International Journal of Computational Geometry & Applications
Vol. 5, Nos. 1 & 2 (1995) 75–91

COMPUTING THE FRÉCHET DISTANCE BETWEEN TWO POLYGONAL CURVES *

HELMUT ALT and MICHAEL GODAU
Freie Universität Berlin, Fachbereich Mathematik und Informatik, Takustraße 9
14195 Berlin, Germany
e-mail: alt@inf.fu-berlin.de

Received 2 November 1992
Revised 10 October 1993
Communicated by F. Aurenhammer

ABSTRACT

As a measure for the resemblance of curves in arbitrary dimensions we consider the so-called *Fréchet-distance*, which is compatible with parametrizations of the curves. For polygonal chains P and Q consisting of p and q edges an algorithm of runtime $\mathcal{O}(pq\log(pq))$ measuring the Fréchet-distance between P and Q is developed. Then some important variants are considered, namely the Fréchet-distance for closed curves, the nonmonotone Fréchet-distance and a distance function derived from the Fréchet-distance measuring whether P resembles some *part* of the curve Q.

Keywords: Fréchet-distance, shape analysis, resemblance of curves, Computational Morphology.

1. Introduction, Definitions

In many applications, two-dimensional *shapes* are given by the planar curves forming their boundaries. Consequently, a natural problem in shape comparison and recognition is to measure, how much two given curves "resemble each other". Naturally, the first question to be answered, is what distance measure between curves should be used to reflect the intuitive notion of "resemblance".

One possibility is the so-called *Hausdorff-metric* δ_H, which for arbitrary bounded sets $A, B \subseteq \mathbb{R}^2$ is defined as follows:

$$\delta_H(A,B) = \max\left(\sup_{a\in A}\inf_{b\in B} d(a,b), \sup_{b\in B}\inf_{a\in A} d(a,b)\right)$$

where d is the underlying metric in the plane, for example the Euclidean metric. Algorithms were developed for determining the Hausdorff-distance between polygonal curves (even for arbitrary finite sets of line segments), also for the case that one

*This research was supported by Deutsche Forschungsgemeinschaft under Grant No. Al 253/1-3, SPP "Datenstrukturen und effiziente Algorithmen"

of the curves can be moved by some rigid motion to "match" the other one as closely as possible.[1] While in many applications the Hausdorff-distance is an appropriate measure, Figure 1 shows an example, where it is not. The two curves have a small Hausdorff-distance, but do not "resemble" each other at all.

Fig. 1. Two curves with Hausdorff-distance δ.

The reason for this discrepancy is that the Hausdorff-distance only takes into account the sets of points on both curves and does not reflect the course of the curves. In many applications, however, the course of curves is important. One example are curves which are input by a digitizer such as handwriting, where a parametrization of the curve is already given by the input device. Also if curves are to be approximated by simpler ones like, for example, curves representing rivers or borderlines in cartography, the course is significant and should be reflected by the metric measuring the quality of the approximation. In order to overcome this discrepancy we will here consider an alternative metric, the definition of which is compatible with orientation-preserving reparametrizations of the curves. It was first defined by Fréchet,[2,3] and is known as the *Fréchet-metric* δ_F.

A popular illustration of the Fréchet-metric is the following: Suppose a man is walking his dog, he is walking on the one curve the dog on the other. Both are allowed to control their speed but are not allowed to go backwards. Then the Fréchet-distance of the curves is the minimal length of a leash that is necessary.

The concept of 2-dimensional space is neither important for the following definition nor for the algorithms described by us, in fact they will work for polygonal curves in arbitrary dimensions. Therefore *let V denote in the following an arbitrary Euclidean vector space.*

Definition 1 *A* curve *is a continuous mapping $f: [a,b] \to V$ with $a, b \in \mathbb{R}$ and $a < b$. A* polygonal curve *is a curve $P: [0,n] \to V$ with $n \in \mathbb{N}$, such that for all $i \in \{0,1,\ldots,n-1\}$ each $P_{|[i,i+1]}$ is affine, i.e. $P(i+\lambda) = (1-\lambda)P(i) + \lambda P(i+1)$ for all $\lambda \in [0,1]$. n is called the* length *of P.*

Definition 2 *Let $f: [a,a'] \to V$ and $g: [b,b'] \to V$ be curves. Then $\delta_F(f,g)$ denotes their* Fréchet-distance, *defined as*

$$\delta_F(f,g) := \inf_{\substack{\alpha:[0,1]\to[a,a'] \\ \beta:[0,1]\to[b,b']}} \max_{t\in[0,1]} ||f(\alpha(t)) - g(\beta(t))||.$$

where α, β range over continuous and increasing functions with $\alpha(0) = a$, $\alpha(1) = a'$, $\beta(0) = b$ and $\beta(1) = b'$ only.

δ_F is obviously symmetric and it can be shown that the triangle inequality holds.[3] Moreover δ_F is invariant under orientation-preserving reparametrizations. By calling two curves equivalent iff their distance is zero then regarding the equivalence classes as the "true" curves — the so-called Fréchet-curves — and defining δ_F appropriately we can realize δ_F as a metric.

This paper whose results already appeared in preliminary form [4,5,6] is concerned with computing the Fréchet-distance for given polygonal curves P and Q. In Section 2 we will first consider an easier variant of the problem, namely the following *decision problem*:

Given: polygonal curves P and Q and some $\varepsilon \geqslant 0$.
Decide, whether $\delta_F(P, Q) \leqslant \varepsilon$.

An algorithm of runtime $\mathcal{O}(pq)$ where p and q are the lengths of P and Q, respectively, solving this decision problem will be given in Section 2.

The problem of actually *computing* $\delta_F(P, Q)$ for given polygonal curves P, Q will be solved by an algorithm of runtime $\mathcal{O}(pq \log pq)$ which makes use of the decision algorithm and the technique of *parametric search* and which improves upon the $\mathcal{O}((pq^2 + p^2q)\log(pq))$ algorithm from one of our previous papers.[4]

In Section 3 we will investigate variants of the Fréchet-metric. One is obtained by dropping the monotonicity condition, the other one by allowing arbitrary starting points in the case of closed curves. Algorithms for both variants will be developed. Additionally, we obtain an algorithm for solving the partial matching problem, i.e. given curves P and Q, find that part of Q to which P has the smallest Fréchet-distance.

