Barking dogs: A Fréchet distance variant for detour detection

Ivor van der Hoog[1], Fabian Klute[2], Irene Parada[3], and Patrick Schnider[4]

1 **Technical University of Denmark, Denmark**
`idjva@dtu.dk`
2 **Universitat Politècnica de Catalunya, Spain**
`fabian.klute@upc.edu`
3 **Universitat Politècnica de Catalunya, Spain**
`irene.parada@upc.edu`
4 **Department of Computer Science, ETH Zürich, Switzerland**
`patrick.schnider@inf.ethz.ch`

Abstract

Imagine you are a dog behind a fence Q and a hiker is passing by at constant speed along the hiking path P. In order to fulfil your duties as a watchdog, you desire to bark as long as possible at the human. However, your barks can only be heard in a fixed radius ρ and, as a dog, you have bounded speed s. Can you optimize your route along the fence Q in order to maximize the barking time with radius ρ, assuming you can run backwards and forward at speed at most s?

We define the barking distance from a polyline P on n vertices to a polyline Q on m vertices as the time that the hiker stays in your barking radius if you run optimally along Q. This asymmetric similarity measure between two curves can be used to detect outliers in Q compared to P that other established measures like the Fréchet distance and Dynamic Time Warping fail to capture at times. In this extended abstract, we consider this measure in the discrete setting, where the traversals of P and Q are both discrete. In this setting, we show how to compute the barking distance in time $O(nm \log s)$.

Related Version arXiv:2402.13159

1 Introduction

A *curve* is any sequence of points in $\mathbb{R}^d$ where consecutive points are connected by their line segment. Curves may be used to model a variety of real-world input such as trajectories [12], handwriting [11, 17] and even strings [3]. Curves in $\mathbb{R}^1$ may be seen as *time series* which model data such as music samples [10], the financial market [13] and seismologic data [16]. A common way to analyse data that can be modeled as curves is to deploy a curve similarity measure, which for any pair of curves series (P, Q) reports a real number (where the number is lower the more 'similar' P and Q are). Such similarity measures are a building block for common analysis techniques such as clustering [7, 15], classification [1, 8, 9] or simplification [2, 6, 14]. The two most popular similarity measures for curve analysis are the Fréchet distance and the Dynamic Time Warping (DTW) distance. The discrete Fréchet distance for two curves $P = (p_1, \ldots, p_n)$ and $Q = (q_1, \ldots, q_m)$ is illustrated as follows. Imagine a dog walking along

Funding statement: F.K. is supported by a "María Zambrano grant for attracting international talent". I.P. is a Serra Húnter fellow. I.,H.: This project has additionally received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 899987.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.

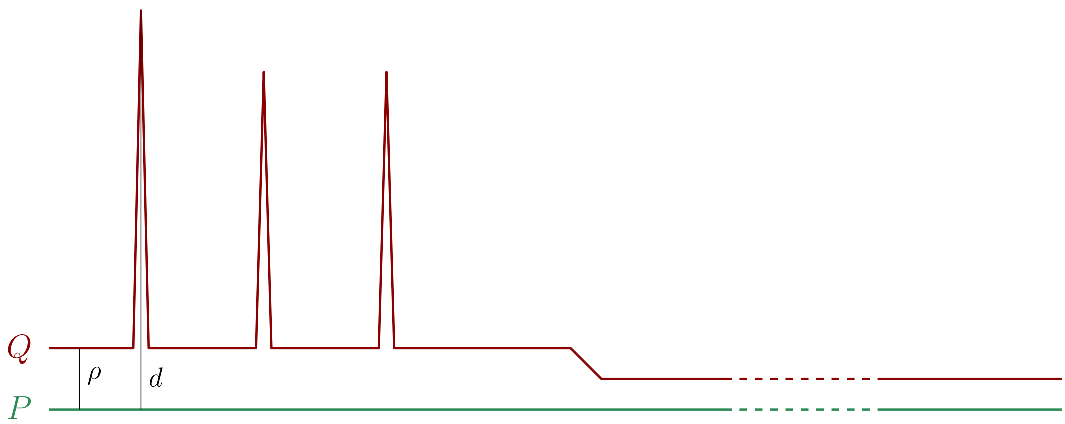

Figure 1 An intended trajectory P and a faulty sample Q of it. The Fréchet distance between P and Q is d and captures the first detour, but fails to capture the others. Continuous DTW, even with a speed bound, cannot distinguish Q from a copy of P translated by ρ if the right part is sufficiently long. Barking distance with barking radius ρ however captures all three detours.

