Locally correct Fréchet matchings

Computational Geometry 76 (2019) 1–18

Contents lists available at ScienceDirect

Computational Geometry: Theory and Applications

www.elsevier.com/locate/comgeo

Locally correct Fréchet matchings ☆

Kevin Buchin [a], Maike Buchin [b], Wouter Meulemans [a,*], Bettina Speckmann [a]

[a] TU Eindhoven, the Netherlands
[b] Ruhr Universität Bochum, Germany

ARTICLE INFO

Article history:
Received 31 August 2017
Accepted 16 August 2018
Available online 10 September 2018

Keywords:
Similarity
Fréchet distance
Matching
Local correctness

ABSTRACT

The Fréchet distance is a metric to compare two curves, which is based on monotone matchings between these curves. We call a matching that results in the Fréchet distance a Fréchet matching. There are often many different Fréchet matchings and not all of these capture the similarity between the curves well. We propose to restrict the set of Fréchet matchings to "natural" matchings and to this end introduce *locally correct* Fréchet matchings. We prove that at least one such matching exists for two polygonal curves and give an $O(N^3 \log N)$ algorithm to compute it, where N is the number of edges in both curves. We also present an $O(N^2)$ algorithm to compute a locally correct discrete Fréchet matching.

1. Introduction

Many problems ask for the comparison of two curves. Consequently, several distance measures have been proposed for the similarity of two curves P and Q, for example, the Hausdorff and the Fréchet distance. Such a distance measure simply returns a number indicating the (dis)similarity. The Hausdorff and the Fréchet distance are both based on a matching between the points on the curves. That is, each point on curve P is matched to a point on curve Q and vice versa. The distance returned is the maximum distance between any two matched points. The Fréchet distance uses *monotone matchings* (and limits of these): if point p on P and q on Q are matched, then any point on P after p must be matched to q or a point on Q after q. The *Fréchet distance* is the maximal distance between two matched points, minimized over all monotone matchings of the curves. Restricting to monotone matchings of only the vertices results in the *discrete Fréchet distance*. We call a matching resulting in the (discrete) Fréchet distance a *(discrete) Fréchet matching*. More details and exact definitions are given in Section 2.

There are often many different Fréchet matchings for two curves. However, as the Fréchet distance is determined only by the maximal distance, not all of these matchings capture the similarity between the curves equally well (see Fig. 1). There are applications that directly use a matching, for example, to map a GPS track to a street network [3] or to morph between the curves [4]. In such situations a "good" matching is important. Furthermore, we believe that many applications of the (discrete) Fréchet distance, such as protein alignment [5] and detecting patterns in movement data [6], would benefit from

☆ A preliminary version of this paper was presented at the European Symposium on Algorithms [1]. An extended version is included in W. Meulemans' PhD thesis [2, Chapter 6]. K. Buchin and B. Speckmann are supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 612.001.207 and project no. 639.023.208 respectively.

* Corresponding author.
E-mail addresses: k.a.buchin@tue.nl (K. Buchin), Maike.Buchin@ruhr-uni-bochum.de (M. Buchin), w.meulemans@tue.nl (W. Meulemans), b.speckmann@tue.nl (B. Speckmann).

https://doi.org/10.1016/j.comgeo.2018.09.002
0925-7721/

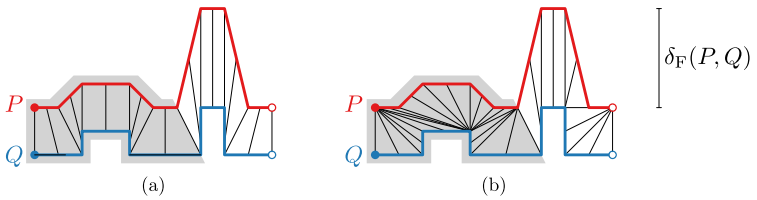

Fig. 1. Two Fréchet matchings for curves P and Q. Intuitively, matching (a) describes the similarity more accurately than matching (b), as the matched distances tend to be lower and the bends in the polyline coincide. If we accept that the Fréchet distance is a good method for quantifying similarity, then difference in matching quality becomes apparent in the shaded region: the submatching in (a) is a Fréchet matching, whereas the submatching in (b) is not. This motivates our definition of "locally correct" Fréchet matchings.

good Fréchet matchings. Indeed, Konzack et al. [7] use and investigate the quality of matchings in sports and gull movement data, considering matched distances for quality.

Contributions It seems counterintuitive that a Fréchet matching is not necessarily a Fréchet matching when considering two of its matched subcurves. Indeed, if we assume that the Fréchet distance is an accurate way of quantifying similarity, then it follows that good matchings to describe similarity remain a Fréchet matching when considering subcurves. We set out to investigate this phenomenon by restricting the set of Fréchet matchings to such "natural" matchings. We do so by introducing *locally correct* Fréchet matchings: matchings that, when restricted to any two matched subcurves, are still a Fréchet matching on these subcurves.

In Section 3 we prove that there exists such a locally correct Fréchet matching for any two polygonal curves. Based on this proof we describe in Section 4 an algorithm to compute such a matching in $O(N^3 \log N)$ time, where N is the total number of edges in both curves. We consider the discrete Fréchet distance in Section 5 and give an $O(N^2)$ algorithm to compute locally correct matchings under this metric.

Under some appropriate general position assumption (see also Section 2), the main ideas underlying the presented results can be distilled into simpler arguments. However, small changes to the points to ensure the general position assumption is met for a certain pair of curves may change Fréchet matchings. As such, we explicitly consider the degeneracy of the input curves and introduce terminology and results necessary to deal with these.

Related work The first algorithm to compute the Fréchet distance was given by Alt and Godau [8]; this algorithm runs in $O(N^2 \log N)$ time. They also consider the weak (or nonmonotone) Fréchet distance; it was later remarked by Har-Peled and Raichel that the algorithm for the weak Fréchet distance results in a "locally correct" weak Fréchet matching [9, Remark 3.5]. Eiter and Mannila gave the first algorithm to compute the discrete Fréchet distance, running in $O(N^2)$ time [10]. For years, both Alt and Godau's and Eiter and Mannila's algorithm have been the best known algorithms for their respective variants in the general case. Only recently, the first asymptotic improvements have been achieved for both variants: Buchin et al. [11] show how to compute the continuous Fréchet distance in $O(N^2\sqrt{\log N}(\log\log N)^{\frac{3}{2}})$ time; Agarwal et al. [12] present an $O(N^2 \log\log N/\log N)$-time algorithm for the discrete variant.

The Fréchet distance has received significant attention in a wide range of applications and variants beyond the three mentioned above. Here we focus on approaches that restrict the allowed matchings. Efrat et al. [4] introduced Fréchet-like metrics, the geodesic width and link width, to restrict to matchings suitable for curve morphing. Their method is suitable only for nonintersecting polylines. Moreover, geodesic width and link width do not resolve the problem illustrated in Fig. 1: both matchings also have minimal geodesic width and minimal link width. Maheshwari et al. [13] studied a restriction by "speed limits", which may exclude all Fréchet matchings and may cause undesirable effects near "outliers" (see Fig. 2). Buchin et al. [14] describe a framework for restricting Fréchet matchings, which they illustrate by restricting slope and path length. The former corresponds to speed limits; we briefly discuss the latter at the end of Section 4.

Since our introduction of locally correct Fréchet matchings in 2012 [1], new research has been performed in the area of Fréchet matchings and their quality. Rote [15] and Maheshwari et al. [16] consider what we refer to later in this paper as "locally optimal Fréchet matchings": that is, the locally correct Fréchet matching that reduces matched distances as quickly as possible. Both algorithms run in $O(N^3 \log N)$, but differ in how speed of traversal of the curves is quantified. A more detailed discussion regarding this extension is postponed until the end of Section 4.

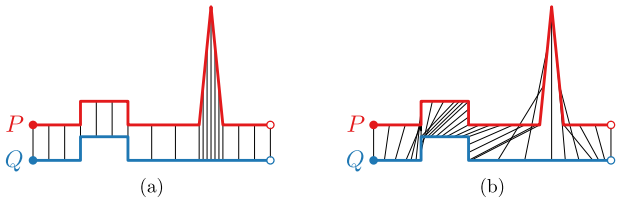

Fig. 2. Two Fréchet matchings. (a) A locally correct matching. (b) Imposing speed limits may yield a matching that is not locally correct.

Konzack et al. [7] compare Dynamic Time Warping, Edit Distance on Real Sequences and discrete locally correct Fréchet matchings. They conclude that Dynamic Time Warping and (discrete) locally correct Fréchet matchings work best, but that the latter are superior when data shows significant "delayed response", that is, when one trajectory follows another with some delay. Konzack et al. also observe that our algorithm for the discrete case works with any premetric,[1] but for simplicity of exposition, we shall restrict our attention to the Euclidean distance.

2. Preliminaries

Curves Let P be a polygonal curve with m edges, defined by $m+1$ vertices $p_0, \ldots, p_m$. We treat a curve as a continuous map $P : [0, m] \to \mathbb{R}^d$. In this map, $P(i)$ is equal to p_i for integer i. Furthermore, $P(i+\lambda)$ is a parametrization of the $(i+1)$st edge, that is, $P(i+\lambda) = (1-\lambda) \cdot p_i + \lambda \cdot p_{i+1}$, for integer i and $0 < \lambda < 1$. As a reparametrization $\sigma : [0, 1] \to [0, m]$ of a curve P, we allow any continuous, nondecreasing function such that $\sigma(0) = 0$ and $\sigma(1) = m$. We denote by $P_\sigma(t)$ the actual location according to reparametrization σ: $P_\sigma(t) = P(\sigma(t))$. By $P_\sigma[a, b]$ we denote the subcurve of P in between $P_\sigma(a)$ and $P_\sigma(b)$. In the following we are always given two polygonal curves P and Q, where Q is defined by its vertices $q_0, \ldots, q_n$ and is reparametrized by $\theta : [0, 1] \to [0, n]$. The reparametrized curve is denoted by Q_θ.

Fréchet matchings We are given two polygonal curves P and Q with m and n edges. A (monotone) *matching* μ between P and Q is a pair of reparametrizations (σ, θ), such that $P_\sigma(t)$ matches to $Q_\theta(t)$. The Euclidean distance between two matched points is denoted by $d_\mu(t) = \|P_\sigma(t) - Q_\theta(t)\|$. The maximum distance over a range is denoted by $d_\mu[a, b] = \max_{a \le t \le b} d_\mu(t)$. The *Fréchet distance* between two curves is defined as $\delta_{\mathrm{F}}(P, Q) = \inf_\mu d_\mu[0, 1]$. A *Fréchet matching* is a matching μ that realizes the Fréchet distance, that is, $\mu = \arg\inf_\mu d_\mu[0, 1]$ or, equivalently, $d_\mu[0, 1] = \delta_{\mathrm{F}}(P, Q)$.

Free-space diagrams Alt and Godau [8] describe an algorithm to compute the Fréchet distance based on the decision variant. This decision variant asks whether $\delta_{\mathrm{F}}(P, Q) \le \varepsilon$ for some given ε. We build on their work, using and extending their results to work with and compute locally correct Fréchet matchings. We briefly describe the important aspects here.