Our *model of computation* is a random access machine doing arithmetic operations $+, -, \times, /$ and square roots in constant time. Although we describe and illustrate all problems and algorithms for polygonal curves in $\mathbb{R}^2$ with Euclidean distance, all algorithms work for polygonal curves in $\mathbb{R}^n$ for arbitrary but fixed n and for the L_1- and L_∞-metrics, as well. For L_1 and L_∞ it even suffices that our model of computation can do integer operations $+, -, \times$ in constant time, if we assume that the input consists of rational numbers and rationals are represented as fractions. (In principle our algorithm also works for L_k, $k \neq 1, 2, \infty$, but then the model of computation must be able to solve algebraic equations of some higher (but fixed) degree in constant time). In the sequel we will denote the underlying space by V, meaning $\mathbb{R}^n$ for some $n \geqslant 2$, and the metric by $d(.,.)$ meaning L_1, L_2, or L_∞.

It should be mentioned, that Natarajan considers the same problems on both Hausdorff- and Fréchet-metric (there called "parametric distance") with respect to the L_∞-metric.[7] Our algorithms improve upon the runtime of the ones given there, which are $\mathcal{O}\left([\min(p, q)]^2 \max(p, q)\right)$ for the decision problem and $\mathcal{O}\left([\min(p, q)]^2 \max(p, q) \log(1/\delta)\right)$ for the computation problem where δ is the precision to which $\delta_F(P, Q)$ is computed.

A problem that we do not address here, but which is important in connection with the Fréchet-metric is the *approximation* of curves by simpler ones. More pre-

cisely, given a polygonal curve P and a tolerance bound $\varepsilon > 0$, find a minimum link chain Q with $\delta_F(P,Q) < \varepsilon$. This approximation problem has already been investigated.[4,8]

2. Computing the Fréchet-distance

Throughout the rest of this paper let $P: [0,p] \to V$ and $Q: [0,q] \to V$ be polygonal curves. Therefore p and q are the numbers of edges of P and Q, respectively.

In order to solve the decision problem let us first consider the case that $p = q = 1$, i.e. P and Q are just simple line segments. Define $F_\varepsilon := \{(s,t) \in [0,1]^2 \mid d(P(s),Q(t)) \leqslant \varepsilon\}$. F_ε describes all pairs of points, one on P, one on Q, whose distance is at most ε. Figure 2 shows line segments P, Q, a distance $\varepsilon > 0$, and F_ε being the white area within the unit square; subsequently called the "free space".

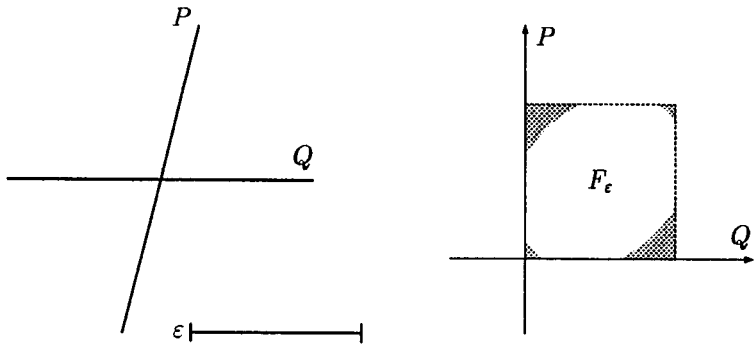

Fig. 2. P, Q, ε and F_ε

Lemma 3 *In the case of line segments P, Q, the free space F_ε is the intersection of the unit square with an ellipse (possibly degenerated to the space between two parallel lines) for L_2 (see Figure 2) and with a parallelogram for L_1 or L_∞. In any case, F_ε is convex.*

Proof: Extend P and Q from affine mappings over $[0,1]$ to affine mappings P', Q' over $\mathbb{R}$. Then the mapping $f: \mathbb{R}^2 \to \mathbb{R}^2$ defined by $f(s,t) = P'(s) - Q'(t)$ is affine, too. Thus, $F_\varepsilon = f^{-1}(D_\varepsilon) \cap [0,1]^2$, where D_ε is the set of all points with norm $\leqslant \varepsilon$. In the case of the L_2-norm D_ε is a disk, in the cases of the L_1- or the L_∞-norm it is a square, so its inverse image under an affine mapping is an ellipse or a parallelogram respectively. □

We extend F_ε to arbitrary polygonal curves P, Q of lengths p, q respectively:

$$F_\varepsilon := \{(s,t) \in [0,p] \times [0,q] \mid d(P(s),Q(t)) \leqslant \varepsilon\}.$$

Consider $[0,p] \times [0,q]$ as composed of the pq cells $C_{ij} := [i-1,i] \times [j-1,j]$ with $1 \leqslant i \leqslant p$, $1 \leqslant j \leqslant q$. Clearly, $F_\varepsilon \cap C_{ij}$ corresponds to the free space with respect to the edge $P(i-1)P(i)$ and the edge $Q(j-1)Q(j)$ according to the original definition. Figure 3 shows polygonal curves P, Q, a distance ε and the corresponding diagram of cells C_{ij} with the free space F_ε.

Our algorithms are based on the following straightforward observation:

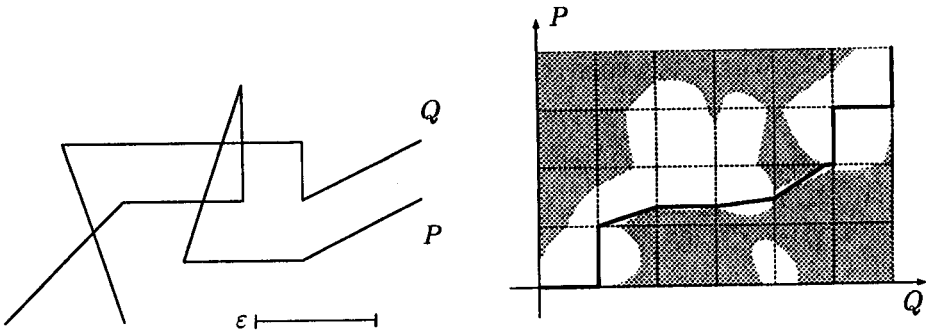

Fig. 3. Diagram for polygonal chains P, Q and the given ε

Lemma 4 *For polygonal curves P and Q we have $\delta_F(P,Q) \leqslant \varepsilon$, exactly if there exists a curve within the corresponding F_ε from $(0,0)$ to (p,q) which is monotone in both coordinates.*

Figure 3 shows such a curve proving that for that example $\delta_F(P,Q) \leqslant \varepsilon$. Observe that the curve as a continuous mapping from $[0,1]$ to $[0,p] \times [0,q]$ directly gives feasible reparametrizations α, β according to Definition 2.