Q and its owner walking along P. Both owner and dog start at the beginning of their curves, and in each step the owner may stay in place or jump to the next point along P and the dog may stay in place or jump to the next vertex along Q, until both of them have reached the end of their curves. Intuitively, the Fréchet distance is the minimal length of the leash between the dog and its owner. The DTW distance is defined analogously but sums over all leash lengths instead.

Both distance measures can be made continuous by defining a traversal as continuous monotone functions $f : [0, 1] \to P$ and $g : [0, 1] \to Q$ which start and end at the respective start and end of the curve. However, for DTW such a direct translation from discrete to continuous traversals invites degenerate behavior. To avoid such degeneracies, Buchin [5] proposed several variants of continuous DTW distances (originally called *average Fréchet* distance) that each penalise the speed of the dog and its owner. The existing curve similarity measures each have their corresponding drawback: The Fréchet distance is not robust versus outliers. The discrete DTW distance is heavily dependent on the sampling rate. The continuous DTW variants are robust to outliers, but they are difficult to compute [4]. Further, all of them fail to capture detours, as can be seen in Figure 1. We present a new curve similarity measure, specifically designed for computing similarities between curves under outliers.

Discrete walks. Given two curves P and Q, we define discrete walks. First, consider the $n \times m$ integer lattice embedded in $\mathbb{R}^2$. We can construct a graph G_{nm} over this lattice where the vertices are all lattice points and two lattice points l_1, l_2 share an edge whenever $d(l_1, l_2) \leq \sqrt{2}$.

▶ **Definition 1.1.** For curves P and Q, a *discrete reparametrization* F is any walk in G_{nm} from $(1, 1)$ to (n, m). F is a curve in $\mathbb{R}^2$ and it is x-monotone whenever its embedding is. The *speed* $\sigma(F)$ is the size $|S|$ for the largest horizontal or vertical subcurve $S \subseteq F$.

Defining Discrete Barking Distance. The barking distance stems from the following illustration, which is again dog-based:[1] assume you are hiking with constant speed along

[1] This illustration is inspired by a dog that some of the authors met while on a hike.

a curve P. A dog is running at bounded speed on a curve Q, constantly barking at you. However, the dogs barks can only be heard within radius $\rho \in \mathbb{R}$. The dog tries to optimize its route in order to maximize the time you hear it. This maximum time is the barking distance of P to Q. Formally, for $\rho \in \mathbb{R}$ we define the *threshold* function as follows:

$$\theta_\rho(p,q) = \begin{cases} 1 & \text{if } d(p,q) > \rho \\ 0 & \text{otherwise.} \end{cases}$$

▶ **Definition 1.2.** For curves P and Q, denote by $\mathbb{F}$ the set of all pairs of discrete x-monotone reparametrizations of (P,Q). For any $\rho, s \in \mathbb{R}$, the discrete barking distance is defined as:

$$\mathbb{D}_B^s(P,Q) = \min_{\substack{F\in\mathbb{F}\\ \sigma(F)\leq s}} \sum_{(i,j)\in F} \theta_\rho(p_i,q_j).$$

2 Computing the Discrete Barking Distance

Let $G_{n,m} = (V,E)$ be a graph defined on top of an $n \times m$ lattice in $\mathbb{R}^2$ where $v_{i,j} \in V$ is identified with the lattice point at coordinate (i,j). We find $(v_{i,j}, v_{i',j'}) \in E$ with distinct $v_{i,j}, v_{i',j'} \in V$ whenever $v_{i,j}$ and $v_{i',j'}$ are identified with points of the lattice at distance $\leq \sqrt{2}$. We say that $v_{i',j'}$ is the *southern*, *south-western*, *western*, *north-western*, or *northern* neighbor of $v_{i,j}$ if $v_{i',j'}$ lies in the corresponding cardinal direction in the lattice.

