.

[EFV07] A. Efrat, Q. Fan, and S. Venkatasubramanian. Curve matching, time warping, and light fields, new algorithms for computing similarity between curves. *J. Mathematic Imaging and Vision, To appear*, 2007.

[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.

[IMV99] P. Indyk, R. Motwani, and S. Venkatasubramanian. Geometric matching under noise: Combinatorial bounds and algorithms. In *Proc. 10th ACM-SIAM Sympos. Discrete Algorithms*, pages 457–465, 1999.

[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.

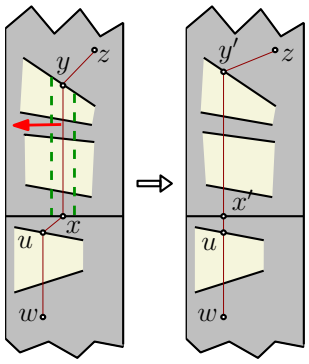
[PP90] M. Parizeau and R. Plamondon. A comparative analysis of regional correlation, dynamic time warping, and skeletal tree matching for signature verification. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(7):710–717, 1990.

[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.

A Proofs Omitted from Section 3

A.1 Proof of Lemma 3.1



(a) (b)

Figure 4: Shifting

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 4 (a)). 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 one such edge with slope ρ, and for simplicity, assume that it is not the first/last one. When e shifts horizontally by distance r, the change in the score induced by e' is $\mathrm{sign(e')}\rho\mathrm{r}$, where $\mathrm{sign(e')}$ is either $+1$ or -1, depending on whether e' is the upper or lower boundary edge of some good region. Note that this change is solely determined by e' and r and independent of the x-coordinate of the vertical edge e. Most importantly, the change becomes $-\mathrm{sign(e')}\rho\mathrm{r}$ if we shift e by distance $-r$ (i.e, by r in the opposite direction). Since $\Delta(r)$ is the summation of the changes induced by each boundary edge intersected by e, 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 (Figure 4 (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 4 (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 $\mathcal{S}_{\pi^*}(s, Q) \geq \mathcal{S}_{\pi}(s, Q)$.

A.2 Proof of Lemma 3.2

Given a cell $M[j]$, for simplicity, let U and B denote its upper and lower boundary edges, respectively; $U[i] = \mathsf{p}_{j+1}[i]$ and $B[i] = \mathsf{p}_j[i]$. 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 $U_1, \ldots, U_c$ and $B_1, \ldots, B_c$, respectively. We refer to such projected vertices as ***interval vertices***, to distinguish them from the set of Steiner points on each edge, although each interval vertex is also a Steiner vertex. Given an entry point $x \in B$ and an exit point $y \in U$, to find an optimal path path connecting x to y within $M[j]$, it turns out that only constant cases need to be inspected (a few examples are shown in Figure A.2), since we only consider conformal paths. In fact, we have the following claim, which can be proved by straightforward but tedious case analysis.

Claim A.1 *Given any pair of entry/exist points (x, y) with $x \in B$ and $y \in U$, the optimal score* $\mathrm{W(x, y)}$ *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*

$$\mathrm{W(x, y)} - \mathrm{W(u, y)} = \mathrm{W(x, v)} - \mathrm{W(u, v)},$$

if all four terms are defined.

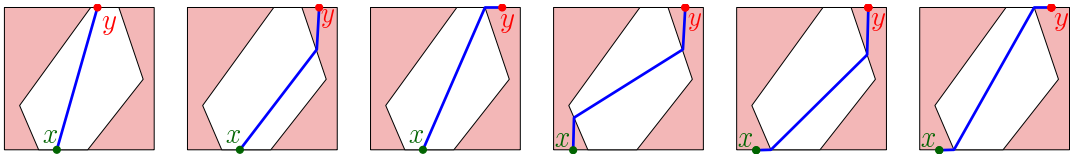


Figure 5: Entry point x and exit point y decide constant number of configurations for optimal critical path within a given cell, some examples are shown.