For $(i,j) \in \{1,\ldots,p\} \times \{1,\ldots,q\}$ let L_{ij} (or B_{ij}) be the left (or bottom) line segment bounding cell C_{ij}. $L_{p+1,j}$ is the right line segment bounding C_{pj} and $B_{i,q+1}$ the upper line segment bounding C_{iq}. Let $L_{ij}^F := L_{ij} \cap F_\varepsilon$ and $B_{ij}^F := B_{ij} \cap F_\varepsilon$. Because of the convexity of F_ε within a cell each L_{ij}^F is a line segment of the form $\{i-1\} \times [a_{ij}, b_{ij}]$. Likewise B_{ij}^F is of the form $[c_{ij}, d_{ij}] \times \{j-1\}$ (see Figure 4).

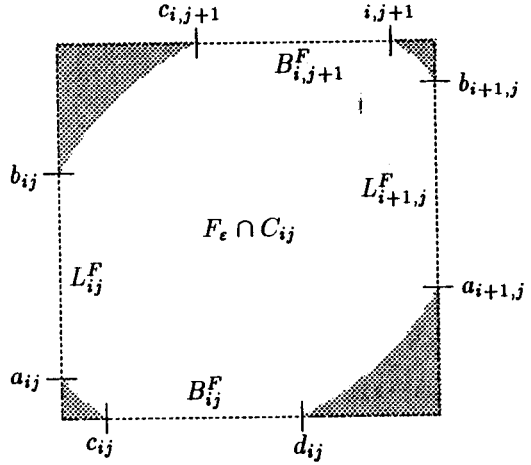

Fig. 4. Intervals of free space on the boundary of a cell.

Furthermore we define:
$R_\varepsilon = \{(s,t) \in F_\varepsilon \mid$ there exists a monotone curve within F_ε from $(0,0)$ to $(s,t)\}$, the set of points in the free space "reachable" from $(0,0)$.

Let $L_{ij}^R = L_{ij} \cap R_\varepsilon$ and $B_{ij}^R = B_{ij} \cap R_\varepsilon$ for all feasible i, j. By definition we have

$\delta_F(P,Q) \leqslant \varepsilon$, exactly if $(p,q) \in L^R_{p+1,q}$. It easily can be seen by induction that all nonempty L^R_{ij}, B^R_{ij} are line segments. Also, given L^R_{ij}, B^R_{ij}, $L^F_{i+1,j}$, and $B^F_{i,j+1}$ then $L^R_{i+1,j}$ and $B^R_{i,j+1}$ can easily be constructed in $\mathcal{O}(1)$ time. These considerations lead to the following algorithm for the decision problem:

Algorithm 1:

for each feasible pair (i,j) **do** compute L^F_{ij} and B^F_{ij};
for $i := 1$ **to** p **do** determine $B^R_{i,1}$;
for $j := 1$ **to** q **do** determine $L^R_{1,j}$;
for $i := 1$ **to** p **do**
  for $j := 1$ **to** q **do**
    construct $L^R_{i+1,j}$ and $B^R_{i,j+1}$ from L^R_{ij}, B^R_{ij}, $L^F_{i+1,j}$, $B^F_{i,j+1}$;
answer "yes" if $(p,q) \in L^R_{p+1,q}$ "no" otherwise.

It follows:

Theorem 5 *For given polygonal curves P, Q and $\varepsilon \geqslant 0$ Algorithm 1 decides in $\mathcal{O}(pq)$ time, whether $\delta_F(P,Q) \leqslant \varepsilon$.*

Next, let us consider the problem of really *computing* $\delta := \delta_F(P,Q)$. Assume that we start with $\varepsilon = 0$ and continuously increase ε. Then the free space F_ε becomes larger and larger and we want to determine the smallest ε such that it contains a monotone curve from $(0,0)$ to (p,q). Observe that this can occur only in one of the following cases:

a) ε is minimal with $(0,0) \in F_\varepsilon$ and $(p,q) \in F_\varepsilon$.

b) ε is minimal with L^F_{ij} or B^F_{ij} becomes nonempty for some pair (i,j). (A new passage opens between two neighboring cells.)

c) ε is minimal with $a_{ij} = b_{kj}$ or $c_{ij} = d_{ik}$ for some i,j,k (possibly, a new horizontal or vertical passage opens within the diagram, see Figure 5).

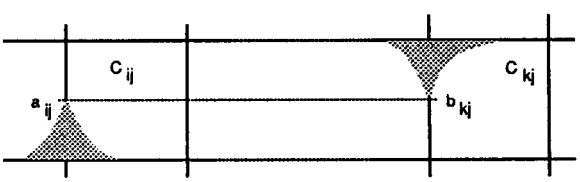

Fig. 5. A new horizontal passage in the j-th row of the diagram.

Figure 6 shows the geometric meaning of ε in cases a), b), c).

