TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Locally correct Fréchet matchings

Citation for published version (APA):
Buchin, K., Buchin, M., Meulemans, W., & Speckmann, B. (2012). Locally correct Fréchet matchings. In L. Epstein, & P. Ferragina (Eds.), *20th European Symposium on Algorithms (ESA)* (pp. 229-240). (Lecture Notes in Computer Science; Vol. 7501). Springer. https://doi.org/10.1007/978-3-642-33090-2_21

DOI:
10.1007/978-3-642-33090-2_21

Document status and date:
Published: 01/01/2012

Document Version:
Accepted manuscript including changes made at the peer-review stage

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Download date: 30. Mar. 2023

Locally Correct Fréchet Matchings*

Kevin Buchin, Maike Buchin, Wouter Meulemans, and Bettina Speckmann

Dep. of Mathematics and Computer Science, TU Eindhoven, The Netherlands.
k.a.buchin@tue.nl m.e.buchin@tue.nl w.meulemans@tue.nl
speckman@win.tue.nl

Abstract. The Fréchet distance is a metric to compare two curves, which is based on monotonous 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 total 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. However, the Hausdorff and the Fréchet distance are both based on matchings of the points on the curves. The distance returned is the maximum distance between any two matched points. The Fréchet distance uses *monotonous 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 monotonous matchings of the curves. Restricting to monotonous 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*. See Section 2 for more details.

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 well (see Fig. 1). There are applications that directly use a matching, for example, to map a GPS track to

* M. Buchin is supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 612.001.106. W. Meulemans and B. Speckmann are supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 639.022.707. A short abstract of the results presented in Section 3 and Section 4 appeared at the informal workshop EuroCG 2012. For omitted proofs we refer to [4].

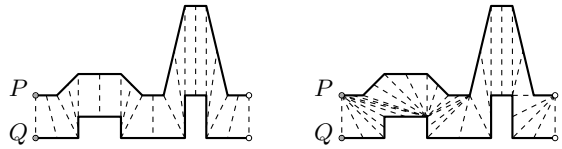

Fig. 1. Two Fréchet matchings for curves P and Q.

a street network [9] or to morph between the curves [5]. In such situations a "good" matching is important. Furthermore, we believe that many applications of the (discrete) Fréchet distance, such as protein alignment [10] and detecting patterns in movement data [3], would profit from good Fréchet matchings.

Results. We restrict the set of Fréchet matchings to "natural" matchings by introducing *locally correct* Fréchet matchings: matchings that for any two matched subcurves are again 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 $O(N^3 \log N)$ algorithm to compute such a matching, 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.

Related work. The first algorithm to compute the Fréchet distance was given by Alt and Godau [1]. They also consider a non-monotone Fréchet distance and their algorithm for this variant results in a locally correct non-monotone matching (see Remark 3.5 in [7]). Eiter and Mannila gave the first algorithm to compute the discrete Fréchet distance [6]. Since then, the Fréchet distance has received significant attention. Here we focus on approaches that restrict the allowed matchings. Efrat *et al.* [5] introduced Fréchet-like metrics, the geodesic width and link width, to restrict to matchings suitable for curve morphing. Their method is suitable only for non-intersecting 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.* [8] 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.* [2] 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.

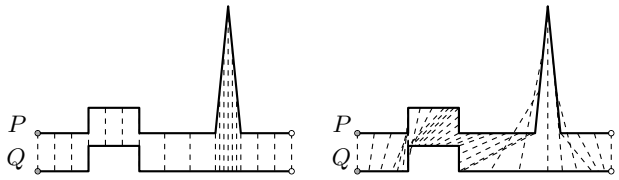

Fig. 2. Two Fréchet matchings. Right: the result of speed limits is not locally correct.

2 Preliminaries

Curves. Let P be a polygonal curve with m edges, defined by vertices $p_0, \ldots, p_m$. We treat a curve as a continuous map $P : [0, m] \to \mathbb{R}^d$. In this map, $P(i)$ equals p_i for integer i. Furthermore, $P(i+\lambda)$ is a parameterization 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, non-decreasing 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 (monotonous) *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: $d_\mu[0, 1] = \delta_{\mathrm{F}}(P, Q)$ holds.

