Computing the cut distance of two curves*

Maike Buchin[1], **Leonie Ryvkin**[2], **and Jérôme Urhausen**[3]

1 **Faculty of Mathematics, Ruhr University Bochum**
`maike.buchin@rub.de`

2 **Faculty of Mathematics, Ruhr University Bochum**
`leonie.ryvkin@rub.de`

3 **Department of Information and Computing Sciences, Utrecht University**
`J.E.Urhausen@uu.nl`

Abstract

The recently introduced k-Fréchet distance extends the well-known Fréchet distance to objects of rearranged pieces. We focus on a variant of this, namely the cut distance, where the input curves are cut into subcurves, which are then matched regarding their similarity with respect to the weak Fréchet distance. It is NP-hard to decide the cut distance of two curves, however, we hereby present a polynomial-time algorithm in case the curves are only cut into two subcurves.

1 Introduction

Comparing geometric shapes is a topic of great interest as it comes up in many applications [4]. Two measures that have proven useful in many applications are the Hausdorff and the Fréchet distance. While the Hausdorff distance can be computed more efficiently, the Fréchet distance gives more information by taking into account how the curves are traversed.

The k-Fréchet distance bridges between Hausdorff and (weak) Fréchet distance. It comes in two variants: the cut and the cover variant. Here we consider the cut distance, which allows to cut a curve into a number of subcurves where the subcurves resemble each other in terms of the (weak) Fréchet distance. Thus it allows to find similarities between objects of rearranged pieces such as chemical structures or handwritten characters.

Characterizing these distance measures in the free space (defined below) shows that the k-Fréchet distance bridges between the (weak) Fréchet distance and Hausdorff distance (see below for details): the weak Fréchet distance can be characterized by one component in the free space projecting surjectively onto both parameter spaces, whereas the Hausdorff distance can be characterized by the free space projecting surjectively onto both parameters. For the cut distance, we need to find a subdivision of the free space in a (possibly non-uniform) grid with $k \times k$ cells such that we can choose exactly one cell per row and column that contains a component that surjectively projects onto the boundaries of this cell. The cells may but need not share a common corner.

Related work. Efficient algorithms for computing the Fréchet distance and the weak Fréchet distance were presented by Alt and Godau in 1995. They first introduced the concept of the free space diagram, which is key to computing this distance measure and its variants [3]. Following their work, numerous variants and extensions have been considered. Here we mention only those most related to our work. Alt, Knauer and Wenk [5] compared Hausdorff to Fréchet distance and showed that for convex closed curves Hausdorff distance equals Fréchet distance. For curves in one dimension Buchin et al. [6] proved equality of Hausdorff and weak Fréchet distance using the well-known Mountain climbing theorem [8].

* Jérôme Urhausen is supported by the Netherlands Organisation for Scientific Research under project 612.001.651

36th European Workshop on Computational Geometry, Würzburg, Germany, March 16–18, 2020.
This is an extended abstract of a presentation given at EuroCG'20. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

The k-Fréchet distance was first studied by Buchin and Ryvkin [7]. They showed that deciding the cut distance is NP-hard, and optimizing is even APX-hard. Later Akitaya et al. [2, 1] considered a second variant of the k-Fréchet distance, called cover distance. For this, the input curves are also divided into subcurves, which are to resemble each other in terms of the weak Fréchet distance. But in contrast to the cut distance, those subcurves are allowed to overlap. Deciding the cover distance is also NP-hard, but it can be approximated efficiently [1]. Algorithmically the cut distance is even more challenging than the cover distance: whereas the latter "only" asks for k components in free space that completely project onto both parameter spaces, the former additionally requires this without overlap.

Overview. First, in Section 2, we recall some necessary definitions and formally define the cut distance. In Section 3 we present a polynomial-time algorithm for the cut distance for $k = 2$, i.e., cutting both input curves into two pieces.

2 Definitions

The Fréchet distance [3], a well-known measure for curves, is defined as follows: For curves $P, Q\colon [0,1] \to \mathbb{R}^2$, the Fréchet distance is given by

$$\delta_{\mathrm{F}}(P,Q) = \inf_{\sigma,\tau} \max_{t\in[0,1]} \|P(\sigma(t)) - Q(\tau(t))\|,$$

where the reparametrisations $\sigma,\tau\colon [0,1] \to [0,1]$ range over all non-decreasing surjections. A variant is the weak Fréchet distance $\delta_{\mathrm{wF}}(P,Q)$, where both curves are re- parameterised by σ and τ, respectively, which range over all continuous surjections. All computation needs discrete input data, so we assume that curves are polygonal chains in the following.