There are $\mathcal{O}(p^2q+pq^2)$ such "critical values" of ε, namely, the distances between starting points and endpoints of P and Q (case a)), the distances between vertices of one curve and edges of the other (case b)), and the common distance of two vertices of one curve to the intersection point of their bisector with some edge of the other (case c)). Each one of these values can be computed in $\mathcal{O}(1)$ time. So we obtain the following simple algorithm for computing $\delta_F(P,Q)$:

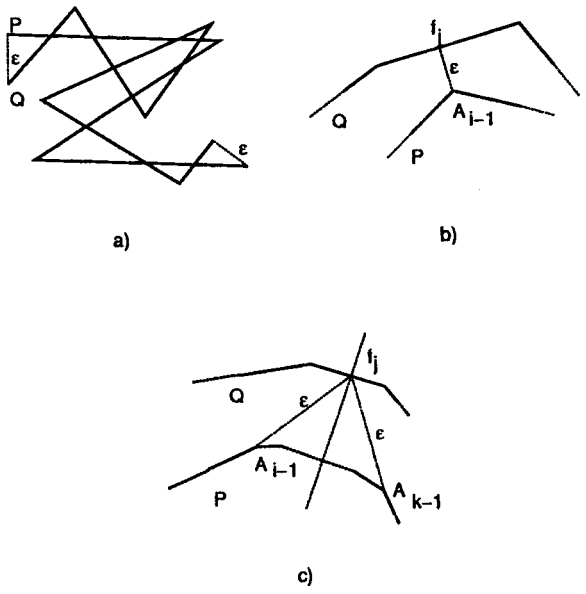

Fig. 6. Critical values of ε. $A_0, A_1, \ldots, A_p$ denote the vertices of P; $f_1, \ldots, f_q$ the edges of Q.

Algorithm 2:

1. Determine all critical values of ε.
2. Sort them.
3. Do a binary search on the sorted sequence in each search step solving the decision problem, continuing with the half containing smaller critical values if it has a positive answer and with the half containing larger critical values otherwise.

Observe that the runtime is majorized by the one of Step 2, which is $\mathcal{O}\left((p^2q + q^2p)\log(pq)\right)$.

We can obtain an asymptotically faster algorithm applying Meggido's technique of *parametric search*[9] on the critical values of type c). In order to do so, usually a parallel algorithm for the decision problem is used to direct the parametric search algorithm. However, as easily can be seen, it is sufficient to use any parallel algorithm whose critical values include the ones of the decision algorithm. Therefore, any parallel comparison-based *sorting* algorithm sorting all $\mathcal{O}(pq)$ values $a_{ij}, b_{ij}, c_{ij}, d_{ij}$ (which depend on ε) will work. In fact, since a critical ε of type c) occurs if $a_{ij} = b_{kj}$ or $c_{ij} = d_{ik}$ for some i, j, k, it is a critical value of the sorting algorithm, too. Furthermore, since a_{ij}, b_{ij}, c_{ij}, d_{ij} are algebraic functions in ε of constant degree, a comparison between two of them has only a constant number

of critical values. [a] Under these circumstances parametric search based on sorting can be further improved asymptotically using a technique presented by Cole.[11] In general, this technique yields an algorithm of runtime $\mathcal{O}((k + T_{\text{dec}}) \log k)$. Here T_{dec} is the (sequential) time for the decision problem and k denotes the number of values to be sorted which is $\mathcal{O}(pq)$ in our case.

Altogether, we have the following algorithm for computing $\delta = \delta_F(P, Q)$:

Algorithm 3:

1. Determine all critical values of ε of types a) and b) and apply onto them the technique of Algorithm 2. This gives two values ε_1, ε_2 with $\delta \in [\varepsilon_1, \varepsilon_2]$ and $\varepsilon \notin [\varepsilon_1, \varepsilon_2]$ for any critical value ε of type a) or b) other than ε_1 or ε_2.

2. Let A be the set of endpoints $a_{ij}, b_{ij}, c_{ij}, d_{ij}$ of intervals L_{ij}^F or B_{ij}^F that are nonempty for $\varepsilon \in [\varepsilon_1, \varepsilon_2]$ (A is determined by the critical values ε of Step 1 with $\varepsilon \leqslant \varepsilon_1$). Use Cole's variant of parametric search based on sorting the values in A to find the actual value of δ.

Since there are $\mathcal{O}(pq)$ critical values of type a) or b), Step 1 takes $\mathcal{O}(pq \log(pq))$ time. As was mentioned before Step 2 takes time $\mathcal{O}((k+T_{\text{dec}}) \log k)$ which is $\mathcal{O}(pq \log(pq))$ using Algorithm 1 as the decision algorithm.

We summarize:

Theorem 6 *For given polygonal curves P, Q Algorithm 3 computes the Fréchet-distance $\delta_F(P, Q)$ in time $\mathcal{O}(pq \log(pq))$.*

It should be mentioned here that Algorithm 3, although it has a low asymptotic complexity, is not really applicable in practice. In fact, Cole's parametric searching technique makes use of the Ajtai-Komlos-Szemeredi (AKS) sorting network[12] which involves enormous constants. One of the authors has implemented a method which determines δ bit by bit using Algorithm 1 in each step. A possibly practically realistic alternative would be to use Megiddo's original parametric search together with a simple parallel sorting algorithm like odd-even merge. This method would have an asymptotic runtime of $\mathcal{O}(pq \log^3(pq))$

3. Variants of the Fréchet-metric

3.1. The nonmonotone Fréchet-metric

Let us define the *nonmonotone Fréchet-metric* δ_N between curves C_1, C_2 as in Definition 2, except that α_1, α_2 are allowed to be non-monotone. It can be shown[5] that δ_N is a *pseudo-metric* (i.e. two objects with distance 0 may be distinct).

[a] It is not obvious that a model of computation involving only arithmetic operations and square roots can determine the critical values of these comparisons. One possibility would be to extend the model involving the computation of roots of low-degree polynomial equations as well, but a detailed consideration[5] shows that, in fact, square roots suffice.

It follows directly from the definitions that for any two curves C_1, C_2

$$\delta_H(C_1, C_2) \leqslant \delta_N(C_1, C_2) \leqslant \delta_F(C_1, C_2).$$

Figure 1 shows that there can be an arbitrary large ratio between δ_H and δ_N, Figure 7 shows that the same is true between δ_N and δ_F, also giving some indication of the drawbacks of δ_N for some problem instances.