Alt and Godau's algorithm uses the *free-space diagram*, a two-dimensional diagram on the domain $[0, m] \times [0, n]$, capturing the parameter space of the two curves. Every point (x, y) in this diagram corresponds to the pair of points $P(x)$ and $Q(y)$; such a point is called *free* (colored white in figures) if and only if $\|P(x) - Q(y)\| \le \varepsilon$. The diagram has m columns and n rows; every cell $\langle c, r \rangle$ $(1 \le c \le m$ and $1 \le r \le n)$ corresponds to the edges $p_{c-1}p_c$ and $q_{r-1}q_r$. The free-space diagram for the two curves of Fig. 1 are illustrated in Fig. 3 for three values of ε. We use $\mathcal{F}_\varepsilon(P, Q)$ to denote the free-space diagram of two curves P and Q using ε to denote which points in the parameter space are free.

Each monotone matching between P and Q corresponds to an x- and y-monotone path from $(0, 0)$ to (m, n) in the free-space diagram. The Fréchet distance is at most ε, exactly when the free space of $\mathcal{F}_\varepsilon(P, Q)$ contains such a (bi)monotone path. In other words, $\delta_{\mathrm{F}}(P, Q) \le \varepsilon$ if and only if (m, n) is reachable from $(0, 0)$ through the free space with a monotone path. Alt and Godau [8] proved that the free space in a single cell is a convex region. This implies that the free space at the cell boundaries provide all necessary information to compute the Fréchet distance. Using the above, solving this decision problem can be done using a simple dynamic program over the free-space diagram.

Note that we scale every row and column in the diagram to correspond to the (relative) length of the actual edge of the curve instead of using unit squares for cells. The advantage of doing so is that the size of the diagram and matchings described by a monotone path therein are independent of sampling: adding extra vertices to P or Q that do not change their shape only lead to cells being split into smaller cells but otherwise do not change the free-space diagram.

To compute the Fréchet distance, one then finds the smallest ε such that there exists a monotone path from point $(0, 0)$ to (m, n) in free space. To do so, we need to check only certain *critical values* [8]. The critical values are defined by *critical events*. These events represent a structural change in the free space: new "passages" that may permit us to reach

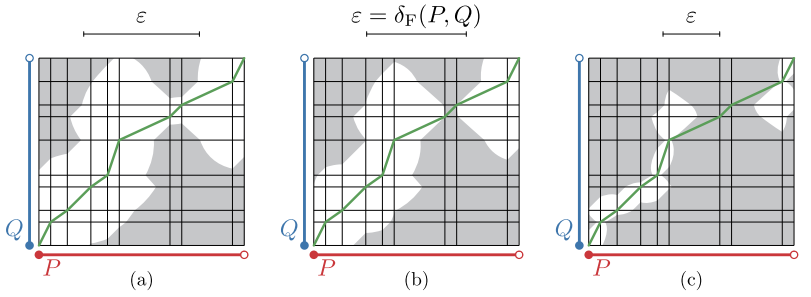

Fig. 3. Free-space diagrams for the curves of Fig. 1 with three different values of ε. The locally correct Fréchet matching is shown using a green path, crossing the diagram from bottom-left to top-right. (We refer the reader to the online article for colored versions of the figures in this article.)

[1] A nonnegative symmetric measure for which in particular the triangle inequality does not need to hold.

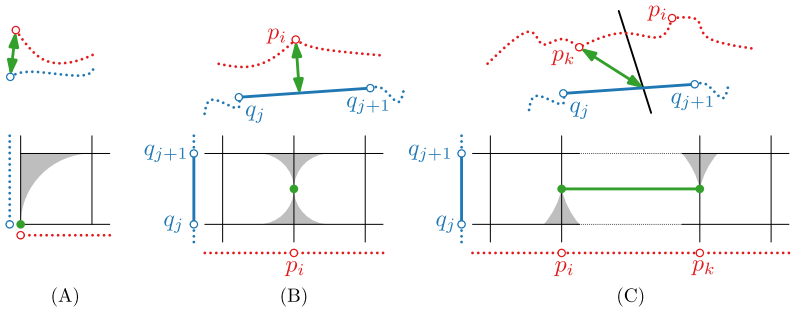

Fig. 4. Schematic illustration of the three critical events. Both the curves (top) and the free-space diagram (bottom) are illustrated. (A) Endpoints can be matched. (B) A passage between adjacent cells opens. (C) A passage opens between nonadjacent cells. The bisector is indicated with a black line.

(with a monotone path) new parts of the free space. To define the critical events, imagine continuously increasing ε from 0 to infinity. There are three types of critical events, which are defined as follows and illustrated in Fig. 4:

(A) The distance between p_0 and q_0 or between p_m and q_n is equal to ε. The former implies that we can start a path from $(0, 0)$, the latter that we can end one at (m, n).
(B) The minimal distance between vertex p_i and edge $q_j q_{j+1}$ (or between edge $p_i p_{i+1}$ and vertex q_j) is equal to ε. A passage opens between two horizontally (vertically) adjacent cells.
(C) The distance from two vertices p_i and p_k (with $i < k$) to the same point on edge $q_j q_{j+1}$ is equal to ε. In other words, the bisector of p_i and p_k intersects $q_j q_{j+1}$ and the distance to this intersection point is ε. A horizontal passage opens that spans multiple columns. Analogously, vertical passages are defined by two vertices of Q and an edge of P.

A critical value is a distance ε at which a critical event occurs. We call two events *concurrent* if they occur at the same critical value.

Degeneracy Concurrent events can be considered degenerate, as they require exactly the same distance to occur between different elements of the two given curves. We could thus aim for defining a general position assumption that states that there are no concurrent events. We note, however, that rotations do not alleviate the degeneracies and other transformations warp the distances, causing the potential Fréchet matchings to change.

We therefore avoid the use of such assumptions and instead explicitly deal with the concurrent events arising from degeneracy. For two given curves, we define the degeneracy K as the maximum number of concurrent events for any value of ε. The degeneracy K is naturally bounded by some cubic function, but as we show in Section 4 (page 10) in the worst case $\Theta(mn)$ of these can be relevant.

3. Locally correct Fréchet matchings

To distinguish between "good" and "bad" matchings, we introduce a new concept: *locally correct* Fréchet matchings. These require that the matching, restricted to any two matched subcurves, is again a Fréchet matching for these subcurves. This is formalized as follows.

Definition 1 *(Local correctness).* Given two polygonal curves P and Q, a matching $\mu = (\sigma, \theta)$ is *locally correct* if for all a, b with $0 \leq a \leq b \leq 1$

$$d_\mu[a, b] = \delta_F(P_\sigma[a, b], Q_\theta[a, b]).$$

As the Fréchet distance is defined via the minimum over all matchings, $d_\mu[a, b] \geq \delta_F(P_\sigma[a, b], Q_\theta[a, b])$ trivially holds for all matchings μ and $0 \leq a \leq b \leq 1$. However, equality is not necessarily true: not every Fréchet matching is locally correct, as illustrated in Fig. 1. The question arises whether a locally correct matching always exists and if so, how to compute it. In this section we resolve the first question positively, by proving the following theorem.

Theorem 1. *For any two polygonal curves P and Q, there exists a locally correct Fréchet matching.*

Overview We prove Theorem 1 by induction on the number of edges in the curves. The overall idea is that we determine for two curves a crucial part of their geometries: the vertices and edges corresponding to the critical event that occurs precisely at $\varepsilon = \delta_F(P, Q)$. We use this information to split both curves into smaller ones, such that we can apply the induction hypothesis. Finally, we must stitch together the locally correct Fréchet matchings for the smaller curves and show that this indeed yields a locally correct Fréchet matching.

In the following, m and n denote the number of edges of P and Q, respectively. We assume $m \leq n$ in formulating and proving these statements, but since the problem is symmetric, analogous statements hold where the roles of P and Q are reversed.

3.1. Base cases

First, we present the two base cases: one of the two curves is a point or both curves are line segments.

Observation 1. *For two polygonal curves P and Q with $m = 0$, a locally correct matching is (σ, θ), where $\sigma(t) = 0$ and $\theta(t) = t \cdot n$ for $0 \leq t \leq 1$.*

Proof. Since $m = 0$, P is just a single point, p_0. The Fréchet distance between a point and a curve is the maximal distance between the point and any point on the curve: $\delta_F(p_0, Q_\theta[a, b]) = d_\mu[a, b]$. This implies that the matching μ is locally correct. □

Observation 2. *For two polygonal curves P and Q with $m = n = 1$, a locally correct matching is (σ, θ), where $\sigma(t) = \theta(t) = t$ for $0 \leq t \leq 1$.*

Proof. The free-space diagram of P and Q is a single cell and thus the free space is a convex area for any value of ε. Since $\mu = (\sigma, \theta)$ is linear, we have that $d_\mu[a, b] = \max\{d_\mu(a), d_\mu(b)\}$: if there was a t with $a < t < b$ such that $d_\mu(t) > \max\{d_\mu(a), d_\mu(b)\}$, then the free space at $\varepsilon = \max\{d_\mu(a), d_\mu(b)\}$ would not be convex. Since $d_\mu[a, b] = \max\{d_\mu(a), d_\mu(b)\} \leq \delta_F(P_\sigma[a, b], Q_\theta[a, b]) \leq d_\mu[a, b]$, we conclude that μ is locally correct. □

3.2. Induction

Our induction hypothesis states that a locally correct Fréchet matching exists for any two curves P' and Q' with m' and n' edges, respectively, such that $m' + n' < m + n$. Hence, to apply the induction hypothesis, we reduce our two curves to two pairs of smaller curves. We do so by splitting them "on events". It is important to choose the right events for splitting. Moreover, splitting on type-A events (the corners of the free-space diagram) does not reduce the complexity of the resulting curves. Hence, the remainder is organized as follows:

- first, we introduce some more terminology and notation to describe "the right events";
- then, we show that type-A events indeed are of little consequence and can be largely ignored;
- subsequently, we describe in detail what it means to "split on an event";
- finally, we complete our proof by showing how these ingredients allow us to find a locally correct Fréchet matching.

Realizing events Already taking into consideration that type-A events are of little importance, we call a free-space diagram $\mathcal{F}_\varepsilon(P, Q)$ *feasible*, if a monotone path exists in the free space of $\mathcal{F}_\varepsilon(P, Q)$ from the top or right boundary of cell $\langle 1, 1\rangle$ to the bottom or left boundary of cell $\langle m, n\rangle$. A *realizing event* is a critical event (of type B or C) at the minimal value ε such that $\mathcal{F}_\varepsilon(P, Q)$ is feasible.

Consider a type-B realizing event between vertex p_i and edge $q_j q_{j+1}$: that is, the closest point q on $q_j q_{j+1}$ is at precisely distance ε from p_i; let λ be such that $q = (1 - \lambda)q_j + \lambda q_{j+1}$. We say that a monotone path in the free-space diagram *uses* this event if it passes through the point $(i, j + \lambda)$ (the green dot in Fig. 4(B)). In other words, for the corresponding matching $\mu = (\sigma, \theta)$, $P_\sigma(t) = p_i$ and $Q_\theta(t) = q$ for some t with $0 \leq t \leq 1$.

Similarly, consider a type-C realizing event between vertices p_i and p_k and edge $q_j q_{j+1}$: that is, the intersection q between the bisector of p_i and p_k and segment $q_j q_{j+1}$ is at precisely distance ε from p_i (and p_k); let λ be such that $q = (1 - \lambda)q_j + \lambda q_{j+1}$. We say that a monotone path in the free-space diagram *uses* this event if it passes through the points corresponding to the pairs $(i, j + \lambda)$ and $(k, j + \lambda)$ (the green dots in Fig. 4(B), and by monotonicity also the green line). In other words, for the corresponding matching $\mu = (\sigma, \theta)$, $P_\sigma(t) = p_i$, $P_\sigma(t') = p_k$ and $Q_\theta(t) = Q_\theta(t') = q$ for some t and t' with $0 \leq t \leq t' \leq 1$.