A well-known characterisation, which is key to efficient algorithms for computing the (weak) Fréchet distance [3], uses the free space diagram. First we define the free space F_ε:

$$F_\varepsilon(P,Q) = \{(t_1,t_2) \in [0,1]^2 \colon \|P(t_1) - Q(t_2)\| \leqslant \varepsilon\}.$$

The free space diagram puts this information into an $(n \times m)$-grid, where n, m are the numbers of segments in P and Q, respectively. Bottom and left boundary of the diagram correspond to the *parameter spaces* of the curves P and Q.

The Fréchet distance of two curves is at most a given value ε if there exists a monotone path through the free space connecting the bottom left to the top right corner of the diagram. For the weak Fréchet distance to be at most ε, we need a continuous path through the free space that connects the four boundaries. For the Hausdorff distance it holds that $\delta_{\mathrm{H}} \leq \varepsilon$ if there is a surjective projection of the free space onto both parameter spaces, see Figure 1.

We define further terms regarding the free space diagram below. A *component* is a maximal connected subset $c \subseteq F_\varepsilon(P,Q)$. Figure 2 shows a component ranging over two cells. A set S of (parts of) components *covers* a set $I \subseteq [0,1]_P$ of the parameter space (corresponding to the curve P) if I is a subset of the projection of S onto the above defined parameter space, i.e., $\forall x \in I \colon \exists c \in S, y \in [0,1]_Q \colon (x,y) \in c$. Covering on the second parameter space is defined analogously.

▶ **Definition 2.1.** For polygonal curves P, Q we define the cut version of the k-Fréchet distance (also called *cut distance* for short) as

$$\delta_{\mathrm{cut}}(k,P,Q) = \inf_{\sigma,\tau} \max_{t\in[0,1]} \|P(\sigma(t)) - Q(\tau(t))\|,$$

where now $\sigma,\tau\colon [0,1] \to [0,1]$ range over all continuous, surjective functions with $k' < k$ jump discontinuities, each.

That is, we cut the curves P and Q into at most k pieces, or subcurves, each, such that two resembling subcurves have small weak Fréchet distance. In the free space diagram, we insert the cuts on our curves as horizontal and vertical grid lines. For every row and column of this “cutting grid”, we select exactly one cell. The cell corresponds to a matched pair of subcurves and hence needs to contain (a part of) a free space component that projects surjectively onto the cell’s boundaries. The cover distance $\delta_{\text{cover}}(k, P, Q)$ is defined analogously, but now the pieces are allowed to overlap [2]. Intuitively, we ask for a selection of at most k components that cover the parameter spaces when using the cover distance.

For the decision problem we ask whether the weak Fréchet distance between pieces can be bounded by a given value ε, where the number of subcurves is upper bounded by k. For fixed ε, we want to minimize k (optimization problem). The value of the cut distance lies in between Hausdorff and weak Fréchet distance and is lower-bounded by the cover distance:

$$\delta_{\text{H}}(P, Q) \leq \delta_{\text{cover}}(k, P, Q) \leq \delta_{\text{cut}}(k, P, Q) \leq \delta_{\text{wF}}(P, Q).$$

Note that it holds that $\delta_{\text{cover}}(1, P, Q) = \delta_{\text{wF}}(P, Q)$ and $\delta_{\text{cover}}(n, P, Q) = \delta_{\text{H}}(P, Q)$. Figure 1 gives an example where cut and cover distance differ.

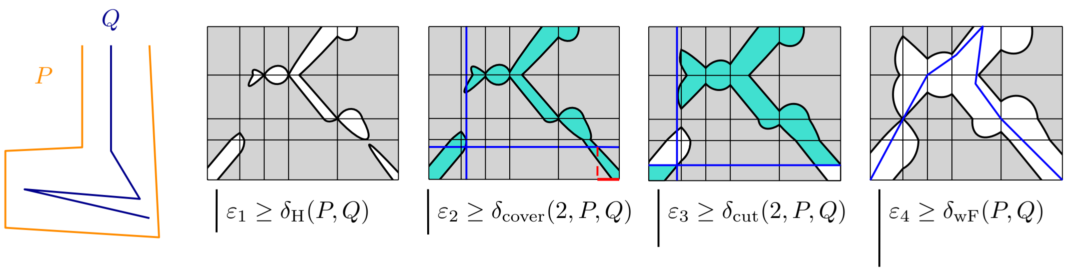
Q
P
$\varepsilon_1 \geq \delta_{\text{H}}(P, Q)$
$\varepsilon_2 \geq \delta_{\text{cover}}(2, P, Q)$
$\varepsilon_3 \geq \delta_{\text{cut}}(2, P, Q)$
$\varepsilon_4 \geq \delta_{\text{wF}}(P, Q)$