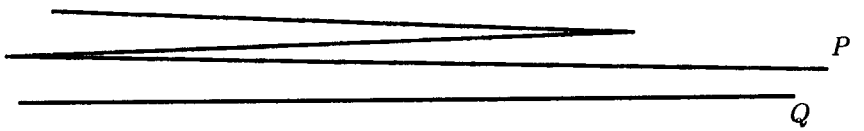

Fig. 7. Polygonal chains with small δ_N – but large δ_F-distance

The problem of computing the nonmonotone Fréchet-distance is closely related to a problem from Robotics, the so-called *ring-width problem*, which was stated and solved by Goodman, Pach, and Yap[13]: Suppose a closed polygon P in $\mathbb{R}^2$ is given with two handles (rays) sticking out from two points on its boundary. What is the minimal width of a ring through which P can be moved starting at one handle and ending at the other? The authors showed that the ring width equals the *elastic ring-width* which is the maximal width of the ring during the motion if it is allowed to expand and shrink. From this it easily follows that the ring-width equals $\delta_N(P_1, P_2)$ where P_1, P_2 are the two parts of P on either side of the handles. Although in the ring-width problem both curves have the same starting and the same endpoint, the algorithm given by Goodman, Pach and Yap can easily be modified to compute δ_N for arbitrary polygonal curves. In fact, it uses similar techniques and considerations as ours and has the same runtime, so that this section is not much more than an alternative and possibly simpler formulation once the reader is familiar with the terminology of this paper.

In order to solve the decision and computation problems for $\delta_N(P, Q)$ where P, Q are polygonal curves, we consider the same $p \times q$-diagram as in Section 2. Now the decision problem has a positive answer exactly if there is an arbitrary path in the free space F_ε from $(0,0)$ to (p,q). So the only important information is, for which values of ε we have passages between cells C_{ij} and their neighbors, i.e. the sets L_{ij}^F and B_{ij}^F are nonempty. In other words, the critical values of ε are the ones of types a) or b).

Consider now the undirected graph $G = (V, E)$ where $V = \{C_{ij} \mid 1 \leqslant i \leqslant p, 1 \leqslant j \leqslant q\} \cup \{s, t\}$. E consists of all edges between neighboring cells, i.e. all edges of the form $\{C_{ij}, C_{i+1,j}\}$, $\{C_{ij}, C_{i,j+1}\}$. In addition, we have edges $\{s, C_{11}\}$ and $\{C_{pq}, t\}$. Furthermore, the edges are labeled with critical values of ε. In fact, at the edges between two neighboring cells we have the minimal ε for which there is a possible direct transition between the two cells within F_ε. The edge $\{s, C_{11}\}$ is labeled with the minimal ε for which $(0,0) \in F_\varepsilon$, $\{C_{pq}, t\}$ with the one for which $(p,q) \in F_\varepsilon$. We define the "weight" of a path within G as the largest weight of its edges.

Clearly, now the *decision problem* has a positive answer, exactly if there is a path of weight at most ε between s and t. After having constructed the labeled graph G this question can be answered by removing all edges with weight $> \varepsilon$, and testing whether s and t are in the same connected component by, say, breadth first search. The runtime of this algorithm is $\mathcal{O}(pq)$.

The *computation problem* consists of determining the minimum weight path from s to t. This can be solved for example by using Prim's minimum spanning tree algorithm[14] starting from s and running it until the minimum spanning tree containing s and t is found. Another possibility is Algorithm 2, where now only $\mathcal{O}(pq)$ critical values occur. Both algorithms have runtime $\mathcal{O}(pq \log pq)$. We summarize:

Theorem 7 *Let P, Q be polygonal curves of length p, q, respectively. Then*

a) it can be decided in $\mathcal{O}(pq)$ time whether $\delta_N(P,Q) \leqslant \varepsilon$ for a given ε.

b) $\delta_N(P,Q)$ can be computed in time $\mathcal{O}(pq \log(pq))$.

3.2. The Fréchet-metric for closed curves

Closed curves are curves with common starting and ending points. Consider for example the parametrized curve $(\cos\varphi, sin\varphi)$, where $\varphi \in [0, 2\pi]$. The common starting and ending point is $(1,0)$ but there is no reason to distinguish this point from any other one on the circle, e.g. $\varphi \in [\pi, 3\pi]$ will do essentially the same. We say that the latter curve is the former curve *shifted by* $+\pi$ and would like to identify such shifted curves with each other. In this context Definition 2 is not suitable. Since closed curves are important in practice, we will in this section modify Definition 2 accordingly and develop algorithms for the corresponding decision and computation problem.

Definition 8 *Let C_1 and C_2 be closed curves in V with given orientations. Then we define*

$$\delta_C(C_1, C_2) := \inf_{s_1, s_2 \in \mathbb{R}} \delta_F(C_1 \text{ shifted by } s_1, C_2 \text{ shifted by } s_2)$$

Coming back to our previous man-dog illustration of the Fréchet-metric, this definition now means that both are not only allowed to control their speed but also to choose optimal starting points on the closed curves to minimize the length of the leash.

For the decision problem, whether $\delta_C(P,Q) \leqslant \varepsilon$ for polygonal curves P and Q, we consider the diagram $\widetilde{D}$ for P, Q, ε . Let D be the $2p \times q$-diagram obtained by concatenating two copies of $\widetilde{D}$ occupying the interval $[0, 2p] \times [0, q] \subseteq \mathbb{R}^2$. The following lemma can easily be verified:

Lemma 9 *$\delta_C(P,Q) \leqslant \varepsilon$ exactly if there exist a $t \in [0, p]$ and a monotone curve from $(t, 0)$ to $(t + p, q)$ in the free space F_ε of D.*

In order to test this property, we will describe a data structure which allows us for given points on the boundary of a diagram to check very fast (and in our application in fact in only constant time) whether there exists a monotone curve in the free space of the diagram between them.

Suppose, a diagram D is given and let B, T, L and R be the bottom, top, left and right side of it, respectively. In the data structure these sides are partioned into finitely many intervals with the property, that for any interval on L there is an interval on R at the same height and vice versa. An analogous property holds between B and T. $L \cup B$ is partitioned into intervals of 3 types:

-type ***n***: a connected subset $I \subseteq L \cup B$ so that from *no* point on I any point on $R \cup T$ can be reached by a monotone path in F_ε.

-type ***r***: a connected subset $I \subseteq L \cup B$ so that from any two points in I the same set of points on $R \cup T$ can be *reached.*

-type ***s***: (*"see-through"*) a connected subset $I \subseteq L$ (or B) so that from any point in I the horizontal (or vertical) line segment connecting it with R (or T) lies completely within F_ε.

For an example see Figure 8.