Let $\mathcal{E}$ denote the set of concurrent realizing events for two curves. A *realizing set* E is a subset of $\mathcal{E}$ such that the free space admits a monotone path from cell $\langle 1, 1\rangle$ to cell $\langle m, n\rangle$ without using an event in $\mathcal{E} \backslash E$: in other words, using only the events in E is already sufficient to admit the monotone path through the free space. When $\mathcal{E}$ contains more than one realizing event, some may be insignificant: they are never required to actually make a path in the free-space diagram. A realizing set is *minimal* if it does not contain a strict subset that is a realizing set. Such a minimal realizing set contains only *significant* events. Note that a realizing set cannot be empty and $\mathcal{E}$ is a trivial realizing set. This readily implies the following.

Observation 3. *For two polygonal curves P and Q with $m \geq 1$, $n \geq 1$ and $m + n > 2$, there exists a minimal realizing set.*

For the above observation, we require at least three edges in total, as the case that both curves have a single edge can be considered degenerate: the free-space diagram is a single cell and Observation 2 handles this case.

Note that $|\mathcal{E}| \leq K$ where K is the degeneracy of the two curves: if the two curves are not degenerate, $K = 1$ and thus the only event in the set is trivially a significant event.

Type-A events As mentioned above, type-A events are of little consequence, when computing the Fréchet distance. Indeed, type-A events do not give any choice as they correspond to the endpoints of the two curves—we must match these points regardless of the remainder of the curves. If we determine the lowest value of ε such that $\mathcal{F}_\varepsilon(P, Q)$ is feasible, we can combine this with the matched endpoints to determine the actual Fréchet distance. This is formalized in Lemma 2. To prove this statement, we first introduce the lemma below.

Lemma 1. *Let $\mu = (\sigma, \theta)$ be a matching for curves P and Q such that*

(1) *$m \geq 1, n \geq 1$ and $m + n > 2$;*
(2) *μ is linear in cell $\langle 1, 1\rangle$ and cell $\langle m, n\rangle$;*
(3) *$d_\mu(t) \leq \varepsilon$ for all t with $\sigma(t) \geq 1$ or $\theta(t) \geq 1$, and $\sigma(t) \leq m - 1$ or $\theta(t) \leq n - 1$.*

Then, it holds that

(a) *$d_\mu(t) \leq \max\{\varepsilon, \|p_0 - q_0\|\}$ for all t with $\sigma(t) \leq 1$ and $\theta(t) \leq 1$;*
(b) *$d_\mu(t) \leq \max\{\varepsilon, \|p_m - q_n\|\}$ for all t with $\sigma(t) \geq m - 1$ and $\theta(t) \geq n - 1$.*

Proof. We prove only claim (a); the proof of claim (b) is analogous. Let t be the lowest value for which $\sigma(t) \geq 1$ or $\theta(t) \geq 1$. The point $(\sigma(t), \theta(t))$ in the free-space diagram lies on the boundary of cell $\langle 1, 1\rangle$, precisely where μ exits the strict interior of the cell. By (1), t meets the conditions of (3) and thus we know that $d_\mu(t) \leq \varepsilon$. Since μ is linear in cell $\langle 1, 1\rangle$ and the free space in a single cell is convex [8], we know that $(\sigma(t'), \theta(t'))$ for $t' \leq t$ lies in the free space at value $\max\{\varepsilon, \|p_0 - q_0\|\}$. Thus, we conclude that $d_\mu(t) \leq \max\{\varepsilon, \|p_0 - q_0\|\}$ for all t with $\sigma(t) \leq 1$ and $\theta(t) \leq 1$, proving claim (a). □

Lemma 2. *Let ε be the lowest value such that $\mathcal{F}_\varepsilon(P, Q)$ is feasible for curves P and Q with $m \geq 1$, $n \geq 1$ and $m + n > 2$. It holds that $\delta_F(P, Q) = \max\{\varepsilon, \|p_0 - q_0\|, \|p_m - q_n\|\}$.*

Proof. As $\mathcal{F}_\varepsilon(P, Q)$ is feasible, there must be some monotone path π through the free space from a point π_s on the boundary of cell $\langle 1, 1\rangle$ to a point π_e on the boundary of $\langle m, n\rangle$. Consider the (monotone) path π' obtained by going straight from $(0, 0)$ to π_s, following π and finally from π_e straight to (m, n). This path describes a matching, that satisfies the requirements for Lemma 1. Hence, we know that the maximal distance in this matching is given by $\max\{\varepsilon, \|p_0 - q_0\|, \|p_m - q_n\|\}$, thus providing an upper bound on $\delta_F(P, Q)$. It also provides a lower bound, since ε is the lowest value such that $\mathcal{F}_\varepsilon(P, Q)$ is feasible, and both p_0 and q_0 as well as p_m and q_n are forced into any matching. □

Splitting on an event Consider a type-B realizing event between a vertex of P and an edge of Q; the symmetric case is analogous. On the curves the event corresponds to a vertex p_i and the closest point q on edge $q_j q_{j+1}$ with $q = (1 - \lambda)q_j + \lambda q_{j+1}$. In the free-space diagram it is a passage between the cells, represented by point $(i, j + \lambda)$ on the boundary between cell $\langle i, j + 1\rangle$ and $\langle i + 1, j + 1\rangle$. Splitting P and Q on this event creates two pairs of curves (see Fig. 5):

1. P_1 with vertices $p_0, \ldots, p_i$ and Q_1 with vertices $q_0, \ldots, q_j, q$;
2. P_2 with vertices $p_i, \ldots, p_m$ and Q_2 with vertices $q, q_{j+1}, \ldots, q_n$.

In the free-space diagram, these two pairs correspond to the rectangular region spanning from $(0, 0)$ to $(i, j + \lambda)$ and from $(i, j + \lambda)$ to (m, n) respectively.

Now, consider a type-C event between two vertices of P and an edge of Q; the symmetric case is again analogous. On the curves the event corresponds to two vertices p_i and p_k (with $i < k$) and a point q on edge $q_j q_{j+1}$ with $q = (1 - \lambda)q_j + \lambda q_{j+1}$. In the free space the passage of this event is a horizontal line segment, from $(i, j + \lambda)$ to $(k, j + \lambda)$. Splitting P and Q on this event creates the following two pairs of curves (see Fig. 6):

1. P_1 with vertices $p_0, \ldots, p_i$ and Q_1 with vertices $q_0, \ldots, q_j, q$;
2. P_2 with vertices $p_k, \ldots, p_m$ and Q_2 with vertices $q, q_{j+1}, \ldots, q_n$.

In the free-space diagram, these two pairs correspond to the rectangular region spanning from $(0, 0)$ to $(i, j + \lambda)$ and from $(k, j + \lambda)$ to (m, n) respectively. Note that any vertices of P in between p_i and p_k do not occur in either of the pairs.

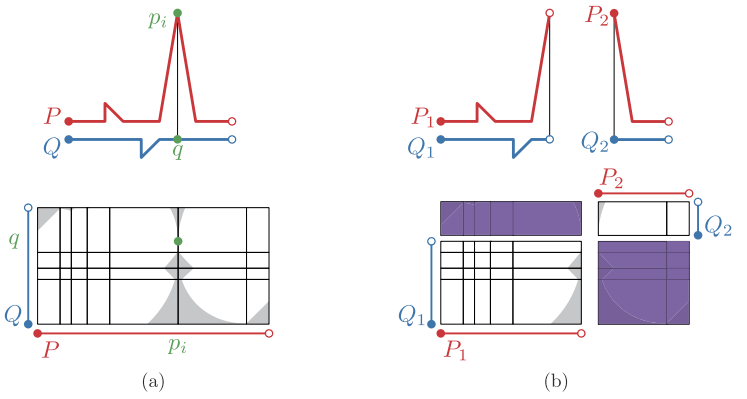

Fig. 5. (a) Curves with the free-space diagram for $\varepsilon = \delta_F(P, Q)$ and a critical event of type B. (b) The event splits each curve into two subcurves, creating two parts. The shaded areas indicate parts of the free-space diagram that are eliminated due to the split.

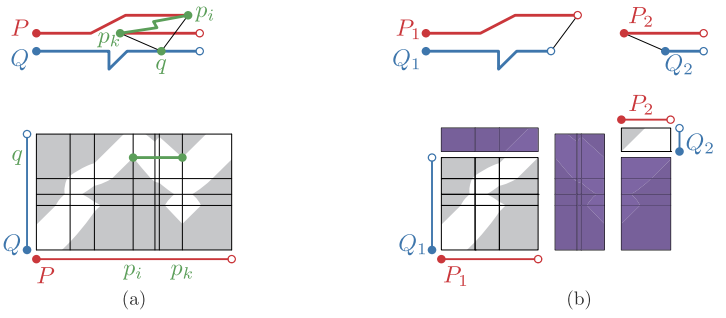

Fig. 6. (a) Curves with the free-space diagram for $\varepsilon = \delta_F(P, Q)$ and a critical event of type C. (b) The event splits each curve into two subcurves, and drops a part of P. The shaded areas indicate parts of the free-space diagram that are eliminated due to the split.

Finishing the proof of Theorem 1 Lemma 3, stated and proven below, directly implies that a locally correct Fréchet matching always exists. Informally, it states that curves have a locally correct matching for which all matched points are within distance ε (except in cell $\langle 1, 1\rangle$ or $\langle m, n\rangle$), if $\mathcal{F}_\varepsilon(P, Q)$ is feasible. Furthermore, this matching is linear inside every cell. The lemma is proven by splitting the curves on the events of a minimal realizing set and combining the locally correct matchings for the pieces in a single locally correct matching for the complete curves.

Lemma 3. *If $\mathcal{F}_\varepsilon(P, Q)$ of two polygonal curves P and Q is feasible, then there exists a locally correct Fréchet matching $\mu = (\sigma, \theta)$ such that $d_\mu(t) \le \varepsilon$ for all t with $\sigma(t) \ge 1$ or $\theta(t) \ge 1$, and $\sigma(t) \le m - 1$ or $\theta(t) \le n - 1$. Furthermore, μ is linear in every cell of the free-space diagram.*

Proof. We prove this by induction on $m + n$. The base cases ($m = 0$, $n = 0$, and $m = n = 1$) follow from the linear matchings prescribed by Observation 1 and Observation 2. For the latter, the matching has no t for which $\delta_\mu(t) \le \varepsilon$ must hold. For the former, note that there is only one matching to begin with: $\delta_\mu(t) \le \varepsilon$ is trivially satisfied for the necessary values of t—there are none if the other curve has complexity at most 1 and otherwise, we derive this immediately from the assumption that $\mathcal{F}_\varepsilon(P, Q)$ is feasible.