Free space diagrams. Alt and Godau [1] describe an algorithm to compute the Fréchet distance based on the decision variant (that is, solving $\delta_{\mathrm{F}}(P, Q) \le \varepsilon$ for some given ε). Their algorithm uses a *free space diagram*, a two-dimensional diagram on the range $[0, m] \times [0, n]$. Every point (x, y) in this diagram is either "free" (white) or not (indicating whether $|P(x) - Q(y)| \le \varepsilon$). The diagram has m columns and n rows; every cell (c, r) $(1 \le c \le m$ and $1 \le r \le n)$ corresponds to the edges $p_{c-1}p_c$ and $q_{r-1}q_r$. 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. To compute the Fréchet distance, one finds the smallest ε such that there exists an x- and y-monotone path from point $(0, 0)$ to (m, n) in free space. For this, only certain *critical values* for the distance have to be checked. Imagine continuously increasing the distance ε starting at $\varepsilon = 0$. At so-called *critical events*, which are illustrated in Fig. 3, passages open in the free space. The critical values are the distances corresponding to these events.

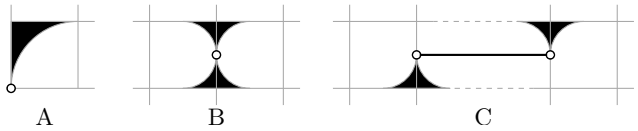

Fig. 3. Three event types. (A) Endpoints come within range of each other. (B) Passage opens on cell boundary. (C) Passage opens in row (or column).

3 Locally correct Fréchet matchings

We introduce *locally correct* Fréchet matchings, for which the matching between any two matched subcurves is a Fréchet matching.

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 $0 \leq a \leq b \leq 1$*

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

Note that not every Fréchet matching is locally correct. See for example Fig. 2. The question arises whether a locally correct matching always exists and if so, how to compute it. We resolve the first question in the following theorem.

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

Existence. We prove Theorem 1 by induction on the number of edges in the curves. The proofs for the lemmata of this section have been omitted but can be found in [4]. First, we present the lemmata for the two base cases: one of the two curves is a point, and both curves are line segments. In the following, n and m again denote the number of edges of P and Q, respectively.

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

Lemma 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 induction, we split the two curves based on events (see Fig. 4). Since each split must reduce the problem size, we ignore any events on the left or bottom boundary of cell $(1, 1)$ or on the right or top boundary of cell (m, n).

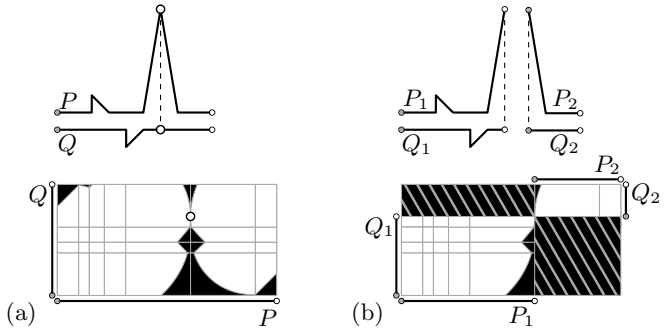

Fig. 4. (a) Curves with the free space diagram for $\varepsilon = \delta_F(P, Q)$ and the realizing event. (b) The event splits each curve into two subcurves. The hatched areas indicate parts that disappear after the split.

This excludes both events of type A. A free space diagram is *connected* at value ε, if a monotonous path exists from the boundary of cell $(1,1)$ to the boundary of cell (m,n). A *realizing event* is a critical event at the minimal value ε such that the corresponding free space diagram is connected.

Let $\mathcal{E}$ denote the set of concurrent realizing events for two curves. A *realizing set* E_r is a subset of $\mathcal{E}$ such that the free space admits a monotonous path from cell $(1,1)$ to cell (m,n) without using an event in $\mathcal{E}\backslash E_\mathrm{r}$. Note that a realizing set cannot be empty. 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.