Figure 1 Comparison of distance measures. In the second diagram, two components are sufficient to cover the parameter spaces, but cutting does not work, because by choosing the bottom left and top right cell the red section on the bottom parameter space would not be covered.

3 Polyonomial Time Algorithm for k=2

As we already know that deciding the cut distance problem is NP- and optimizing k is APX-hard [7], we do not expect to solve it or approximate the number of cuts in polynomial time. Instead, we give a polynomial-time algorithm for $k = 2$. As we observe below, already for $k > 2$ placing cuts becomes more difficult and we leave this as an open problem. In the following we always assume that both curves have complexity n.

Cut placements The first difficulty is that there are infinitely many possibilities of placing a cut. For $k = 2$, we can reduce this amount to a finite set of discrete positions. We define *interesting points* to be local extrema of a component’s boundary. We call a horizontal or vertical cut line through an interesting point an *interesting line*. We want cut lines ℓ and h such that there are two components covering opposing quadrants formed by ℓ and h. Such a placement of cut lines is called *valid*.

▶ **Theorem 3.1.** *If there exists a valid placement of cut lines in $F_\varepsilon(P, Q)$ for $k = 2$, it is possible to move these cut lines such that at least one of them becomes an interesting line.*

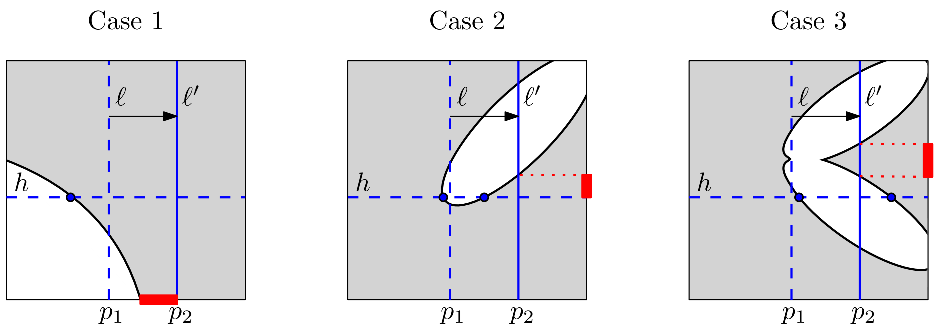

Figure 2 Given valid cut lines, moving one cut line beyond (new) interesting points may alter coverage of a quadrant. New interesting points n_i re marked with blue disks.

Proof. We are given a valid placement of cut lines and assume none of them features an interesting point. The cut lines subdivide the free space diagram into four quadrants of which two opposing ones are covered by components c_1 and c_2.

We fix the horizontal cut line h. It intersects c_1 and c_2 at *new interesting points* $n_1, \ldots n_k$. Note that moving the vertical cut line to the right from ℓ through p_1 to ℓ' through p_2 can change the coverage in general, see Figure 2:

1. The left component might not cover a subinterval of $[p_1, p_2]$ on the curve P.
2. The right component might no longer cover an interval on the curve Q.
3. The right component might become disconnected.

These cases can only occur when there is a (new) interesting point between p_1 and p_2. Symmetrically, this holds for moving ℓ to the left, or moving the horizontal line h. Thus, as long as we do not move cut lines past (new) interesting points, the cut lines stay valid.

Let $I = [n_i, n_{i+1}]$ be the interval of adjacent new interesting points containing the given vertical cut line ℓ. If there is an interesting point $p \in I$, we move ℓ to the closest interesting point in I. Else we move ℓ to n_i. The intersection z of ℓ and h now lies on the boundary of a component. Next, we can move ℓ and h simultaneously such that z moves along the cell's boundary until one of the cut lines features an interesting point. ◀

Note that for $k > 2$, moving one cut line can make it necessary to move a second cut line, see Figure 3. This second cut line can cause the necessity to move a third, and so on.