For $v_{i,j} \in V$ we set $w(v_{i,j}) = \theta(p_i,q_j)$, with p_i the i-th corner of P and q_j the j-th corner of Q. Similarly, we set $w(\pi) = \sum_{a=0}^{k} w(v_{i_a,j_a})$ for a walk $\pi = (v_{i_1,j_1}, \ldots, v_{i_k,j_k})$ in $G_{n,m}$. We say that π is *monotone* if $j_a \leq j_{a'}$ whenever $a \leq a'$ and we define the *length* of π as $|\pi|$, i.e., the number of vertices in the walk. A sub-walk of π is said to be *horizontal* if all its vertices correspond to lattice points with the same y-coordinate and *vertical* if all its vertices correspond to lattice points with the same x-coordinate. Moreover, we say that π has *speed s* if the longest horizontal or vertical sub-walk of π has length at most s. Let $\Pi(s,\rho)$ be the set of all monotone walks in $G_{n,m}$ starting at $v_{1,1}$ and ending at $v_{n,m}$ with speed s and weight function depending on the threshold ρ. The next observation now follows from Definitions 1.1 and 1.2.

▷ Observation 1. Given two polygonal curves P and Q, a threshold ρ, and a speed bound s, let $G_{n,m}$ be defined as above, then $w(\pi) = \mathbb{D}_B^s(P,Q)$ for any $\pi \in \Pi(s,\rho)$ of minimum weight.

By Observation 1 we can restrict our attention to monotone paths from $v_{1,1}$ to $v_{n,m}$ that have speed at most s and are of minimum weight. Our strategy is to compute for each vertex $v_{i,j} \in V$ the weight of such a path from $v_{1,1}$ and to $v_{i,j}$. Our computation will proceed in n rounds, where in each round we consider the m vertices of column j. The challenge is to compute the length of a minimum weight monotone path of speed s in time $O(\log s)$.

Let $R_i(j_1,j_2)$ be the weight of path $(v_{i,j_1+1}, v_{i,j_1+2}, \ldots, v_{i,j_2})$ and $C_j(i_1,i_2)$ be the weight of path $(v_{i_1+1,j}, v_{i_1+2,j}, \ldots, v_{i_2,j})$. Observe, that these values can be computed in constant time if we have arrays containing at position i the length of a path from the first element in the row or column to the i-th element of the row or column. For each row and column and taking either side as the starting vertex. We precompute these arrays for all rows at the beginning and for each column only when we process this column in the computation.

Let $F_\delta(i,j)$ with $\delta \in D = \{\uparrow, \nearrow, \rightarrow, \searrow, \downarrow\}$ be the minimum weight of a monotone path of speed s from $v_{1,1}$ to $v_{i,j}$ where the vertex preceding $v_{i,j}$ on the path is the southern, south-western, western, north-western, or northern neighbor of $v_{i,j}$, respectively. We set $F_\delta(i,j) = \infty$ if $v_{i,j}$ cannot be reached with any monotone path of speed s from $v_{1,1}$.

We then compute the minimum weight monotone path of speed s from $v_{1,1}$ to $v_{i,j}$ as $F(i,j) = \min\{F_\delta(i,j) \mid \delta \in D\}$. To compute $F(i,j)$ from left to right along the columns we maintain the relevant minima of paths F_δ ending at vertices around $v_{i,j}$ for each row and for the current column in separate heaps. Moreover, instead of updating the weights of all heap-elements explicitly for each $v_{i,j}$, we precompute the lengths of paths starting at the beginning or end of a row or column. From this we can in constant time compute the necessary offsets. The runtime of $O(nm \log s)$ then follows as every of the $O(nm)$ elements gets only inserted and deleted from some min-heap a constant number of times and at no point any min-heap contains more than s elements.

For the following proof we rewrite $F_d(i,j)$ as a recurrence taking the speed-bound s into account for $j > 1$. Recall that $w(v_{i,j})$ contributes to the values of C_j and R_i.

$$F_d(i,j) = \begin{cases} \min\{C_j(i-k,i) + F_\delta(i-k,j) \mid \delta \in \{\nearrow, \rightarrow, \searrow\} \wedge k \in [1,s]\} & \text{if } d = \uparrow \\ F(i-1,j-1) + w(v_{i-1,j-1}) & \text{if } d = \nearrow \\ \min\{R_i(j-k,j) + F_\delta(i,j-k) \mid \delta \in \{\uparrow, \nearrow, \searrow, \downarrow\} \wedge k \in [1,s]\} & \text{if } d = \rightarrow \\ F(i+1,j+1) + w(v_{i+1,j+1}) & \text{if } d = \searrow \\ \min\{C_j(i+k,i) + F_\delta(i+k,j) \mid \delta \in \{\nearrow, \rightarrow, \searrow\} \wedge k \in [1,s]\} & \text{if } d = \downarrow \end{cases}$$