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

The following lemma directly implies that a locally correct Fréchet matching always exists. Informally, it states that curves have a locally correct matching that is "closer" (except in cell $(1,1)$ or (m,n)) than the distance of their realizing set. Further, this matching is linear inside every cell. In the remainder, we use realizing set to indicate a minimal realizing set, unless indicated otherwise. Now, the following lemma can be proven by induction: a realizing set is used to split the curves into pieces; combining the locally correct matchings of the pieces results in a single, locally correct matching.

Lemma 4. *If the free space diagram of two polygonal curves P and Q is connected at value ε, then there exists a locally correct Fréchet matching $\mu = (\sigma, \theta)$ such that $d_\mu(t) \leq \varepsilon$ for all t with $\sigma(t) \geq 1$ or $\theta(t) \geq 1$, and $\sigma(t) \leq m-1$ or $\theta(t) \leq n-1$. Furthermore, μ is linear in every cell.*

4 Algorithm for locally correct Fréchet matchings

The existence proof directly results in a recursive algorithm, which is given by Algorithm 1. Fig. 1 (left), Fig. 2 (left), Fig. 5, Fig. 6, and Fig. 7 (left) illustrate matchings computed with our algorithm. This section is devoted to proving 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.*

Using the notation of Alt and Godau [1], $L^F_{i,j}$ denotes the interval of free space on the left boundary of cell (i,j); $L^R_{i,j}$ denotes the subset of $L^F_{i,j}$ that is reachable from point $(0,0)$ of the free space diagram with a monotonous path in the free space. Analogously, $B^F_{i,j}$ and $B^R_{i,j}$ are defined for the bottom boundary.

With a slight modification to the decision algorithm, we can compute the minimal value of ε such that a path is available from cell $(1,1)$ to cell (m,n). This requires only two changes: $B^R_{1,2}$ should be initialized with $B^F_{1,2}$ and $L^R_{2,1}$ with $L^F_{2,1}$; the answer should be "yes" if and only if $B^R_{m,n}$ or $L^R_{m,n}$ is non-empty.

Algorithm 1 `ComputeLCFM`(P, Q)

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

if $m = 0$ **or** $n = 0$ **then**
  return (σ, θ) where $\sigma(t) = t \cdot m$, $\theta(t) = t \cdot n$
else if $m = n = 1$ **then**
  return (σ, θ) where $\sigma(t) = \theta(t) = t$
else
  Find event e_r of a minimal realizing set
  Split P into P_1 and P_2 according to e_r
  Split Q into Q_1 and Q_2 according to e_r
  $\mu_1 \to$ `ComputeLCFM`(P_1, Q_1)
  $\mu_2 \to$ `ComputeLCFM`(P_2, Q_2)
  return concatenation of μ_1, e_r, and μ_2

Realizing set. By computing the Fréchet distance using the modified Alt and Godau algorithm, we obtain an ordered, potentially non-minimal realizing set $\mathcal{E} = \{e_1, \ldots, e_l\}$. The algorithm must find an event that is contained in a realizing set. Let E_k denote the first k events of $\mathcal{E}$. For now we assume that the events in $\mathcal{E}$ end at different cell boundaries. We use a binary search on $\mathcal{E}$ 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 realizing set and can be used to split the curves. Note that r is unique due to monotonicity. For correctness, the order of events in $\mathcal{E}$ must be consistent in different iterations, for example, by using a lexicographic order. Set E_r contains only realizing sets that use e_r. Hence, E_{r-1} contains a realizing set to connect cell $(1, 1)$ to e_r and e_r to cell (m, n). Thus any event found in subsequent iterations is part of E_{r-1} and of a realizing set with e_r.

To determine whether some E_k contains a realizing set, we check whether cells $(1, 1)$ and (m, n) are connected without "using" the events of $\mathcal{E} \backslash E_k$. To do this efficiently, we further modify the Alt and Godau decision algorithm. We require only a method to prevent events in $\mathcal{E} \backslash E_k$ from being used. After $L^R_{i,j}$ is computed, we check whether the event e (if any) that ends at the left boundary of cell (i, j) is part of $\mathcal{E} \backslash E_k$ and necessary to obtain $L^R_{i,j}$. If this is the case, we replace $L^R_{i,j}$ with an empty interval. Event e is necessary if and only if $L^R_{i,j}$ is a singleton. To obtain an algorithm that is numerically more stable, we introduce

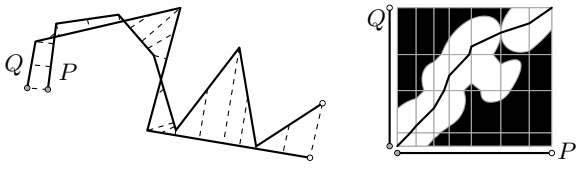

Fig. 5. Locally correct matching produced by Algorithm 1. Free space diagram drawn at $\varepsilon = \delta_\mathrm{F}(P, Q)$.