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

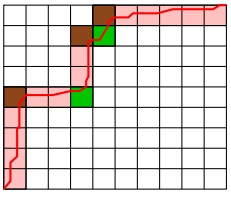
$$\mathcal{S}(i,j+1) = \max\{\mathcal{S}(i-1,j+1), \quad \max_{1\le l\le c} \mathrm{T(I_i[l],i)}\ \}.$$

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 A.1, the difference $\mathrm{T(k,i+1)-T(k,i)}$ remains the same for all $k \in B_l$ and $k \le i$. 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 $1 \le l \le c$. 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 $1 \le i \le N$ is $O(cN) = O(N)$, and the space complexity is $O(c)$.

B Approximation Algorithms for General Curves

In this section, we outline the two algorithms corresponding to Theorems 4.1 and 4.2, to approximate $\mathcal{D}_\partial(P,Q)$ for two general 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 under 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 under the L_1 and L_∞ norms. This algorithm works for any L_p norm, but its running time then deteriorates to $O((n+m)^4/\varepsilon^2)$.

B.1 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 form a ***trail***. See the right figure. We subdivide the trail into three portions: (1) right-turning cells $\mathrm{R}(\pi)$, where the trail makes a right turn; (2) left-turning cells $\mathrm{L}(\pi)$, where the trail makes a left turn; and (3) remaining vertical/horizontal segments $\mathrm{HV}(\pi)$. 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 $\mathrm{HV}(\pi)$. The removed end-cells, i.e, the turning cells (darker cells in the right figure), will be added into sets $\mathrm{R}(\pi)$ and $\mathrm{L}(\pi)$ appropriately.

Let $\mathcal{S}(X,\pi)$ denote the score induced by the intersection of a path π and a collection of cells X, and let $\mathcal{S}(\pi) = \mathcal{S}(M,\pi)$. Now given an optimal conformal path π^* that realizes $\mathcal{D}_\partial(P,Q)$, by the pigeon hole principle, we have that

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

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

B.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 B.1 *For any monotone path π in M, no two cells in set $\mathrm{R}(\pi)$ shares the same column nor the same row of M.*

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

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

where $\mathrm{W^*_C}$ 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 $\mathrm{W^*_C}$ 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 $\mathrm{S_R(n,m)}$, as well as a path π_1 to realize it, can be computed in $O(nm)$ time and space. Observe that $\mathcal{S}(\pi_1) \geq \mathrm{S_R}(\pi_1) = \mathrm{S_R(n,m)} \geq \mathcal{S}(\mathrm{R}(\pi^*), \pi^*)$.

B.1.2 Computation of π_3

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

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

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 pre-compute $h(k,i,j)$ and $v(i,k,j)$ for all i,j,ks. 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 ith edge of P and the subchain of Q between its kth and jth 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 $\mathrm{S_{HV}(n,m)}$, as well as a path π_3 to realize it, in $O((n+m)^4)$ time and $O((n+m)^3)$ space. Note that $\mathcal{S}(\pi_3) \geq \mathrm{S_{HV}(n,m)} \geq \mathcal{S}(\mathrm{HV}(\pi^*), \pi^*)$.

Improving the time complexity. The time and space complexity for computing π_3 can be further improved. In particular, 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 $\mathrm{S_{HV}(i,j)}$ as defined before to $\mathrm{S_h(k,i,j)}$ and

$\mathrm{S_v(k,i,j)}$, where $\mathrm{S_h(k,i,j)}$ (resp., $\mathrm{S_v(k,i,j)}$) is the best $\mathrm{S_{HV}}$-score of any monotone path from $M[1][1]$ to the kth point on the (i,j)th horizontal grid edge (resp., the kth point on the (i,j)th vertical edge). Note that $\mathrm{S_h(N_1,n,m)}$ (or $\mathrm{S_v(N_2,n,m)}$)) gives $\mathrm{S_{HV}(n,m)}$ where N_1 (resp. N_2) is the number of Steiner points on the (n,m)th horizontal grid edge (resp. vertical grid edge).

Now given a path π, $\mathrm{HV}(\pi)$ 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 $\mathrm{HV}(\pi)$ 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 $\mathrm{S_{HV}}$-score. Hence we now only need to consider critical paths.