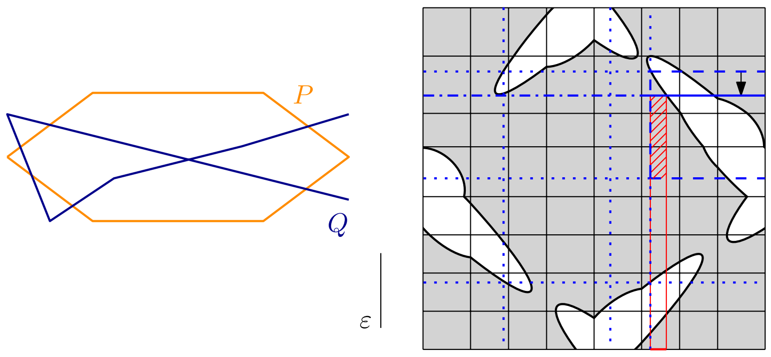

Figure 3 The dotted blue lines are validly placed cuts. The dashed parts enclose the cut cell we focus on: moving the top cut line downwards onto the dash dotted line (such that it coincides with the bottommost point of the upper left component) leaves part of the cut cell uncovered, thus its vertical cut line would need to be moved to the right. This process may repeat.

Algorithm First, we observe the following: The cuts induce a 2×2 grid on the free space diagram, and the cells featuring opposing corners are selected. Those selected cells each need to contain (a part of) one component that surjectively projects onto the cell's boundaries, and thus needs to touch two consecutive boundaries of the free space diagram. We call a component with this property a *candidate component*. As each of the $n \times n$ cells corresponding to a pair of segments contains no more than one cell, we can upper bound the number of candidate components by n. Note that for any pair of cut lines there exists at most one component per cell touching its cell's boundaries.

We give a brief overview of the algorithmic steps before going into detail. Note that we first run the algorithm assuming we select components in the bottom left and top right quadrant of F_ε and repeat steps 2-5 for the other option if the first run yields no solution. If there is still no solution, we repeat the whole process for horizontal interesting lines h instead of vertical interesting lines ℓ.

1. Compute the free space diagram $F_\varepsilon(P, Q)$ and its representative graph G;
2. Identify candidate components $A = \{a_1, a_2, \ldots, a_{|A|}\}$, that touch the bottom and the left boundaries of the free space, and $B = \{b_1, b_2, \ldots, b_{|B|}\}$, that touch the upper and the right boundaries. Sort A from left to right by their first intersection point with the bottom boundary and B from right to left by first intersection with the top boundary;
3. Compute all vertical interesting lines ℓ, i.e., vertical lines that are either tangent to a candidate component's boundary or run through an intersection of the component and the bottom or top boundary of the free space diagram. Store them in a sorted list;
4. For any line ℓ and components a_i, b_j we identify the parts of a_i and b_j that touch ℓ as well as the bottom and left or top and right boundaries, respectively. We call the parts of a_i and b_j fulfilling these requirements a_i^ℓ and b_j^ℓ. If there is no such a_i^ℓ and/or b_j^ℓ, delete ℓ from its list;
5. For each ℓ we recall a_i^ℓ and b_j^ℓ and do:
   a. Determine the topmost (bottommost) horizontal line $h_a^\uparrow$ ($h_b^\downarrow$) that intersects a_i^ℓ (b_j^ℓ);
   b. Determine the bottommost (topmost) horizontal line $h_a^\downarrow$ ($h_b^\uparrow$) such that the part of a_i^ℓ (b_j^ℓ) below (above) that line is still connected and touches ℓ;
   c. Any horizontal line through $[h_a^\downarrow, h_a^\uparrow] \cap [h_b^\downarrow, h_b^\uparrow]$ is a valid horizontal cut h.
6. In case no solution is found, repeat from step 2 for candidate components touching bottom and right or top and left boundary of the free space diagram, respectively;
7. In case no solution is found, repeat from step 3 for interesting horizontal lines h, compute left- and rightmost vertical lines in step 5.
8. If there is a solution, return ℓ and h, as well as the matching of the subcurves.

We first compute the free space diagram and the combinatorial graph G representing it in time $\mathcal{O}(n^2)$. A vertex of G corresponds to a cell or boundary of the diagram and an edge is drawn between vertices of neighboring cells iff the cells share a connected free space component, or between a boundary vertex and a cell vertex iff a component touches the boundary within that cell.