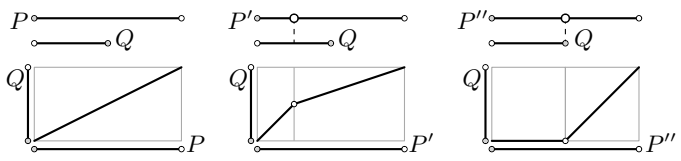

Fig. 6. Different sampling may result in different matchings.

entry points. The *entry point* of the left boundary of cell (i, j) is the maximal $i' < i$ such that $B^R_{i',j}$ is non-empty. These values are easily computed during the decision algorithm. Assume the passage corresponding to event e starts on the left boundary of cell (i_s, j). Event e is necessary to obtain $L^R_{i,j}$ if and only if $i' < i_s$. Therefore, we use the entry point instead of checking whether $L^R_{i,j}$ is a singleton. This process is analogous for horizontal boundaries of cells.

Earlier we assumed that each event in $\mathcal{E}$ ends at a different cell boundary. 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). This justifies the assumption and ensures that $\mathcal{E}$ contains $O(mn)$ events. Thus computing e_r (Algorithm 1, line 6) takes $O(mn \log(mn))$ time, which is equal to the time needed to compute the Fréchet distance. Each recursion step splits the problem into two smaller problems, and the recursion ends when $mn \leq 1$. This results in an additional factor $m + n$. Thus the overall running time is $O((m + n)mn \log(mn))$.

Sampling and further restrictions. Two curves may still have many locally correct Fréchet matchings: the algorithm computes just one of these. However, introducing extra vertices may alter the result, even if these vertices do not modify the shape (see Fig. 6). 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. Based on this idea, we are currently investigating "locally optimal" Fréchet matchings. The idea is to restrict to the locally correct Fréchet matching that decreases the matched distance as quickly as possible.

We also considered restricting to the "shortest" locally correct Fréchet matching, where "short" refers to the length of the path in the free space diagram. However, Fig. 7 shows that such a restriction does not necessarily improve the quality of the matching.

5 Locally correct discrete Fréchet matchings

Here we study the discrete variant of Fréchet matchings. For the discrete Fréchet distance, only the vertices of curves are matched. The discrete Fréchet distance can be computed in $O(m \cdot n)$ time via dynamic programming [6]. Here, we show how to also compute a locally correct discrete Fréchet matching in $O(m \cdot n)$ time.

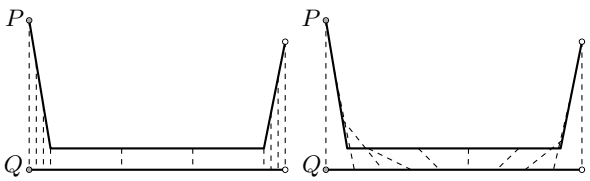

Fig. 7. Two locally correct Fréchet matchings for P and Q. Right: shortest matching.

Grids. Since we are interested only in matching vertices of the curves, we can convert the problem to a grid problem. Suppose we have two curves P and Q with m and n edges respectively. These convert into a grid G of non-negative 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 monotonous 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 monotonous path unless indicated otherwise. A monotonous discrete matching of the curves corresponds to a path π such that $\pi(1) = G[0,0]$ and $\pi(k) = G[m,n]$. We call a path π 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 $\pi'(k') = \pi(t_2)$.