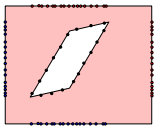
Consider the cell $M[i][j]$. Assume that the $\mathrm{S_v}$- and $\mathrm{S_h}$-scores have already been computed for the Steiner points on its left and bottom edges (i.e, the (i,j)th horizontal and vertical edges). We now wish to compute $\mathrm{S_h(k,i,j+1)}$ for all $1 \le k \le N$, where $N = O(m)$ is the number of Steiner points. (The computation for $\mathrm{S_v(k,i+1,j)}$s is symmetric.) To find the path with optimal $\mathrm{S_{HV}}$-score to the kth point on the $(i,j+1)$th horizontal edge, there are two possible cases: (i) cell $M[i][j]$ is a turning cell for this path, and (ii) it 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 as $\mathrm{S_h(k,i,j+1)}$. For case (ii), we inspect all potential entry points on the bottom cell boundary of $M[i][j]$, and return the one that induces the largest score. Overall, we have that

$$\mathrm{S_h(k,i,j+1)} = \max\{\max_{\mathrm{l}} \mathrm{S_v(l,i,j)} \ , \ \max_{1 \le \mathrm{l} \le \mathrm{k}} \{\mathrm{S_h(l,i,j)} + \mathrm{W_{M[i][j]}(l,k)}\ \},$$

where $\mathrm{W_{M[i][j]}(l,k)}$ is the best score of any path within the cell $M[i][j]$ that connects the lth point on the bottom boundary of cell $M[i][j]$, to the kth 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 the same method as in the proof of Lemma 3.2 to compute $\mathrm{S_h(k,i,j+1)}$s, for all $1 \le k \le N$, in $O(N) = O(m)$ time. Overall, the dynamic programming procedure takes $O((n+m)^3)$ time and $O((n+m)^2)$ space to compute π_3. Putting everything together, we conclude with Theorem 4.1.

B.2 $(1-\varepsilon)$-approximation

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 $\mathrm{W_C(x,y)}$, as defined earlier, is the best score of any path connecting x to y within cell C.

Observation B.2 *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,*

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

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 $\widehat{M}$. Since any monotone path in M passes through $O(n+m)$ number of cells, the following lemma follows easily from Observation B.2.

Lemma B.3 *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 programing. 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 3.2. Hence we conclude with Theorem 4.2.

C Proofs Omitted from Section 5

C.1 Time Complexity for Computing F_k

Lemma C.1 *The lifting and enveloping stages can compute $F_{k+1}(\cdot)$ in $O(N)$ time and the resulting function has complexity $N + O(1)$, where N is the complexity of F_k.*

Proof: First, recall that the lifting stage can be implemented in $O(N)$ time and the resulting function $f(x) = F_k(x) + \mathcal{V}(x)$ is a PL-function of complexity $N + O(1)$. We now focus on the enveloping stage, the goal of which is to compute $F_{k+1}(x) = \max_{a\in[x_1,x_4]}\{F_k(a) + \mathcal{I}_a(x)\}$ for $x \in [x_1, x_4]$.

Our algorithm sweeps the graph of $f(x)$ once 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 vertex $(w, f(w))$, then 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.

It is easy to verify that the resulting function is indeed the required $F_{k+1}(x)$. We now bound its descriptive complexity. First, observe that only constant 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 at most three vertices, and they are always at same coordinates for any $a \in [x_1, x_4]$ (i.e, x_1, x_3 and x_4, recall Figure 3). Since we only need to attach the graph $\mathcal{R}$ at vertices of f, the only other new vertices are those hitting points created. Consider the hitting vertex $v = (z, f(z))$ corresponding to the first intersection between $\mathcal{R}_w$ attached at $(w, f(w))$ and the graph of f. Note 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. ■

C.2 Proof of Lemma 5.5

Suppose that the PL-function H has $N+1$ break points $b_1 = l, b_2, \ldots, b_{N+1} = r$, with a linear function $h_i : [b_i, b_{i+1}] \to \mathbb{R}$ defined over the ith interval. 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 ith leaf of T, denoted by L_i, corresponds to interval $[b_i, b_{i+1}]$. Every leaf node L_i stores a linear function $f_{L_i} : [b_i, b_{i+1}] \to \mathbb{R}$, called the ***base function*** at leaf L_i. Each internal node $v \in T$ is associated with an interval $I_v = [b_{v_1}, b_{v_2+1}]$ where v_1 and v_2 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 $H_i = f_{L_i} + \sum_{v \in A_i} c_v$.