For induction, we assume that $m \ge 1$, $n \ge 1$, and $m + n > 2$. By Observation 3, a minimal realizing set E exists for P and Q, say at value ε_r. Since the realizing set is minimal, any monotone path from $\langle 1, 1\rangle$ to $\langle m, n\rangle$ in $\mathcal{F}_{\varepsilon_r}(P, Q)$ that uses these events must do so in a unique order: let $e_1, \ldots, e_k$ ($k \ge 1$) denote the events of E in this order. By definition, $\varepsilon_r \le \varepsilon$. Suppose that E splits curve P into $P_1, \ldots, P_{k+1}$ and curve Q into $Q_1, \ldots, Q_{k+1}$, where P_i has m_i edges, Q_i has n_i edges; see Fig. 7(a)–(b). By definition of a realizing event, none of the events in E occur on the left or bottom boundary of cell $\langle 1, 1\rangle$ or on the right or top boundary of cell $\langle m, n\rangle$. Hence, for any i ($1 \le i \le k+1$), it holds that $m_i \le m$, $n_i \le n$, and $m_i + n_i < m + n$. Since a path exists in the free-space diagram at ε_r through all events in E, the induction hypothesis implies that, for any i ($1 \le i \le k+1$), a locally correct matching $\mu_i = (\sigma_i, \theta_i)$ exists for P_i and Q_i such that μ_i is linear in every cell and $d_{\mu_i}(t) \le \varepsilon_r$ for all t with $\sigma_i(t) \ge 1$ or $\theta_i(t) \ge 1$, and $\sigma_i(t) \le m_i - 1$ or $\theta_i(t) \le n_i - 1$. Fig. 7(b) illustrates such matchings, whereas Fig. 7(c) highlights in orange, the cells that are "ignored" by the induction hypothesis—that is, where $d_{\mu_i}(t) \le \varepsilon_r$ does not necessarily hold. Concatenating these matchings with the passages of the events in E yields a matching $\mu = (\sigma, \theta)$ for (P, Q), as illustrated in Fig. 7(d). As we argue below, this matching satisfies the additional properties and is locally correct.

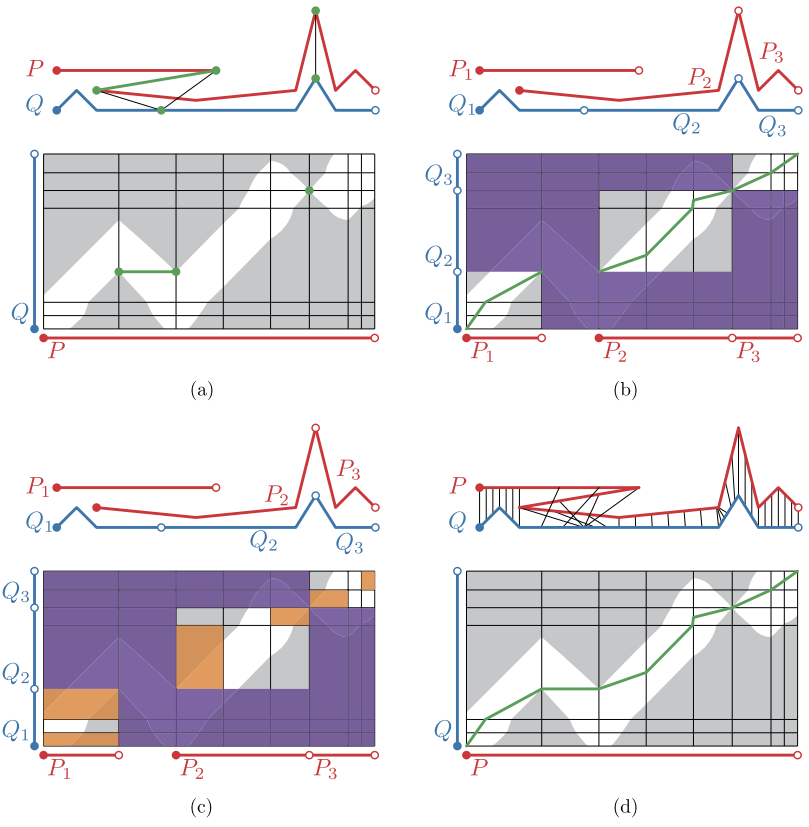

Fig. 7. Illustrations for the proof of Lemma 3. (a) Two concurrent critical events occur for P and Q: one of type C and one of type B. (b) These split the curves into three pairs of subcurves; the induction hypothesis provides us with a locally correct Fréchet matching for each of the three pairs. (c) The induction hypothesis does not give us information about the distances inside the orange cells. (d) Concatenating the matchings of the subcurves (b) with those of the critical events (a) yields a locally correct Fréchet matching between P and Q.

The matching of an event corresponds to a single point (type B, Fig. 5) or a horizontal or vertical line (type C, Fig. 6) in the free-space diagram. By induction, μ_i is linear in every cell. Since all events occur on cell boundaries, the cells of the matchings and events are disjoint. Therefore, the matching μ is also linear inside every cell.

We must also prove that the combined matching μ only matches points within distance ε except in cell $\langle 1,1\rangle$ and $\langle m,n\rangle$; note that the excepted cells are the leftbottom-most and topright-most orange cell in Fig. 7(c). For the parts of μ that originate from the passages of the realizing events in E, we have $d_\mu(t) \leq \varepsilon_r \leq \varepsilon$ by definition, as these passages must lie in the free space at ε_r. The induction hypothesis readily tells us that any parts that originate from some μ_i between the subcurves is also within this distance, excepting the parts in cells $\langle 1,1\rangle$ and $\langle m_i,n_i\rangle$ of the free-space diagrams of the subcurves: these are all marked in orange in Fig. 7. Since the last vertices of P_i and Q_i for $i>1$ as well as the first vertices of P_i and Q_i for $i<k$ are defined by the critical realizing events, we know that the distance between each of these pairs is ε_r. Hence, we may intuitively see that the line segments of μ_i in the orange cells in Fig. 7 must be within the free space of $\mathcal{F}_\varepsilon(P,Q)$. More formally, Lemma 1 tells us that $d_{\mu_i}(t) \leq \varepsilon_r$, even for t with: $\sigma_i(t) \leq 1$ and $\theta_i(t) \leq 1$ for all $1 < i \leq k$; and $\sigma_i(t) \geq m_i - 1$ and $\theta_i(t) \geq n_i - 1$ for all $1 \leq i < k$. Thus, it must hold that $d_\mu(t)$ is at most $\varepsilon_r \leq \varepsilon$ for all t with $\sigma(t) \geq 1$ or $\theta(t) \geq 1$, and $\sigma(t) \leq m-1$ or $\theta(t) \leq n-1$.

To show that μ is locally correct, suppose for contradiction that values a,b exist such that $\delta_F(P_\sigma[a,b], Q_\theta[a,b]) < d_\mu[a,b]$. If a,b are in between two consecutive events in E along the monotone path through the free space, we know that the submatching corresponds to one of the matchings μ_i. Since these are locally correct, we find that $\delta_F(P_\sigma[a,b], Q_\theta[a,b]) = d_\mu[a,b]$.

Hence, suppose that a and b are separated by at least one event of E. There are two possibilities: either $d_\mu[a,b] = \varepsilon_r$ or $d_\mu[a,b] > \varepsilon_r$. Since $d_\mu[a,b]$ includes a realizing event, $d_\mu[a,b] < \varepsilon_r$ cannot hold.

First, assume $d_\mu[a,b] = \varepsilon_r$. If $\delta_F(P_\sigma[a,b], Q_\theta[a,b]) < \varepsilon_r$, then a matching exists that does not use the events between a and b and has a lower maximum than $d_\mu[a,b] = \varepsilon_r$. Hence, the free space admits a monotone path from point $(\sigma(a),\theta(a))$ to point $(\sigma(b),\theta(b))$ at a lower value than ε_r. This implies that all events between a and b can be omitted, contradicting that E is a minimal realizing set.

Now, assume $d_\mu[a,b] > \varepsilon_r$. We argue as follows, to see that the maximum value of $d_\mu[a,b]$ is attained either at a or b. Let t' denote the highest t for which $\sigma(t) \leq 1$ and $\theta(t) \leq 1$, that is, the point at which the matching exits cell $(1,1)$. Sim-

ilarly, let t'' denote the lowest t for which $\sigma(t) \geq m-1$ and $\theta(t) \geq n-1$. Since $d_\mu(t) \leq \varepsilon_r$ for any $t' \leq t \leq t''$, $d_\mu(t) > \varepsilon_r$ is true only for $t < t'$ or $t > t''$. Suppose that $d_\mu(a) > \varepsilon_r$. Then $a < t'$ and μ is linear between a and t'. Therefore, $d_\mu(a) > d_\mu(t)$ for any t with $a < t < t'$. Analogously, if $d_\mu(b) > \varepsilon_r$, then $d_\mu(b) > d_\mu(t)$ for any t with $t'' < t < b$. Hence, we may indeed conclude that $d_\mu[a,b] = \max\{d_\mu(a), d_\mu(b)\}$. This maximum is a lower bound on the Fréchet distance, contradicting the assumption that $d_\mu[a,b]$ is larger than the Fréchet distance. Matching μ is therefore locally correct. □

4. Algorithm for locally correct Fréchet matchings

The existence proof directly translates into a recursive algorithm, which is given in Algorithm 1. Fig. 1 (left), Fig. 2 (left), Fig. 8, Fig. 10, and Fig. 11 (left) illustrate matchings computed with our algorithm. In this section, we prove the following theorem.

Theorem 2. *Algorithm 1 computes a locally correct Fréchet matching of two polygonal curves P and Q with m and n edges in $O((m+n)mn\log mn)$ time.*

The main idea is to partition the curves around a significant realizing event, which is then by the proof of the previous section suitable to split the curves for recursion. The crucial three steps in the algorithm aim to find this significant event e_r efficiently (lines 6 to 8). First, we compute the lowest value of ε such that $\mathcal{F}_\varepsilon(P,Q)$ is feasible. Second, we compute a realizing set E. Finally, we find a significant event in E, that is, an event in some minimal realizing set. Such a significant event is then appropriate to split the two curves on and allow for valid recursion, analogous to the induction proof given in the previous section.

Below, we give details for each of these steps. We use the notation of Alt and Godau [8]: $L^F_{i,j}$ denotes the interval of free space on the left boundary $L_{i,j}$ of cell $\langle i,j\rangle$; $L^R_{i,j}$ denotes the *reachable interval*, that is, the subset of $L^F_{i,j}$ that is reachable from point $(0,0)$ with a monotone path in the free space. Analogously, $B^F_{i,j}$ and $B^R_{i,j}$ are defined for the bottom boundary $B_{i,j}$.

Computing feasibility (line 6) First, we compute the lowest value of ε such that $\mathcal{F}_\varepsilon(P,Q)$ is feasible. That is, we compute the minimal value of ε such that the free space admits a monotone path from cell $\langle 1,1\rangle$ to cell $\langle m,n\rangle$. This corresponds to computing a "modified" Fréchet distance, one in which we ignore the type-A events. To this end, we apply the algorithm by Alt and Godau [8]. It needs only minor modifications in the decision algorithm. $L^R_{1,1}$ and $B^R_{1,1}$ are set to a nonempty interval including the origin, $(0,0)$, of the free-space diagram. Essentially, this ensures that any free space on the top and right side of cell $\langle 1,1\rangle$ is reachable. $L^F_{m+1,n}$ and $B^F_{m,n+1}$ are set to a nonempty interval including the destination, (m,n), of the free-space diagram. This ensures that (m,n) is reachable if any point on the bottom or left side of cell $\langle m,n\rangle$ is reachable. These changes have no effect on the asymptotic execution time of the algorithm, and thus it still runs in $O(mn\log n)$ time, assuming without loss of generality that $m \leq n$.