Algorithm. The algorithm needs to compute a locally correct path between $G[0,0]$ and $G[m,n]$ in a grid G of non-negative values. To this end, the algorithm incrementally constructs a tree T on the grid such that each path in T is locally correct. The algorithm is summarized by Algorithm 2. We define a *growth node* as a node of T that has a neighbor in the grid that is not yet

Algorithm 2 `ComputeDiscreteLCFM`(P, Q)

Require: P and Q are curves with m and n edges
Ensure: A locally correct discrete Fréchet matching for P and Q

```
Construct grid G for P and Q
Let T be a tree consisting only of the root G[0,0]
for i ← 1 to m do
    Add G[i,0] to T
for j ← 1 to n do
    Add G[0,j] to T
for i ← 1 to m do
    for j ← 1 to n do
        AddToTree(T, G, i, j)
return path in T between G[0,0] and G[m,n]
```

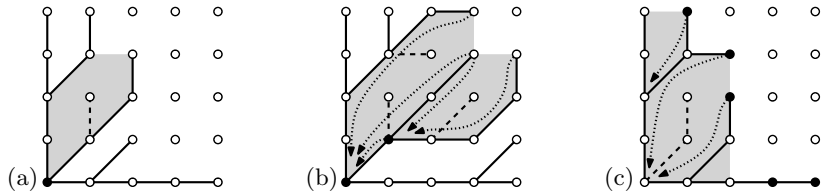

Fig. 8. (a) Face of tree (gray area) with its unique sink (solid dot). A dashed line represents a dead path. (b) Two adjacent faces with some shortcuts indicated. (c) Tree with 3 faces. Solid dots indicate growth nodes with a growth node as parent. These nodes are incident to at most one face. All shortcuts of these nodes are indicated.

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. Every pair of nodes in this tree has a *nearest common ancestor* (NCA). When we have to decide what parent to use for a new node in the tree, we look at the maximum value on the path in the tree between the parents and their NCA (excluding the value of the latter). A *face* of the tree 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 NCA. 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. 8 (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 sink of these faces, associating the maximum value encountered on the path between the node and the sink (excluding the value of the sink). Fig. 8 (b) illustrates some shortcuts. With these shortcuts, it is possible to determine the maximum up to the NCA of two (potentially diagonally) neighboring growth nodes in constant 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. 8 (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 NCA. 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 NCA is at most the maximum value from c' to their NCA—both excluding the NCA itself. We must be consistent when breaking ties between candidate parents. To this end, we use

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

Require: G is a grid of non-negative values; any path in tree T is locally correct
Ensure: 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 NCA
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

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.

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. 9 illustrates the incoming shortcuts at a sink and the effect of removing a dead path on the incoming shortcuts. Note that 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.

Correctness. To prove correctness of Algorithm 2, we require a stronger version of local correctness. A path π is *strongly locally correct* if for all paths π' with the same endpoints $\max_{1<t\leq k} \pi(t) \leq \max_{1<t'\leq k'} \pi'(t')$ holds. Note that the first node is excluded from the maximum. Since $\max_{1<t\leq k} \pi(t) \leq \max_{1<t'\leq k'} \pi'(t')$ and $\pi(1) = \pi'(1)$ imply $\max_{1\leq t\leq k} \pi(t) \leq \max_{1\leq t'\leq k'} \pi'(t')$, a strongly locally correct path is also locally correct. Lemma 5 implies the correctness of Algorithm 2. We only sketch the proof here. For the full proof see [4].

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

Proof (sketch). Initially any path in T spans either only the first row or only the first column. Hence, the path is unique between its endpoints and this path is strongly locally correct.

The algorithm extends tree T to T' by including a node g that has three candidate parents which are growth nodes. From the invariant, we derive that

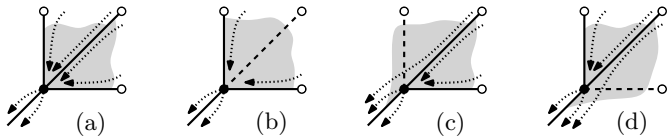

Fig. 9. (a) Each sink has up to four sets of shortcuts. (b-d) Removing a dead path (dashed) extends at most one set of shortcuts.