A depth first search (DFS) on G returns all candidate components and a vertical sweep of F_ε yields the interesting lines ℓ. We need $\mathcal{O}(n^2 \log n)$ time to find the candidate components since G has size $\mathcal{O}(n^2)$, and $\mathcal{O}(n^2)$ time to compute and sort all ℓ.

For each interesting line ℓ we then compute the correct pair of candidate components a_i^ℓ and b_j^ℓ as follows: By means of a breadth first search (BFS), where we limit the breadth to the current position of ℓ, we check whether a component touches the neighboring boundaries and ℓ. Most importantly, we ensure that the component is connected to the left (right) of ℓ

(in Figure 2, the dashed vertical cut line disconnects the component to the right of it). We need to check at most two candidates for each ℓ: considering them from left to right, the first interesting line has only one possible candidate, the first component. In the following, this candidate component either still intersects the next ℓ, or the next component in the sorted list has to be checked. We stop and continue the same BFS for all ℓ and all candidate components of a quadrant, and visiting each candidate once we only take quadratic time.

For each valid ℓ we perform a sweep to identify the interval in which the correct candidate components overlap vertically, and delete ℓ if they do not overlap. Within the overlap, any horizontal line is a valid cut placement, w.l.o.g. we choose the lower bound. There are $\mathcal{O}(n)$ extrema of a component's boundary curve to consider, so we take linear time here.

Overall, the runtime of the algorithm is bounded by $\mathcal{O}(n^2 \log n)$. Hence we conclude:

▶ **Theorem 3.2.** *The cut distance for $k = 2$ of two polygonal curves of complexity n can be decided in $\mathcal{O}(n^2 \log n)$ time.*

4 Conclusion

We presented a polynomial time algorithm for deciding the cut distance in case $k = 2$. For general k, we know that deciding the cut distance is NP-hard, and approximating the number of cuts k is APX-hard. For $k \geq 3$, we conjecture that cuts may have to be placed at non-interesting points, see Figure 4.

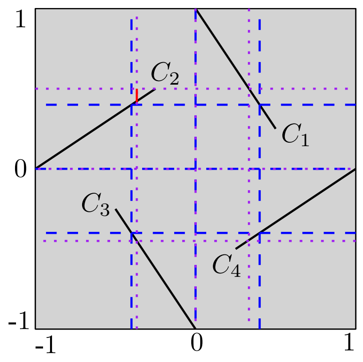

Figure 4 We are given four components $C_1, \dots, C_4$ and apply four valid cut lines (dashed blue). The solution is unique and does not involve interesting points. The purple dotted lines feature the tip of C_2, but, as indicated by the red segment, C_2 does not cover its (purple) cell's boundaries.

References

1 Hugo Akitaya, Maike Buchin, Leonie Ryvkin, and Jérôme Urhausen. The k-Fréchet distance revisited and extended. In *35th European Workshop on Computational Geometry (EuroCG)*, page 7, 2019. URL: `http://www.eurocg2019.uu.nl/papers/41.pdf`.

2 Hugo Alves Akitaya, Maike Buchin, Leonie Ryvkin, and Jérôme Urhausen. The k-Fréchet distance: How to walk your dog while teleporting. In *ISAAC*, volume 149 of *LIPIcs*, pages 50:1–50:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

3 Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *Internat. J. Comput. Geom. Appl.*, 5(1-2):75–91, 1995.

4 Helmut Alt and Leonidas Guibas. Discrete geometric shapes: Matching, interpolation, and approximation: A survey. *Handbook of Computational Geometry*, 1997.

5 Helmut Alt, Christian Knauer, and Carola Wenk. Comparison of distance measures for planar curves. *Algorithmica*, 38(1):45–58, 2004.

6 Kevin Buchin, Maike Buchin, Christian Knauer, Günther Rote, and Carola Wenk. How difficult is it to walk the dog? In *Proc. 23rd Europ. Workshop on Comp. Geom.*, pages 170–173, 2007.

7 Maike Buchin and Leonie Ryvkin. The k-Fréchet distance of polygonal curves. In *34th European Workshop on Computational Geometry (EuroCG), Book of Abstracts*, page 4, 2018. URL: `conference.imp.fu-berlin.de/eurocg18/`.

8 Jacob E. Goodman, János Pach, and Chee-K. Yap. Mountain climbing, ladder moving, and the ring-width of a polygon. *Amer. Math. Monthly*, 96(6):494–510, 1989.