Algorithm 1 ComputeLCFM(P, Q)

Input: P and Q are curves with m and n edges
Output: A locally correct Fréchet matching for P and Q

1: **if** $m=0$ **or** $n=0$ **then**
2: **return** (σ,θ) where $\sigma(t)=t\cdot m$, $\theta(t)=t\cdot n$
3: **else if** $m=n=1$ **then**
4: **return** (σ,θ) where $\sigma(t)=\theta(t)=t$
5: **else**
6: Determine lowest value ε such that $\mathcal{F}_\varepsilon(P,Q)$ is feasible
7: Compute a realizing set E of events at value ε
8: Extract event e_r of a minimal realizing set contained in E
9: Split P into P_1 and P_2 according to e_r
10: Split Q into Q_1 and Q_2 according to e_r
11: $\mu_1 \to$ ComputeLCFM(P_1, Q_1)
12: $\mu_2 \to$ ComputeLCFM(P_2, Q_2)
13: **return** concatenation of μ_1, e_r, and μ_2

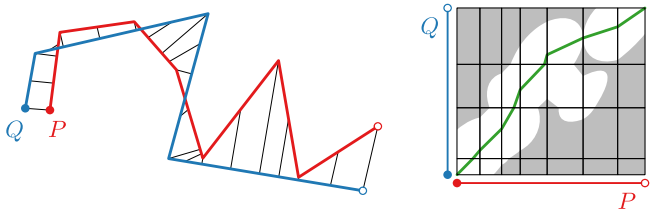

Fig. 8. Locally correct Fréchet matching produced by Algorithm 1 and the corresponding free-space diagram for $\varepsilon = \delta_F(P,Q)$.

Computing a realizing set (line 7) In this second step, we compute some (possibly nonminimal) realizing set E at the value of ε computed in the previous step. Recall that this is a subset of $\mathcal{E}$, all concurrent realizing events at value ε, such that the free space admits a monotone path from cell $\langle 1,1\rangle$ to cell $\langle m,n\rangle$ without using events in $\mathcal{E}\backslash E$. We present a simple $O(mn)$-time algorithm to find a realizing set for a given value of ε. To do so, we run the decision algorithm by Alt and Godau [8] again with value ε, which we now modify as follows: we keep track of any realizing events that are encountered as the reachable space at value ε is established.

We observe that the occurrence of a *singleton reachable interval*—a reachable interval that contains only a single point—corresponds directly to a realizing event. In particular, that realizing event *ends* at that singleton reachable interval and the corresponding boundary, meaning that the rightmost and upmost point in the free-space diagram of the corresponding passage (see also Fig. 4) coincides with the singleton reachable interval and thus lies on its boundary. Although not all realizing events in $\mathcal{E}$ have to correspond to a singleton reachable interval, the events that do correspond to such an interval in fact form a realizing set. This is formalized in the following lemma.

Lemma 4. *If $L^R_{i,j}$ or $B^R_{i,j}$ is a singleton, then a realizing event ends at $L_{i,j}$ or $B_{i,j}$ respectively. The set of all events ending at boundaries with a singleton reachable interval is a realizing set.*

Proof. The reachable interval $L^R_{i,j}$ is determined by the maximum value in the free space of the boundary, $L^F_{i,j}$, and the minimum in some reachable interval $L^R_{i',j}$ with $i' \le i$. If $L^R_{i,j}$ is a singleton, then this minimum and maximum coincide and thus this corresponds to an event of type B ($i' = i$) or type C ($i' < i$). The argument for a reachable interval $B^R_{i,j}$ is analogous.

Let π be some monotone path in the free space between the first and last cell. Path π passes through one or more realizing events. We are done if all these events end at a boundary for which the reachable interval is a singleton. So, assume that e is the last event along π that ends on boundary $L_{i,j}$ for which $L^R_{i,j}$ is not a singleton. Note that $L^R_{i,j}$ cannot be empty as π passes through it. Event e must be of type C, as type B implies that $L^F_{i,j}$ is a singleton and thus $L^R_{i,j}$ would be as well. Hence, we let $L_{i',j}$ denote the boundary where event e starts and $i' < i$. There must be some boundary $B_{i^*,j}$ such that $i' \le i^* < i$ and $B^R_{i,j}$ is nonempty. This implies that there is a monotone path π' in the free space up through $B_{i^*,j}$ to $L_{i,j}$. In particular, π' does not pass through e. Hence, the concatenation of π' with the subpath of π starting at $L_{i,j}$ is a monotone path in the free space between the first and last cell. Moreover, any event along π' that does not end at a boundary with a singleton reachable interval must end at some boundary $L_{x,y}$ or $B_{x,y}$ with $x < i$ and $y < j$. Hence, repeating the replacement above must terminate as there are only a finite number of boundaries. This proves that there is some monotone path between the first and last cell that passes only through events that end on a boundary with a singleton reachable interval. Thus these events form a realizing set. □

From the above, we learn how to compute a realizing set: we simply record occurrences of singleton intervals. However, this gives us only the boundary on which the event ends; we also need to know the boundary on which the event starts. Assume a singleton reachable interval is found on boundary $L_{i,j}$. Let $h < i$ be the largest value such that $B^R_{h,j}$ is nonempty: all bottom boundaries between h and i are not reachable. Then the singleton is caused by the maximum of the lower endpoints (minima) of intervals $L^F_{g^*,j}$ with g^* with $h < g^* \le i$. This means that the event that causes the singleton starts at $L_{g^*,j}$ and ends at $L_{i,j}$. We refer to g^* as the *event horizon*. For each row we maintain this event horizon. After computing $L^R_{i,j}$ but before checking for a singleton, the event horizon is updated to j if either $B^R_{i-1,j}$ is nonempty or the minimum of $L^F_{g^*,j}$ is less than or equal to the minimum of $L^F_{i,j}$. If this is not the case, g^* maintains its value. This ensures that, if $L^R_{i,j}$ is a singleton and there would be multiple events ending at this boundary, we find the event that starts at the rightmost column. Columns and horizontal boundaries are dealt with analogously. Maintaining the event indices incurs no asymptotic overhead on the basic decision algorithm. Hence, this modified algorithm that finds a realizing set also runs in $O(mn)$ time.

Checking whether an interval of floating-point numbers is a singleton must be able to deal with imprecision, and thus lead to instability in an implementation: intervals may be considered singletons when they are in fact very small intervals and vice versa. Instead of checking whether $L^R_{i,j}$ is a singleton, we may compute the value of the critical event between $L_{i,j}$ and $L_{g^*,j}$ directly and compare it to ε. This depends on the input coordinates directly, rather than on a sequence of computations that establish the lower bound of interval $L^R_{i,j}$. Though still working with floating-point numbers, this tends to be slightly more stable.

Finding a significant event (line 8) In this last step, we find a significant event e_r in the realizing set E computed in the previous step for value ε. That is, e_r must be contained in some *minimal* realizing set.

If events end at the same boundary, then these occur in the same row (or column) and it suffices to consider only the event that starts at the rightmost column (or highest row). The algorithm described above for line 7 to compute E collects exactly those events. Hence, we may assume here that the events in E end at different cell boundaries. As a result, E has at most $O(mn)$ events. Note that in degenerate cases, the size of E can be $\Theta(mn)$ as illustrated in Fig. 9. However, only minimal realizing set can have at most $O(m+n)$ events, since a matching can visit only this many cells of the free-space diagram.

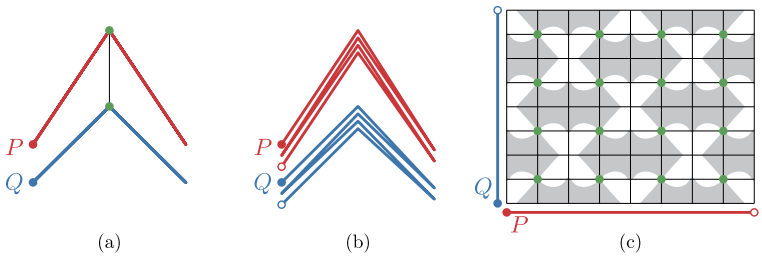

Fig. 9. Two curves with $\Theta(mn)$ realizing events. (a) Two curves zigzagging back and forth, with coinciding vertices. The realizing events are indicated. (b) Conceptual view of the two curves. (c) The free-space diagram and the corresponding realizing events (green dots).

To find a significant event in E, we proceed as follows. Fix the order of events in E. Let E_k denote the first k events of E and let e_k denote the kth event. We use a binary search on E to find the r such that E_r contains a realizing set, but E_{r-1} does not. This implies that event e_r is contained in a minimal realizing set. Note that r is unique due to monotonicity. What remains is to describe an algorithm that checks whether E_k is a realizing set.

To determine whether some E_k is a realizing set, we check whether $\mathcal{F}_\varepsilon(P, Q)$ remains feasible without using the events of $\mathcal{E}\backslash E_k$: is there still a monotone path through the free space from cell $\langle 1,1\rangle$ to cell $\langle m,n\rangle$? We again use a third (and last) modified version of the decision algorithm by Alt and Godau [8]. For each event, we store its index in E with the boundary it ends at. When $L^R_{i,j}$ is computed, we check whether $L^R_{i,j}$ is a singleton and whether an index k' is stored with $L_{i,j}$. If this is not the case, then no realizing event ends at $L_{i,j}$ or it is not required to reach it. If the reachable interval is a singleton and k' exists, event $e_{k'}$ is required to reach $L_{i,j}$ (as argued in the previous paragraph). We then check whether this event may be used by comparing k and k'. If $k' \leq k$, no action is taken; otherwise, the event may not be used and $L^R_{i,j}$ is replaced with the empty interval. In this modified algorithm (m,n) is reachable if and only if E_k is a realizing set. The additional check takes only constant time per cell boundary. Thus, we decide in $O(mn)$ time whether E_k is a realizing set.

As mentioned before, checking for singleton intervals may lead to numeric instability in an implementation. To remedy this problem, we again maintain the event horizon g^* for each row and column. Assume that $L_{i,j}$ has some associated event e that starts on $L_{g,j}$. Event e forces $L^R_{i,j}$ to be a singleton if and only if $g = g^*$. Therefore, we can replace checking whether $L^R_{i,j}$ is a singleton with an equality check of two integers.

Analysis Algorithm 1 follows a sequence of steps to enable valid recursion. The first and second step (lines 6 and 7) take $O(mn \log mn)$ time combined. The third step (line 8) performs a binary search on the realizing set and thus depends on its size. As argued, it contains at most $O(mn)$ events. However, concurrent events can be considered degenerate. To nuance this analysis, we therefore consider the degeneracy K, that is, the maximum number of concurrent events. The third step then takes $O(mn \log K)$ time. Splitting the curves P and Q according to e_r and concatenating the matchings can be done in $O(m+n)$ time and is thus subsumed under the previous steps. Each recursion step splits the problem into two smaller problems, and the recursion ends when $mn \leq 1$. This results in an additional factor of $O(m+n)$. Thus the total execution time is $O((m+n)mn \log mn)$.

Substituting other methods We used Alt and Godau's method [8] to compute the Fréchet distance. However, it can be substituted by any other algorithm, provided that the modification can be made to "ignore" the first and last cell. In general, this leads to an algorithm that runs in $O((m+n)(T(m,n) + mn \log K))$ time, where $T(m,n)$ indicates the computation time of the (modified) algorithm used to compute the Fréchet distance. Whereas earlier the degeneracy was subsumed under the computation of the Fréchet distance, it is now a relevant factor if $T(m,n) = o(mn \log mn)$.