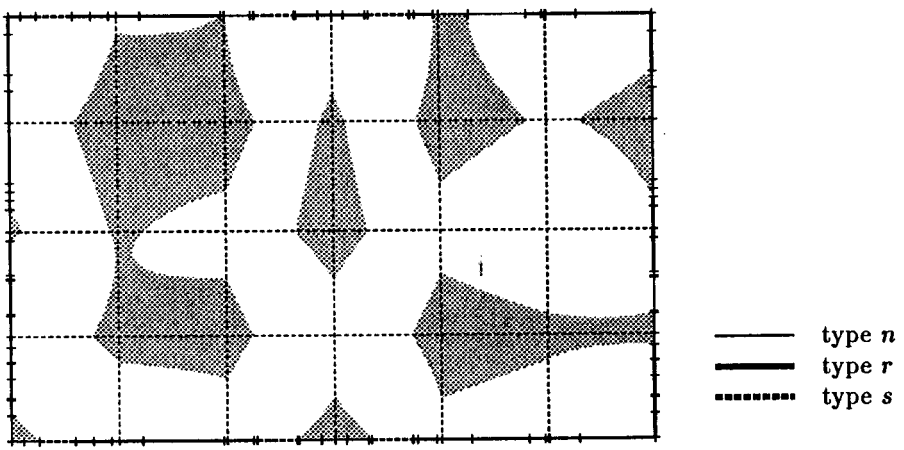

Fig. 8. Partition of the boundary of the diagram.

To each r-interval I of $L \cup B$ we attach pointers h to the highest point on $R \cup T$ (or leftmost, if there is more than one highest, namely on T) that can be reached from I, and ℓ to the lowest reachable point on $R \cup T$ (or rightmost if there is more than one lowest, namely on T). In addition, each s-interval on L has an h- and each s-interval on B an ℓ-pointer.

Likewise, $R \cup T$ is partitioned into n-, s-, and r-intervals depending on their reachability from $L \cup B$. Analogously, there are pointers from these intervals to the lowest (or rightmost) (for s-intervals on R and r-intervals) and to the highest (or

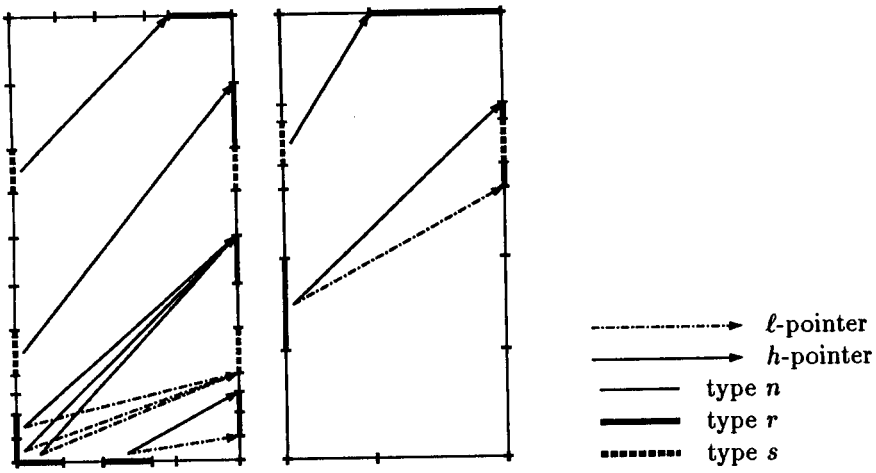

Fig. 9. Two diagrams for merging

leftmost) (for s-intervals on T and r-intervals) point on $L \cup B$ from which they can be reached.

It will be shown later (in Lemma 10) that once this data structure has been constructed it is easy to check for given points on the boundary of the diagram whether there exists the desired curve between them. Now we are showing how to construct the structure recursively:

Clearly, for 1×1-diagrams the partitioning and the pointers can be determined in constant time.

For larger diagrams we apply a divide-and-conquer strategy by splitting the diagram in half at its longer side. So a $p \times q$-diagram D, $p \geqslant q$ is split into a $\lfloor p/2 \rfloor \times q$ diagram D_1 on the right and a $\lceil p/2 \rceil \times q$-diagram D_2 on the left. For D_1, D_2 the problem is solved recursively.

In order to merge the two solutions into one for D, first the intervals in the partition of the right side R_1 of D_1 are merged with the ones of the left side L_2 of D_2. This causes a refinement of the partitions of R_1 and L_2 which is transferred to L_1 and R_2, as well. Each new interval gets the same type and the same pointers as the old interval from which it is a subset. For an example see Figures 9 and 10.

Then the types and pointers of the intervals on $L_1 \cup B_1$ and on $R_2 \cup T_2$ are updated. (Notice, that intervals and pointers on T_1 and B_2 do not change.) We will describe here the updating of the intervals on $L_1 \cup B_1$, the ones on $R_2 \cup T_2$ are updated analogously:

In a first scan we mark as *nonpassable* all intervals of L_2 (or R_1) which are type-n themselves or where there is a type-n interval on R_1 (or L_2) at the same height. Any h-pointer of an interval $K \subseteq B_1 \cup L_1$ into a nonpassable R_1-interval is set to the nearest passable point below. Likewise, the ℓ-pointers into nonpassable intervals are set to the nearest passable point above. If for some s-interval $K \subseteq L_1$ the corresponding interval K' of R_1 is nonpassable, set the type of K to r and $\ell(K)$

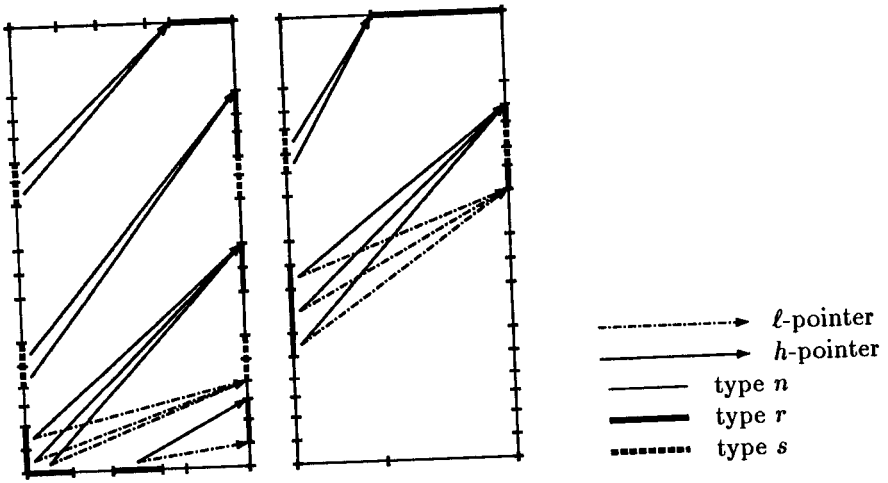

Fig. 10. Two diagrams after refinement for merging

to the nearest passable point above K'. If in this process the h-pointer of some interval $K \subseteq B_1 \cup L_1$ becomes lower than its ℓ-pointer, the pointers are removed and K is set to type-n.