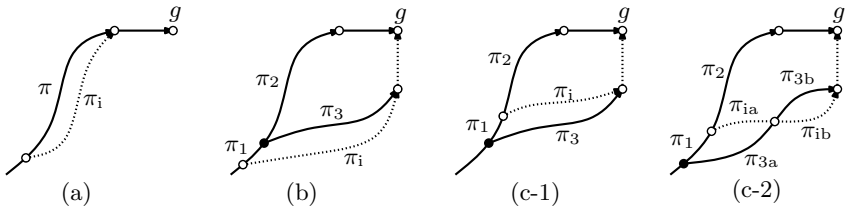

Fig. 10. The four cases for the proof of Lemma 5.

if a path in T' is not strongly locally correct, then an invalidating path exists that ends at g and starts at an ancestor of g. The main observation is that this invalidating path uses one of the candidate parents of g as its before-last node. We distinguish three cases on how this path is situated compared to T'. The last case, however, needs two subcases in order to deal with candidate parents that have the same maximum value to their NCA. Fig. 10 illustrates these cases. For the first three cases, we derive that there is a path in T that is not strongly locally correct, contradicting the induction hypothesis. These paths are π, (π_1, π_3), and π_3 respectively. For case (c-2), either π_{3a} is not strongly locally correct or (π_1, π_{ia}) conflicts with the preference order. □

Execution time. 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 omitted proof can be found in [4]. The main observation is that the path requiring extensions starts at $\pi_d(1)$ and ends at either $G[i-1, j]$ 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 can result in at most $2 \cdot k - 1$ extensions.*

Hence, we can charge every extension to one of the $k - 1$ dead nodes (all but $\pi_d(1)$). A node gets at most 3 charges, since it is a (non-first) node of a dead path at most once. Because an extension can be done in constant time, the execution time of the algorithm is $O(mn)$. Note that shortcuts that originate from a living node with outdegree 1 could be removed instead of extended. 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.*

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. This proof resulted in 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.

Future work. Computing a locally correct discrete Fréchet matching takes $O(N^2)$ time, just like the dynamic program to compute only the discrete Fréchet distance. However, computing a locally correct continuous Fréchet matching takes $O(N^3 \log N)$ time, a linear factor more than computing the Fréchet distance. An interesting question is whether this gap in computation can be reduced as well.

Furthermore, it would be interesting to investigate the benefit of local correctness for other matching-based similarity measures, such as the geodesic width [5].

References

1. H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *Int. J. of Comp. Geometry and Appl.*, 5(1):75–91, 1995.
2. K. Buchin, M. Buchin, and J. Gudmundsson. Constrained free space diagrams: a tool for trajectory analysis. *Int. J. of GIS*, 24(7):1101–1125, 2010.
3. K. Buchin, M. Buchin, J. Gudmundsson, M. Löffler, and J. Luo. Detecting Commuting Patterns by Clustering Subtrajectories. *Int. J. of Comp. Geometry and Appl.*, 21(3):253–282, 2011.
4. K. Buchin, M. Buchin, W. Meulemans, and B. Speckmann. Locally correct Fréchet matchings. *CoRR*, abs/1206.6257, June 2012.
5. A. Efrat, L. Guibas, S. Har-Peled, J. Mitchell, and T. Murali. New Similarity Measures between Polylines with Applications to Morphing and Polygon Sweeping. *Discrete & Comp. Geometry*, 28(4):535–569, 2002.
6. T. Eiter and H. Mannila. Computing Discrete Fréchet Distance. Technical Report CD-TR 94/65, Christian Doppler Laboratory, 1994.
7. S. Har-Peled and B. Raichel. Fréchet distance revisited and extended. *CoRR*, abs/1202.5610, Feb. 2012.
8. A. Maheshwari, J. Sack, K. Shahbaz, and H. Zarrabi-Zadeh. Fréchet distance with speed limits. *Comp. Geometry: Theory and Appl.*, 44(2):110–120, 2011.
9. C. Wenk, R. Salas, and D. Pfoser. Addressing the need for map-matching speed: Localizing global curve-matching algorithms. In *Proc. 18th Int. Conf. on Sci. and Stat. Database Management*, pages 379–388, 2006.
10. T. Wylie and B. Zhu. A polynomial time solution for protein chain pair simplification under the discrete Fréchet distance. In *Proc. Int. Symp. on Bioinformatics Research and Appl.*, pages 287–298, 2012.