Clearly, for a PL function of complexity N, such a balanced binary tree representation has size $O(N)$, and supports each operation such as insertion/deletion, or computing H_i, in $O(\log N)$ time. Given any interval $[b_i, b_j]$, one can also identify $O(\log N)$ 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 μ over an interval $[a, a']$. First, we insert a and a' if they are not already break points of H. Next, we identify the set of nodes corresponding to the canonical decomposition of $[a, a']$, and increase the additive constant c_v associated with each such node $v \in T$ by μ. The entire process takes $O(\log N)$ time.

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

C.3 Proof of Claim 5.6

For the kth iteration, as described earlier, the lifting stage takes $O(\log N_k \log(N_k/\xi))$ time, where N_k is the complexity of G_k. The output is a tridirectional PL-function H of complexity $N_k + \log(N_k/\xi)$. We now focus on time complexity of the enveloping stage to update H to G_{k+1}. Let r_k be the number of new rays appearing in the upper envelope, 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 sweeping procedure described earlier updates H into G_{k+1}, as well as adjusting the corresponding data structure T, in $O(w_k + r_k \log N_k)$ time.

Observe that w_k and r_k can be $\Omega(N_k)$. However, each of the w_k vertices covered by the diagonal rays will be deleted from the list of break points forever. Since only the lifting stage can create new break points, it is easy to verify that overall there are most $O(m \log(m/\xi))$ break points ever created. Hence we have that $\sum_{k=1}^{m} w_k = O(m \log(m/\xi))$. 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. But for the left endpoint of this interval, it will never becomes a base vertex again. Since there are $O(m \log(m/\xi))$ break points ever created, we have that $\sum_{k=1}^{m} r_k = O(m \log(m/\xi))$ as well. Putting everything together, the total cost for the

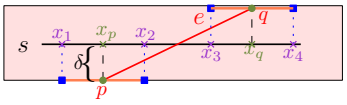
m rounds of updates to compute G_m from G_0 is

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

C.4 Proof of Lemma 5.7

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 $x\in[0,1]$, $F_k(x)/\gamma-k\xi\Delta^*/m\le G_k(x)\le F_k(x)$. We wish to show that $F_{k+1}(x)/\gamma-(k+1)\xi\Delta^*/m\le G_{k+1}(x)\le F_{k+1}(x)$. 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 $\mathcal{I}_a(x)=\mathcal{V}(a)+\rho_I(x-a)$ by $\widehat{\mathcal{I}}_a(x)=\widehat{\mathcal{V}}(a)+(x-a)$, where $\rho_I=1+\sqrt{1+\rho(e)^2}$ and $e=q_kq_{k+1}$ is the $(k+1)$th edge of Q. 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 earlier, $\widehat{\mathcal{V}}(x)\ge\mathcal{V}(x)/2-\xi\Delta^*/m$.



For the error induced by the enveloping stage, if $|\rho(e)|\le 1$, then we have that $\rho_I(x-a)\le(1+\sqrt{2})(x-a)$. Combined with the approximation factor from the lifting stage, this implies that, for any $a<x$, we have that $\widehat{\mathcal{I}}_a(x)\ge\mathcal{I}_a(x)/(1+\sqrt{2})$ for an ξ small enough (the ξ 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, than 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 2), 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 $\rho_I-1\ge\sqrt{2}$, this implies that

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

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 $\mathcal{I}_0,\mathcal{I}_1,\ldots,\mathcal{I}_\beta$ when approximating $\mathcal{V}$. Now given $x\in\mathcal{I}_i=[l,r]$, for $i>0$, we have that, for any $a\in[x_1,x]$, it holds

$$\widehat{\mathcal{I}}_x(x) \quad = \quad \widehat{\mathcal{V}}(x)+\widehat{\mathcal{R}}_x(x)=\widehat{\mathcal{V}}(x)=\mathcal{V}(l)\ge\frac{\mathcal{V}(r)}{2}\ge\frac{\mathcal{V}(x)}{2}\ge\frac{\mathcal{I}_a(x)}{2+\sqrt{2}}.$$

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

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