Next we scan through the intervals on $B_1 \cup L_1$. For each r- or s-interval K we update type and pointers as follows:

1. If $h(K)$ exists and points into some r- or s-interval I of L_2, then $h(K) := h(I)$.
2. If $\ell(K)$ exists and points into some interval $I \subseteq L_2$ then:
   2.1 If I is type-r, then $\ell(K) := \ell(I)$.
   2.2 If I is type-s, then set $\ell(K)$ to the point on R_2 opposite of the lower endpoint of I.
3. If K is type-s and K' is the corresponding interval at the same height on L_2, then:
   3.1 If K' is type-r then set K to type-r and $\ell(K) := \ell(K')$.

For an example see Figures 10 and 11.

It is not hard to verify that the total time of the merging is proportional to the number of intervals in the partitionings of D_1 and D_2. Since any interval boundary is one of the values $a_{ij}, b_{ij}, c_{ij}, d_{ij}$ as defined in Section 2, their number is at most $\mathcal{O}(pq)$. Consequently, the runtime of the merging step is $\mathcal{O}(pq)$ and that of the whole divide- and- conquer-algorithm $\mathcal{O}(pq \log pq)$. Once the partition of the boundary of D and the pointers are known, the decision problem can easily been solved according to Lemma 9 and by the following

Lemma 10 *Let D be a diagram with the data structure explained above. Furthermore let P be a point in an interval I on B and Q a point in an interval J on T.*

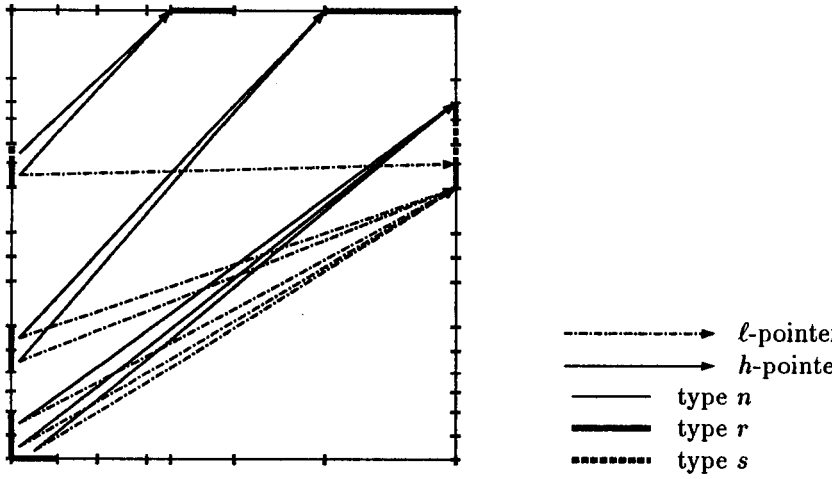

Fig. 11. Diagram merged from the ones above

Then there exists a monotone path from P to Q within the free space F_ε of D iff the following two conditions are true:

1. *I is type-r and Q lies between $h(I)$ and $\ell(I)$* **or** *I is type-s and Q lies to the right of P and to the left of $\ell(I)$.*
2. *J is type-r or type-s*

Proof: Clearly 1. and 2. hold if there exists such a path. Conversely let 1. and 2. be true, see Figure 12. By 2. it follows that there exists a monotone path from a point $P' \in L \cup B$ to Q. So this path must intersect either the path from P to $h(I)$ or, as shown in the figure, the one from P to $\ell(I)$. Let S be the intersection point. So the desired path from P to Q is the one obtained by concatenating the path from P to S and the one from S to Q. □

In order to use Lemma 9 we scan the intervals on the bottom and the top of the diagram simultaneously from left to right. Observe that we have to compare the boundaries of the intervals in the following way: Let for example u be the x-coordinate of the interval boundary at the bottom currently considered and v be the one at the top of the diagram. Then we have to compare $u + p$ with v. So we have to consider a new type of critical values in addition to types a), b), c). Observe furthermore that the values u and v have appeared somewhere in the diagram as some boundaries of the free space in some cells. Thus we can characterize these critical values as follows:

d) ε is minimal with $c_{i,j} + p = c_{i+p,k}$ or $c_{i,j} + p = d_{i+p,k}$ or $d_{i,j} + p = c_{i+p,k}$ or $d_{i,j} + p = d_{i+p,k}$ for some i, j, k.

Now we can solve the computation problem by parametric search just as in section 2, where we now have to add the values $c_{ij} + p, d_{ij} + p$ to the set of values $a_{ij}, b_{ij}, c_{ij}, d_{ij}$ to which the parallel sorting algorithm is applied.