▶ **Theorem 2.1.** *Given two polygonal curves P and Q with n and m vertices, respectively, the discrete Barking distance of P to Q can be computed in time $O(nm(\log s))$ where s is the speed bound or time $O(nm \log(nm))$ if $s > n$ or $s > m$.*

Proof. For the first column, i.e., $j = 1$, we can compute $F(i,j)$ as follows. Clearly, $F(1,1) = w(v_{1,1}) = \theta_\rho(p_1, q_1)$. We set $F(i,1) = \infty$ for all $i > s$. Finally, we find that the remaining entries $F(i,1)$ with $i \in [2,s]$ in a bottom-up traversal as the values $C_1(1,i)$. We conclude this step by initializing a min-heap H_i for each row i containing vertex $v_{i,1}$ as its sole element and $F(i,1) = F_\rightarrow(i,1)$ as the key.

Assume now that we want to compute the entries $F_d(i,j)$ for $i \in [1,m]$ where all entries $F_d(i,j')$ with $j' < j$ are already computed and for row i we have a min-heap H_i containing all $F_\rightarrow(i,j-k)$ for $k \in [1,s]$ ordered by key $F_\rightarrow(i,j-k) + R_i(j-k,j-1)$. From this information we can for each i immediately compute $F_\rightarrow(i,j)$ as the minimum of H_i, say $v_{i,j'}$ plus $R_i(j-j',j)$. We then update H_i by deleting all entries for $v_{i,j-s}$ and then inserting $(v_{i,j}, F_\uparrow(i,j))$, $(v_{1,j}, F_\downarrow(i,j))$, $(v_{1,j}, F_\searrow(i,j))$, and $(v_{1,j}, F_\nearrow(i,j))$ using as key for comparison $F_.(i,j) + R_i(j-k,j)$ in the insertion. Note that since for all elements already present in H_i their keys change only by $w(v_{i,j})$ and hence their order remains the same. Moreover, we can directly compute $F_\nearrow(i,j)$ and $F_\searrow(i,j)$ for each i.

It remains to compute $F_\uparrow(i,j)$ and $F_\downarrow(i,j)$ for each $i \in [1,m]$ in column j. We describe how to compute $F_\uparrow(i,j)$, $F_\downarrow(i,j)$ can be computed symmetrically. We start from $v_{1,j}$. Clearly, $F_\uparrow(1,j) = \infty$. We also initialize a min-heap H and insert $(v_{1,j}, F_\rightarrow(1,j))$, $(v_{1,j}, F_\searrow(1,j))$, and $(v_{1,j}, F_\nearrow(1,j) = \infty)$ where the second element is used as key. Assume that we now want to compute $F_\uparrow(i,j)$ and that we have a heap H containing for $k \in [1,s]$ the elements $(v_{i-k,j}, F_\rightarrow(i-k,j))$, $(v_{i-k,j}, F_\searrow(i-k,j))$, and $(v_{i-k,j}, F_\nearrow(i-k,j))$ ordered by key $F_.(i-k,j) + C_j(i-k,i-1)$. This can be done as for the row by just extracting the minimum element from the heap H, say $(v_{i',j}, F_\delta(i',j))$, and setting $F_\uparrow(i,j) = F_\delta(i',j) + C_j(i',i)$. We update the heap as in the case of H_i, with the only difference being that we need to insert the three elements $(v_{i,j}, F_\rightarrow(i,j))$, $(v_{1,j}, F_\searrow(i,j))$, and $(v_{1,j}, F_\nearrow(i,j))$

Correctness follows since the algorithm computes directly the above recurrence. Moreover, since every element vertex and partial weight combination gets deleted and inserted at most once from some heap over the whole computation and no heap contains more than $3s$ elements at a time, we obtain the claimed running time of $O(nm \log s)$. Note, that if $s > m$ or $s > n$ we obtain a runtime of $O(nm(\log(m) + \log(n)))$ since again never more than $O(s)$ elements are contained in a heap and no more than $O(nm)$ elements can be inserted or deleted. ◀

3 Outlook and Conclusion

In the full version of this paper, we also study the barking distance in two other settings, namely the *semi-discrete* and the *continuous* setting. In the semi-discrete setting, the traversal of Q is continuous while the one of P is again discrete. We show the following.

▶ **Theorem 3.1.** *Given two polygonal curves P and Q with n and m vertices, respectively, the semi-discrete Barking distance of P to Q can be computed in time $O(nm \log(nm))$.*

In the continuous setting, both traversals are continuous. Here our algorithm is slower, but still polynomial.