As an example, we may substitute the algorithm by Buchin et al. [11]. Similar to Alt and Godau's algorithm, this algorithm can easily be adapted by treating the four free-space intervals on cells $\langle 1,1\rangle$ and $\langle m,n\rangle$ as reachable. For very degenerate curves—$K = (m+n)$—this substitution does not lead to an improved execution time. However, there is another pitfall here: the improved algorithm is faster only if $m = \omega(n/\log n)$, assuming $m \leq n$ (see [2, Chapter 4] for details). In other words, the curves must be "balanced". As we cannot control the recursion, we cannot guarantee that this is the case, even if the curves initially are balanced.

Vertex sampling Polygonal curves are often a discrete representation of continuous curves or motion, the vertices being samples of the underlying curve. Increasing the sampling rate thus leads to more vertices describing this same continuous curve. We observe that supersampling (introducing extra vertices) may alter the result of the algorithm, even if these vertices do not modify the shape itself (see Fig. 10). This implies that the algorithm depends not only on the shape of the curves, but also on the sampling. Increasing the sampling further and further seems to result in a matching that decreases the matched distance as much as possible within a cell. However, since cells are rectangles, there is a slight preference for taking longer diagonal paths.

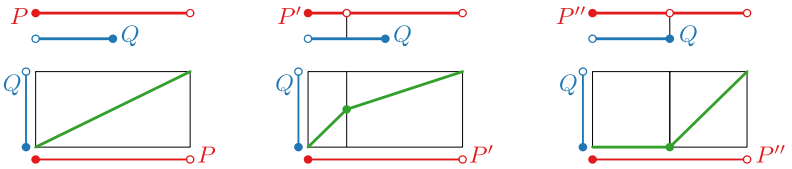

Fig. 10. Different sampling may result in different matchings being computed by the algorithm; note that all three matchings are a valid locally correct Fréchet matching for any of the three inputs.

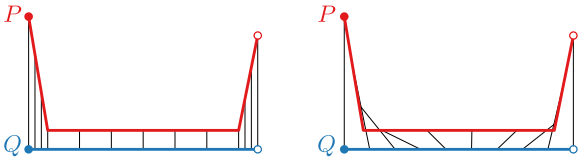

Fig. 11. Two locally correct Fréchet matchings for P and Q. (a) Matching that decreases distances as quickly as possible. (b) Shortest matching.

Further restrictions Two curves may still have many locally correct Fréchet matchings: the algorithm computes just one of these. For most applications, we expect that it is desirable to restrict to locally correct matchings, as this builds purely on the assumption that the Fréchet distance is a good way of quantifying similarity. However, it is often desirable to find the "best" Fréchet matching, though what defines "best" likely depends on the intended application. For example, the curves in Fig. 11 admit two matchings, either of which may be considered better than the other: this depends on what the curves represent and on the purpose of the matching. Corresponding to these two examples, we mention two possible criteria to further restrict locally correct Fréchet matchings.

The first criterion is the "length" of the matching, measured by its path length in the free-space diagram. That is, we prefer the matching to progress on both curves when possible; matching a short subcurve of P to a long subcurve of Q would be undesirable. Of course, the matching should still be locally correct to ensure that the matching doesn't match far-away points unnecessarily. A slope constraint [14] indirectly restricts the length of a matching. Thus this criterion may also be considered for applications in which speed limits are desirable. An alternative to the shortest locally correct Fréchet matching would be to consider the computation of a locally correct matching that adheres to length or slope constraints.

The second criterion again considers the matched distances. Local correctness is about avoiding large distances that would exceed the (local) Fréchet distance, by considering subcurves that are induced by the matching. We may strengthen this idea by desiring that all matched distances should be kept as low as possible. These "locally optimal" Fréchet matchings may require a steepest descent method to find the matching. An important question is how to parameterize the descent. Using the L_2 norm for the descent results in nonlinear matchings with possible algebraic issues. Since the initial publication of the results described here, Rote [15] has shown that a descent under the L_∞ norm yields a linear locally optimal matching (referred to as a lexicographic Fréchet matching in [15]). He shows that these are computable in $O(N^3 \log N)$ time assuming nondegenerate curves ($K = 1$). Using a different model to measure "speed" of a matching, Maheshwari et al. [16] process the result of our algorithm to reduce the matched distances, though this increases the length of the matching.

5. Locally correct discrete Fréchet matchings

Here we study the discrete variant of Fréchet matchings, one in which only the vertices of curves are matched. The discrete Fréchet distance can be computed in $O(mn)$ time via dynamic programming [10]. Here, we extend this simple algorithm to show that a locally correct discrete Fréchet matching can also be computed in $O(mn)$ time.

Grids Since we are interested only in matching vertices of the curves, the free-space diagram turns into a grid. Suppose we have two curves P and Q with m and n edges respectively. These convert into a grid G of nonnegative values with $m+1$ columns and $n+1$ rows. Every column corresponds to a vertex of P, every row to a vertex of Q. Any node of the grid $G[i, j]$ corresponds to the pair of vertices (p_i, q_j). Its value is the distance between the vertices: $G[i, j] = \|p_i - q_j\|$. Analogous to free-space diagrams, we assume that $G[0, 0]$ is the bottomleft node and $G[m, n]$ the topright node.

Matchings A monotone path π is a sequence of grid nodes $\pi(1), \ldots, \pi(k)$ such that every node $\pi(i)$ $(1 < i \leq k)$ is the above, right, or above/right diagonal neighbor of $\pi(i-1)$. In the remainder of this section a path refers to a monotone path unless indicated otherwise. A monotone discrete matching of the curves corresponds to a path π such that $\pi(1) = G[0, 0]$ and $\pi(k) = G[m, n]$. A path π is called *locally correct* if for all $1 \leq t_1 \leq t_2 \leq k$, $\max_{t_1 \leq t \leq t_2} \pi(t) = \min_{\pi'} \max_{1 \leq t \leq k'} \pi'(t)$, where π' ranges over all paths starting at $\pi'(1) = \pi(t_1)$ and ending at $\pi'(k') = \pi(t_2)$.

We also define a slightly stronger version, in which the common endpoints of submatchings—which may otherwise be the maximum—are not taken into account. Formally, a path π is *strongly locally correct* if for all $1 \leq t_1 \leq t_2 \leq k$, $\max_{t_1 < t < t_2} \pi(t) = \min_{\pi'} \max_{1 < t < k'} \pi'(t)$, where π' ranges over all paths starting at $\pi'(1) = \pi(t_1)$ and ending at $\pi'(k') =$

Algorithm 2 ComputeDiscreteLCFM(P, Q).

Input: P and Q are curves with m and n edges
Output: A strongly locally correct discrete Fréchet matching for P and Q

```
1: Construct grid G for P and Q
2: Let T be a tree consisting only of the root G[0, 0]
3: for i ← 1 to m do
4:     Add G[i, 0] to T
5: for j ← 1 to n do
6:     Add G[0, j] to T
7: for i ← 1 to m do
8:     for j ← 1 to n do
9:         AddToTree(T, G, i, j)
10: return path in T between G[0, 0] and G[m, n]
```

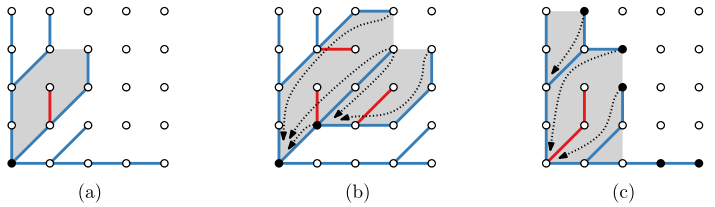

Fig. 12. (a) Face of tree (gray area) with its unique sink (solid dot). A red line represents a dead path. (b) Two adjacent faces with some shortcuts indicated. (c) Tree with three faces. Solid dots indicate growth nodes with a growth node as parent. These nodes are incident to at most one face. All shortcuts of these nodes are indicated.

$\pi(t_2)$. Note the inequalities in the max statements are now strict inequalities. The items excluded from the domain here have in fact identical values, since the π' must start and end at the same points as the subpath of π: as a result, a strongly locally correct path is also locally correct.

Algorithm In order to compute a locally correct discrete Fréchet matching, we present an algorithm that in fact computes a strongly locally correct discrete Fréchet matching. In other words, the algorithm computes a strongly locally correct path from $G[0,0]$ to $G[m,n]$ in a grid G of nonnegative values. To this end, the algorithm incrementally constructs a tree T on the grid. Tree T is rooted at $G[0,0]$ and each path in T is strongly locally correct. This is summarized in Algorithm 2. We define a *growth node* as a node of T that has a neighbor in the grid that is not yet part of T: a new branch may sprout from such a node. The growth nodes form a sequence of horizontally or vertically neighboring nodes. A *living node* is a node of T that is not a growth node but is an ancestor of a growth node. A *dead node* is a node of T that is neither a living nor a growth node, that is, it has no descendant that is a growth node; a *dead path* is a sequence of dead nodes and the links to their parents. Every pair of nodes in this tree has a *lowest common ancestor* (LCA). When we add a new node to T, we have to decide on a growth node to be its parent such that paths to the new node are locally correct. To this end, we compare the maximum values encountered after the LCAs of the growth nodes. We provide more details on this procedure later in this section. A *face* of T is the area enclosed by the segment between two horizontally or vertically neighboring growth nodes (without one being the parent of another) and the paths to their LCA. The unique *sink* of a face is the node of the grid that is in the lowest column and row of all nodes on the face. Fig. 12(a)–(b) shows some examples of faces and their sinks.

Shortcuts To avoid repeatedly walking along the tree to compute maxima, we maintain up to two *shortcuts* from every node in the tree. The segment between the node and its parent is incident to up to two faces of the tree. The node maintains shortcuts to the sinks of these faces: the shortcut stores the maximum value encountered on the path between the node and the sink (excluding the value of the sink, but including the node itself). Fig. 12(b) illustrates some shortcuts.

With these shortcuts, the maximum up to the LCA of two (potentially diagonally) neighboring growth nodes is computed in constant time, as the LCA is either one or two shortcuts away from the two nodes. This is done as follows (refer also to Fig. 12(b)). For two horizontally or vertically adjacent growth nodes, we observe that the segment connecting these nodes defines a unique face in the tree: the LCA of the two growth nodes is the sink of this face. Hence, the shortcuts stored with these two nodes provide us the information we need to decide in $O(1)$ time which has the lower maximum value along the path to the LCA. For two diagonally adjacent growth nodes, we observe that they must have a common growth node that is horizontally adjacent to one and vertically adjacent to the other. This then defines two faces and we observe that the LCA we seek is the sink of one of these faces. We need at most two shortcuts to go from either growth node to the LCA and can thus find the maximum value along the path to the LCA in $O(1)$ time.

Note that a node g of the tree that has a growth node as parent is incident to at most one face (see Fig. 12(c)). We need the "other" shortcut only when the parent of g has a living parent. Therefore, the value of this shortcut can be obtained

in constant time by using the shortcut of the parent. When the parent of g is no longer a growth node, then g obtains its own shortcut.

Extending the tree Algorithm 3 summarizes the steps required to extend the tree T with a new node. Node $G[i, j]$ has three *candidate parents*, $G[i-1, j]$, $G[i-1, j-1]$, and $G[i, j-1]$. Each pair of these candidates has an LCA. For the actual parent of $G[i, j]$, we select the candidate c such that for any other candidate c', the maximum value from c to their LCA is at most the maximum value from c' to their LCA—both excluding the LCA itself. As explained above, the shortcuts allow us to make this decision in $O(1)$ time as the candidates are neighboring growth nodes.