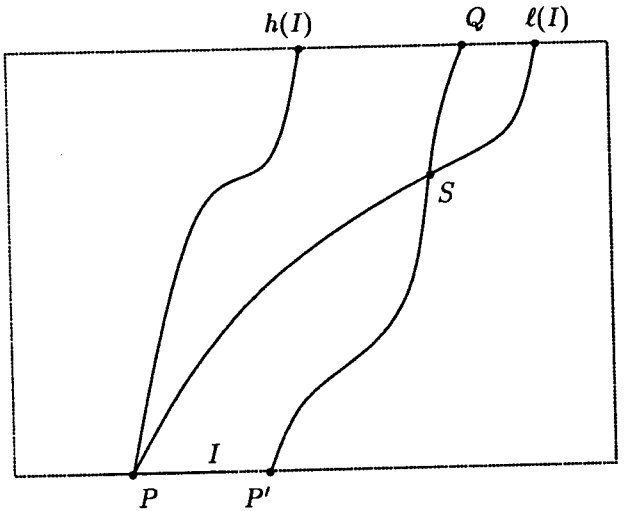

Fig. 12. Illustrating Lemma 10

Altogether, we have:

Theorem 11 *For the closed Fréchet-metric δ_C the decision problem, whether $\delta_C(P,Q) \leqslant \varepsilon$ can be solved in time $\mathcal{O}(pq \log pq)$, the problem of computing $\delta_C(P,Q)$ in time $\mathcal{O}(pq \log^2(pq))$.*

We conclude this section considering some further variants of the problem:

The idea of the divide-and-conquer algorithm can also be used to design *parallel* algorithms. In fact, for the decision problem it is possible to obtain an algorithm of runtime $\mathcal{O}(log^2(pq))$ using pq processors.[5] A parallelization of the parametric search by Meggido[9] then leads to a runtime of $\mathcal{O}(log^4(pq))$ with pq processors for the computation problem.[5]

For completeness, it should be mentioned, that the problem becomes much easier for *convex* polygons in the plane. In fact, it can be shown[15] that $\delta_C(C_1,C_2) = \delta_H(C_1,C_2)$ for curves C_1 and C_2 describing the boundary of convex sets in the same orientation. The Hausdorff-distance between polygons P and Q can be computed in time $\mathcal{O}((p+q)\log(p+q))$.[1]

It seems natural to combine both variants of the Fréchet-metric considered in this paper, i.e. define the *nonmonotone Fréchet-metric for closed curves*. However, it turns out that several definitions are possible. For the most suitable one examples can be found showing that the distance function obtained does not even satisfy the triangle inequality any more,[5] i.e. it is not a pseudo-metric. Therefore it seemed not appropriate to us to consider this generalization, although the ideas from this paper may be applicable.

3.3. The partial matching problem

Certainly, there are applications, where one wants to know whether a given curve g resembles some *part* of a larger curve f. Therefore we consider the following

(asymmetric) distance measure for curves f and g:

$\delta_P(f,g) := \inf\{\delta_F(h,g) \mid h \text{ is a subcurve of } f\}$

The computation of δ_P is a simple application of the data structure and techniques from the last section. First we consider the decision problem again. Given two polygonal curves P and Q and an $\varepsilon \geqslant 0$. Is there a subcurve R of P with $\delta_F(R,Q) \leqslant \varepsilon$?

This is obviously the case iff in the corresponding diagram there exists a monotone curve from the bottom side to the top side. Once we have constructed the data structure from the last section we only have to check the types of the intervals on the bottom of the diagram. If all intervals are of type n, then the answer is "no", otherwise it is "yes".

The computation problem can be solved by parametric search again. Altogether we get algorithms with the same running times as in Section 3.2:

Corollary 12 *For the partial matching distance δ_P the decision problem, — namely to decide whether $\delta_P(P,Q) \leqslant \varepsilon$ — can be solved in time $\mathcal{O}(pq \log pq)$. The problem of computing $\delta_P(P,Q)$ can be solved in time $\mathcal{O}(pq \log^2(pq))$.*

Acknowledgement

The authors would like to thank *Kurt Mehlhorn* for contributing some useful ideas to the divide-and-conquer algorithm in Section 3.2.

References

1. H. Alt, B. Behrends, J. Blömer, "Approximate Matching of Polygonal Shapes", *Proccedings of the 7th ACM Symposium on Computational Geometry* (1991) 186–193.
2. M. Fréchet, "Sur quelques points du calcul fonctionnel", *Rendiconti del Circolo Mathematico di Palermo*, **22** (1906) 1–74.
3. G.M. Ewing, *Calculus of Variations with Applications* (Dover Publ., New York, 1985).
4. M. Godau, "A Natural Metric for Curves-Computing the Distance for Polygonal Chains and Approximation Algorithms", *Proceedings Symposium on Theoretical Aspects of Computer Science*, STACS, Springer Lecture Notes in Computer Science **480** (1991) 127–136.
5. M. Godau, Die Fréchet-Metrik für Polygonzüge — Algorithmen zur Abstandsmessung und Approximation, Diplomarbeit, Fachbereich Mathematik, FU Berlin 1991.
6. H. Alt, M. Godau, "Measuring the Resemblance of Polygonal Curves", *Proccedings of the 8th ACM Symposium on Computational Geometry* (1992) 102–109.
7. B.K. Natarajan, "On Piecewise Linear Appproximations to Curves", Hewlett-Packard Laboratories, Technical Report, HPL-91-36, Palo Alto, USA, 1991.
8. L.J. Guibas, J.E. Hershberger, J.S.B. Mitchell, J.S. Snoeyink, "Minimum link approximation of polygons and subdivisions", In *ISA'91 Algorithms*, eds. W.L. Hsu and R.C.T. Lee, in Lecture Notes in Computer Science **557** (Springer-Verlag, 1991) 151–162.
9. N. Megiddo, "Applying Parallel Computation Algorithms in the Design of Serial Algorithms", *J. of the Assoc. for Comp. Machinery* **30** (1983) 852–866.

10. R. Cole, "Parallel Merge Sort", *SIAM J. on Computing* **17** (1988) 770–785.
11. R. Cole, "Slowing Down Sorting Networks to Obtain faster Sorting Algorithms", *Journal of the Association for Computing Machinery* **34** (1987, Number 1) 200–208
12. M. Ajtaj, J. Komlos, E. Szemeredi, "An $\mathcal{O}(n \log n)$ sorting network". *Combinatorica* **3**, (1993) 1-19.
13. J.E. Goodman, J. Pach, C.K. Yap, "Mountain Climbing, Ladder Moving, and the Ring-Width of a Polygon", *American Math. Monthly* **96** (1989) 494–510.
14. T.H. Cormen, C.E. Leiserson, R.L. Rivest, *Introduction to Algorithms* (MIT-Press, 1990).
15. H. Alt, J. Blömer, M. Godau, H. Wagener "Approximation of Convex Polygons", *Proceedings ICALP, International Colloquium on Automata, Languages and Programming* (Warwick, England, 1990) 703–716.