We must be consistent when breaking ties between candidate parents that have the same maximum value along the path to their LCA. To this end, we use the preference order of $G[i-1, j] \succ G[i-1, j-1] \succ G[i, j-1]$. Since paths in the tree cannot cross, this order is consistent between two paths at different stages of the algorithm. Note that a preference order that prefers $G[i-1, j-1]$ over both other candidates or vice versa results in an incorrect algorithm, as shown in Fig. 13. The root cause is the comparison at the cells with the thick outline, at which the algorithm must decide between parents with a maximum value of 6 to their LCA. With the incorrect preference order, it always selects the diagonal parent but for one, it is preferred over a downward parent and for the other over a leftward parent. The fact that this swaps from a counterclockwise to a clockwise preference causes a failure of the algorithm. This shows that the preference order can affect the correctness of the algorithm and we explicitly mark where we need this during the proof.

When a dead path is removed from the tree, adjacent faces merge and a sink may change. Hence, shortcuts have to be extended to point toward the new sink. Fig. 14 illustrates the incoming shortcuts at a sink and the effect of removing a dead path on the incoming shortcuts. The algorithm does not need to remove dead paths that end in the highest row or rightmost column.

Finally, $G[i-1, j]$, $G[i, j-1]$, and $G[i, j]$ receive shortcuts where necessary. $G[i-1, j]$ or $G[i, j-1]$ needs a shortcut only if its parent is $G[i-1, j-1]$. $G[i, j]$ needs two shortcuts if $G[i-1, j-1]$ is its parent, only one shortcut otherwise.

Algorithm 3 AddToTree(T, G, i, j).

Input: G is a grid of nonnegative values; any path in tree T is locally correct
Output: node $G[i, j]$ is added to T and any path in T is locally correct

1: $parent(G[i, j]) \leftarrow$ candidate parent with lowest maximum value to LCA
2: **if** $G[i-1, j-1]$ is dead **then**
3: Remove the dead path ending at $G[i-1, j-1]$ and extend shortcuts
4: Make shortcuts for $G[i-1, j]$, $G[i, j-1]$, and $G[i, j]$ where necessary

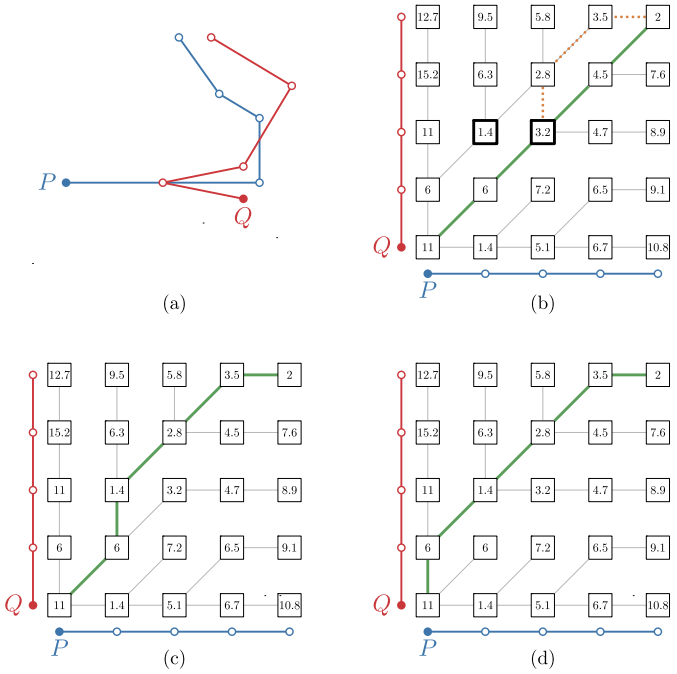

Fig. 13. Example illustrating that a preference order that prefers the diagonal parent over the two orthogonal ones may lead to incorrect results. (a) The input curves. (b) The tree constructed by the algorithm using the incorrect order. The orange dotted path proves that the matching described by the thick line is not locally correct. (c)–(d) The tree constructed by the algorithm when preferring the downwards and leftward neighbor respectively.

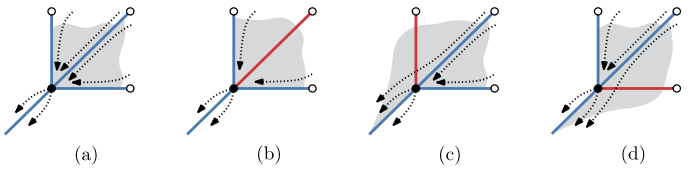

Fig. 14. (a) Each sink has up to four sets of shortcuts. (b)–(d) Removing a dead path (red) extends at most one set of shortcuts.

Correctness Lemma 5 below implies the correctness of Algorithm 2.

Lemma 5. *Algorithm 2 maintains the following invariant: any path in T is strongly locally correct.*

Proof. To prove this lemma, we strengthen the invariant to make the role and necessity of the preference order explicit.

Invariant. We are given a tree T such that every path in T is strongly locally correct. In constructing T, any ties were broken using the preference order.

Initialization. Tree T is initialized such that it contains two types of paths: either between grid nodes in the first column or in the first row. In both cases there is only one path between the endpoints of the path. Therefore, this path must be strongly locally correct. Since every node has only one candidate parent, T adheres to the preference order.

Maintenance. The algorithm extends T to T' by including node $g = G[i, j]$. Recall that this is done by connecting g to one of its candidate parents ($G[i-1, j]$, $G[i-1, j-1]$, or $G[i, j-1]$): we select one such that, compared to either other candidate, its maximum value along its path to the LCA is at most the maximum value along the path to the LCA from the other candidate. We must now prove that any path in T' is strongly locally correct. From the invariant, we conclude that only paths that end at g could falsify this statement. We prove this via contradiction.

Assuming T' is not strongly locally correct, there must be an invalidating path that ends at g. This path must use one of the candidate parents of g as its second-to-last node. We distinguish three cases on how this path is situated compared to T'. The last case, however, needs two subcases to deal with candidate parents that have the same maximum value on the path to their LCA. The four cases are illustrated in Fig. 15. First, we introduce some common notation.

The invalidating path must diverge from the path in T to g. For each case, we consider the path π_i starting at the node before the first node that is different and end at a candidate parent of g in the invalidating path. Note that π_i need not be disjoint of the paths in T'. Slightly abusing notation, we also use a path π' to denote its maximum value, excluding the first and last node, i.e. $\max_{1<t<k'} \pi'(t)$. We use p to denote the parent of g in T', that is, the selected candidate.

Case (a). Path π_i ends at p. Path π is the path in T' between the first and last vertex of π_i. Since (π_i, g) is the invalidating path, we know that $\max\{\pi_i, g\} < \max\{\pi, g\}$. This implies that $\pi_i < \pi$. In particular, this means that π, a path in T, is not strongly locally correct: a contradiction.

Case (b). Path π_i does not end at p and $\pi_i(1)$ is not a descendant of the LCA of p and the last node of π_i. Path π_1 is the path in T from $\pi_i(1)$ to this LCA. Paths π_2 and π_3 are paths in T that start at this LCA and end at p and the last node of π_i respectively. Since the endpoint of π_2 was chosen as parent over the endpoint of π_3, we know that $\pi_2 \leq \pi_3$. Furthermore, since (π_i, g) is the invalidating path, we know that $\max\{\pi_i, g\} < \max\{\pi_1, \pi_2, g\}$. These two inequalities imply $\max\{\pi_i, g\} < \max\{\pi_1, \pi_3, g\}$. This in turn implies $\pi_i < \max\{\pi_1, \pi_3\}$. Since (π_1, π_3) is a path in T and the inequality implies that it is not strongly locally correct, we again have a contradiction.

Case (c). Path π_i does not end at p and the first node of π_i is a descendant of the LCA of p and the last node of π_i. Let π_1 be the path from this LCA to $\pi_i(1)$. Path π_2 starts at $\pi_i(1)$ and ends at p. Path π_3 starts at $\pi_1(1)$ and ends at the last node of π_i. In this case, we must explicitly consider the possibility of two paths having equal values. Hence, we distinguish two subcases.

Case (c-i). In the first subcase, we assume that the endpoint of π_2 was chosen as parent since its maximum value is strictly lower: $\max\{\pi_1, \pi_2\} < \pi_3$. Since (π_i, g) is the invalidating path, we know that $\max\{\pi_i, g\} < \max\{\pi_2, g\}$. Since $\pi_2 \leq \max\{\pi_1, \pi_2\}$ is always true, we obtain that $\max\{\pi_i, g\} < \max\{\pi_3, g\}$. This in turn implies that $\pi_i < \pi_3$. Similarly, since $\pi_1 \leq \max\{\pi_1, \pi_2\}$, we know that $\pi_1 < \pi_3$. Combining these last two inequalities yields $\max\{\pi_1, \pi_i\} < \pi_3$. Since π_3 is a

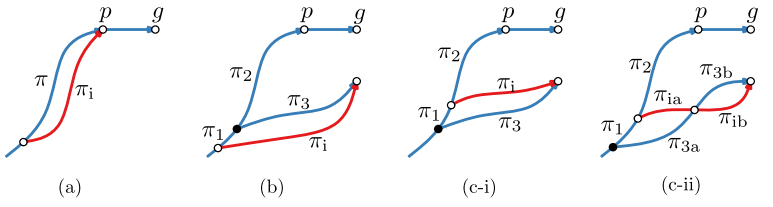

Fig. 15. Four cases of the invalidating path π_i (in red) for the proof of Lemma 5.

path in T and the inequality implies that it is not strongly locally correct, we again have a contradiction. (Note that with $\max\{\pi_1, \pi_2\} \leq \pi_3$, we can at best derive $\max\{\pi_1, \pi_i\} \leq \pi_3$ which is not strong enough to contradict the invariant on T.)

Case (c-ii). In the second subcase, we assume that the endpoint of π_2 was chosen as parent based on the preference order: the maximum values are equal, thus $\max\{\pi_1, \pi_2\} = \pi_3$. If π_i does not intersect π_3 before their common last node, we must conclude that $\pi_i > \pi_3$: otherwise, the preference order would be violated at this common last node. In particular, this implies that π_i cannot be an invalidating path. If π_i does intersect π_3 before their common last node, we proceed as follows. We partition π_i into π_{ia} and π_{ib}: the split is based on the first node that π_i and π_3 have in common. At the same node, we also partition π_3 into π_{3a} and π_{3b}. We now obtain two more cases, $\pi_{3a} < \max\{\pi_1, \pi_{ia}\}$ and $\pi_{3a} \geq \max\{\pi_1, \pi_{ia}\}$. In the former case, we obtain that $\max\{\pi_{3a}, \pi_{ib}\} < \max\{\pi_1, \pi_2\}$ and thus (π_{3a}, π_{ib}, g) is also an invalidating path. Since this path starts at the LCA of π_2 and π_3, this is already covered by case (b). In the latter case, we have that either path π_{3a}—which is in T—is not strongly locally correct (contradicting the invariant) or there is equality between the two paths (π_1, π_{ia}) and π_{3a}. In case of equality, we observe that (π_1, π_{ia}) and π_{3a} arrive at their endpoint in the same order as π_2 and π_3 arrive at g.[2] Thus T does not adhere to the preference order to break ties. This contradicts the invariant.

In all cases, we find that the assumption of an invalidating path contradicts the invariant. Therefore, we conclude that Algorithm 2 maintains the required invariant: all paths in T are strongly locally correct. □

Execution time Most steps in the algorithm are easily done in $O(1)$ time; the only exception is the removal of a dead path (Algorithm 3, line 3). When a dead path π_d is removed, we may need to extend a list of incoming shortcuts at $\pi_d(1)$, the node that remains in T. Let k denote the number of nodes in π_d. The lemma below relates the number of extended shortcuts to the size of π_d. The main observation is that the path requiring extensions starts at $\pi_d(1)$ and ends at $G[i-1, j]$ or $G[i, j-1]$, since $G[i, j]$ has not yet received any shortcuts.

Lemma 6. *A dead path π_d with k nodes results in at most $2 \cdot k - 1$ extensions.*

Proof. Since π_d is a path with k nodes, it spans at most k columns and k rows. When a dead path is removed, its endpoint is $G[i-1, j-1]$. Let π_e denote the path of T that requires extensions. Both paths start at the same node: $\pi_d(1) = \pi_e(1)$. The endpoint of π_e is at either $G[i-1, j]$ or $G[i, j-1]$, since $G[i, j]$ has not yet received shortcuts when the dead path is removed. If the endpoint of π_e is not the parent of $G[i, j]$, then it has at most one child; it is a growth node and thus any of its descendants are also growth nodes. Hence, these descendants have a parent that is a growth node and thus do not have shortcuts that need to be extended. Fig. 16 illustrates these situations. Hence, we know that π_e spans either $k+1$ columns and k rows or vice versa. Therefore, the maximum number of nodes in π_e is $2 \cdot k$, since it must be monotone. Since $\pi_e(1)$ does not have a shortcut to itself, there are at most $2 \cdot k - 1$ incoming shortcuts from π_e at $\pi_d(1)$. □

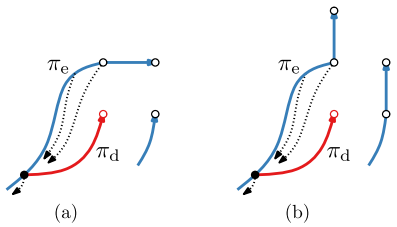

Fig. 16. (a) A dead path π_d and the corresponding path π_e in which shortcuts that need extension must start. (b) Endpoint of π_e has one child. None of its descendants has a shortcut to $\pi_e(1)$.

Hence, we can charge every extension to one of the $k-1$ dead nodes (all but $\pi_d(1)$). Since these nodes are removed from T, a node gets at most 3 charges. Due to the existing shortcuts, each extension can be done in constant time. Thus the total execution time of the algorithm is $O(mn)$.

The dynamic program to compute the discrete Fréchet distance can be processed on a per-row or per-column basis, thus requiring only $O(\min\{m, n\})$ additional memory. This can also be done in our algorithm to compute a locally correct discrete Fréchet matching. However, we maintain tree T to store the locally correct paths. Naively speaking, this tree spans the entire grid, thus requiring $O(mn)$ space. By appropriately choosing per-row or per-column processing, T has at most $O(\min\{m, n\})$ growth nodes: any leaves that do not have an unprocessed neighbor are part of a dead branch and are thus removed from T. Any living node with exactly one child is never used, as it cannot be the sink of a face, nor is it ever the root of a dead branch. Therefore, such nodes can easily be removed without additional overhead: this yields a compressed tree T in which

[2] This is the reason why a preference order, that prefers $G[i-1, j-1]$ over both other candidates or vice versa, does not work: the cyclic order of the paths arriving at the vertices may be the same, but the preference order can be different.

all living nodes (that are not growth nodes) have at least two children. Thus we conclude that the compressed tree contains at most $O(\min\{m,n\})$ nodes.

We summarize the findings of this section in the following theorem.

Theorem 3. *Algorithm 2 computes a locally correct discrete Fréchet matching of two polygonal curves P and Q with m and n edges in $O(mn)$ time and $O(\min\{m,n\})$ additional space.*

Further restrictions As in the continuous case, we may think of further restrictions such minimizing the number of matched pairs (length in the continuous case) or keeping low distances (local optimality).

It would be comparatively easy to augment the tree with additional information such as the number of diagonals to the root, which would allow of assessing the length of two discrete matchings. However, if we come across a tie, we must use the preference order, otherwise the computation may be faulty; as a result, we cannot use such augmented information to break the tie differently. Indeed, this would lead to the wrong conclusion as shown in Fig. 13.

For locally optimal Fréchet matchings (also called lexicographic matchings [15]), intuitively speaking, we need to minimize the use of large values. As the computed discrete matching is strongly locally correct, the maximum value in the definition does not consider common endpoints between compared subpaths. This hints at decreasing distances as fast as possible, but this is done very locally and in so far that the preference order permits this. That is, no distinction is made between paths between two grid nodes, depending on how often the maximal distance along the path is attained—this would be necessary for a locally optimal Fréchet matching. If we were to augment the tree with such information, this would again lead to breaking ties differently and thus an incorrect algorithm. Note that, if all distances in the grid are unique, then our algorithm indeed computes a locally optimal discrete Fréchet matching.

6. Conclusion

We set out to find "good" matchings between two curves. To this end we introduced the local correctness criterion for Fréchet matchings. We have proven that there always exists at least one locally correct Fréchet matching between any two polygonal curves. We translated this proof into an $O(N^3 \log N)$ algorithm, where N is the total number of edges in the two curves. Furthermore, we considered computing a locally correct matching using the discrete Fréchet distance. By maintaining a tree with shortcuts to encode locally correct partial matchings, we have shown how to compute such a matching in $O(N^2)$ time. In other words, the overhead is subpolynomial compared to the best known algorithm [12] and there is no asymptotic overhead with the standard dynamic program to compute the discrete Fréchet distance. Throughout, we paid particular attention to dealing with degeneracy of the input curves that cause multiple events to coincide at the same value of ε in the continuous case and at equal distances between pairs of vertices in the discrete case.

Future work Computing a locally correct discrete Fréchet matching with our algorithm takes $O(N^2)$ time, just like the basic dynamic program to compute only the discrete Fréchet distance. Recently, Agarwal et al. [12] have shown how to compute the discrete Fréchet distance in $O(N^2 \log\log N / \log N)$ time. This raises the question whether a similar improvement can be made to compute a locally correct discrete Fréchet matching.

Our algorithm for computing a locally correct (continuous) Fréchet matching takes $O(N^3 \log N)$ time, approximately a linear factor more than computing the Fréchet distance. An interesting question is whether this gap in computation can be reduced, as we have shown is possible for the discrete case. Our algorithm for discrete matchings constructs a tree of locally correct paths in a single sweep of the parameter space. This suggests that going away from the "decision-and-search" paradigm for the continuous case may yield improvements here. Recently, Buchin et al. [17] have given such an algorithm that computes the Fréchet distance in $O(N^2 \log^2 N)$ time. However, proceeding in the exact same way as for the discrete case is not feasible: much of the information in the algorithm of Buchin et al. is maintained implicitly, to be constructed and retrieved only when it is actually needed. Moreover, a single cell boundary does not have a unique structure for the locally correct Fréchet matchings end there. Rather, the matchings ending at a cell may in the worst case fall into at least linearly many, structurally different categories. As a result, the tree encoding the various matchings might become too big to maintain efficiently. On the other hand, we may be able to significantly gain in terms of speed, if we are able to avoid recomputing the Fréchet distance on the subcurves of the input. Buchin et al. also apply their techniques to use polyhedral distances and use this to approximate the Euclidean case. This leads us to pose the question whether polyhedral distances also simplify the computation of locally correct Fréchet matchings, and how we may approach defining and computing approximate locally correct Fréchet matchings.

Finally, it would be interesting to further investigate criteria of restricting matchings. On the one hand, this includes further restrictions to locally correct Fréchet matchings, such as length and matched-distances constraints (as discussed at the end of Section 4 and Section 5). For example, can we compute the locally optimal (or lexicographic) discrete Fréchet matching in $O(N^2)$ time, even if pairs of points have the same distance? On the other hand, this also includes the benefit of local correctness for other matching-based similarity measures, such as the geodesic width [4].

Acknowledgements

The authors would like to thank Carola Wenk for helpful discussions on the topic of this paper.

References

[1] K. Buchin, M. Buchin, W. Meulemans, B. Speckmann, Locally correct Fréchet matchings, in: Proceedings of the 20th European Symposium on Algorithms, in: LNCS, vol. 7501, 2012, pp. 229–240.
[2] W. Meulemans, Similarity Measures and Algorithms for Cartographic Schematization, Ph.D. thesis, Technische Universiteit Eindhoven, 2014.
[3] C. Wenk, R. Salas, D. Pfoser, Addressing the need for map-matching speed: localizing global curve-matching algorithms, in: Proceedings of the 18th International Conference on Scientific and Statistical Database Management, 2006, pp. 379–388.
[4] A. Efrat, L. Guibas, S. Har-Peled, J. Mitchell, T. Murali, New similarity measures between polylines with applications to morphing and polygon sweeping, Discrete Comput. Geom. 28 (4) (2002) 535–569.
[5] T. Wylie, B. Zhu, A polynomial time solution for protein chain pair simplification under the discrete Fréchet distance, in: Proceedings of the International Symposium on Bioinformatics Research and Applications, in: LNBI, vol. 7292, 2012, pp. 287–298.
[6] K. Buchin, M. Buchin, J. Gudmundsson, M. Löffler, J. Luo, Detecting commuting patterns by clustering subtrajectories, Int. J. Comput. Geom. Appl. 21 (3) (2011) 253–282.
[7] M. Konzack, T. McKetterick, T. Ophelders, M. Buchin, L. Giuggioli, J. Long, T. Nelson, M.A. Westenberg, K. Buchin, Visual analytics of delays and interaction in movement data, Int. J. Geogr. Inf. Sci. 31 (2) (2017) 320–345.
[8] H. Alt, M. Godau, Computing the Fréchet distance between two polygonal curves, Int. J. Comput. Geom. Appl. 5 (1) (1995) 75–91.
[9] S. Har-Peled, B. Raichel, Fréchet distance revisited and extended, ACM Trans. Algorithms 10 (1) (2014) 3.
[10] T. Eiter, H. Mannila, Computing Discrete Fréchet Distance, Tech. Rep. CD-TR 94/65, Christian Doppler Laboratory, 1994.
[11] K. Buchin, M. Buchin, W. Meulemans, W. Mulzer, Four Soviets walk the dog: improved bounds for computing the Fréchet distance, Discrete Comput. Geom. 58 (1) (2017) 180–216.
[12] P.K. Agarwal, R.B. Avraham, H. Kaplan, M. Sharir, Computing the discrete Fréchet distance in subquadratic time, SIAM J. Comput. 43 (2) (2014) 429–449.
[13] A. Maheshwari, J. Sack, K. Shahbaz, H. Zarrabi-Zadeh, Fréchet distance with speed limits, Comput. Geom. Theory Appl. 44 (2) (2011) 110–120.
[14] K. Buchin, M. Buchin, J. Gudmundsson, Constrained free space diagrams: a tool for trajectory analysis, Int. J. Geogr. Inf. Sci. 24 (7) (2010) 1101–1125.
[15] G. Rote, Lexicographic Fréchet matchings, in: Abstracts of the 30th European Workshop on Computational Geometry, 2014.
[16] A. Maheshwari, J. Sack, C. Scheffer, Approximating the integral Fréchet distance, Comput. Geom. Theory Appl. 70–71 (2018) 13–30.
[17] K. Buchin, M. Buchin, R. van Leusden, W. Meulemans, W. Mulzer, Computing the Fréchet distance with a retractable leash, Discrete Comput. Geom. 56 (2) (2016) 315–336.