▶ **Theorem 3.2.** *Given two polygonal curves P and Q with n and m vertices, respectively, the continuous Barking distance of Q to P can be computed in time $O(n^4 m^3 \log(nm))$.*

For all the settings we show that, assuming the Strong Exponential Time Hypotheis (SETH), no truly subquadratic algorithm can exist.

▶ **Theorem 3.3.** *Let P and Q be two disjoint polygonal curves with n vertices. Assuming OVC, solving the barking decision problem where the maximum speed of the dog matches the speed of the hiker and with constant barking radius ρ requires $\Omega(n^{2-\epsilon})$ time for any $\epsilon > 0$.*

In the discrete and semi-discrete setting, the runtime of our algorithms match the lower bound up to logarithmic factors. For the continuous setting we give an algorithm that is likely not optimal. We believe that using techniques as for the proof of 3.1 we can improve the runtime to $O(nm^3 \log m)$, but this would still leave a gap between upper and lower bound. While we conjecture that it is possible to obtain an $O(nm \log(nm))$ algorithm, it is likely that new ideas are necessary for this. It would also be interesting to find more efficient algorithms in the continuous setting for restricted types of curves such as time series. Throughout our paper, we assumed that the barking radius ρ and the speed bound s are fixed. Considering them as variables leads to other interesting algorithmic problems where we ask for the minimal speed or barking radius required for the dog such that the hiker can hear it the entire time. We leave the study of these problems for future work.

References

1 Kevin Adistambha, Christian H. Ritz, and Ian S. Burnett. Motion classification using dynamic time warping. In *Proceedings of the 2008 IEEE 10th Workshop on Multimedia Signal Processing (MMSP'08)*, pages 622–627. IEEE, 2008. doi:10.1109/MMSP.2008.4665151.

2 Pankaj K. Agarwal, Sariel Har-Peled, Nabil H. Mustafa, and Yusu Wang. Near-linear time approximation algorithms for curve simplification. *Algorithmica*, 42:203–219, 2005. doi:10.1007/S00453-005-1165-Y.

3 Karl Bringmann and Marvin Künnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. In *Proceedings of the IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS 2015)*, pages 79–97. IEEE, 2015. doi:10.1109/focs.2015.15.

4 Kevin Buchin, André Nusser, and Sampson Wong. Computing continuous dynamic time warping of time series in polynomial time. In *Proceedings of the 38th International Symposium on Computational Geometry (SoCG'22)*, volume 224 of *LIPIcs*, pages 22:1–22:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.SoCG.2022.22.

5 Maike Buchin. *On the computability of the Fréchet distance between triangulated surfaces*. PhD thesis, FU Berlin, 2007. doi:10.17169/refubium-6111.

6 Siu-Wing Cheng and Haoqiang Huang. Curve simplification and clustering under Fréchet distance. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'23)*, pages 1414–1432. SIAM, 2023. doi:10.1137/1.9781611977554.CH51.

7 Anne Driemel, Amer Krivošija, and Christian Sohler. Clustering time series under the Fréchet distance. In *Proceedings of the 2016 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'16)*, pages 766–785. SIAM, 2016. doi:10.1137/1.9781611974331.CH55.

8 Young-Seon Jeong, Myong K. Jeong, and Olufemi A Omitaomu. Weighted dynamic time warping for time series classification. *Pattern recognition*, 44(9):2231–2240, 2011. doi:10.1016/J.PATCOG.2010.09.022.

9 Rohit J. Kate. Using dynamic time warping distances as features for improved time series classification. *Data Mining and Knowledge Discovery*, 30:283–312, 2016. doi:10.1007/S10618-015-0418-X.

10 Joan Serra, Emilia Gómez, Perfecto Herrera, and Xavier Serra. Chroma binary similarity and local alignment applied to cover song identification. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(6):1138–1151, 2008. doi:10.1109/TASL.2008.924595.

11 E Sriraghavendra, K Karthik, and Chiranjib Bhattacharyya. Fréchet distance based approach for searching online handwritten documents. In *Proceedings of the 9th international conference on document analysis and recognition (ICDAR'07)*, volume 1, pages 461–465. IEEE, 2007. doi:10.1109/ICDAR.2007.4378752.

12 Koh Takeuchi, Masaaki Imaizumi, Shunsuke Kanda, Yasuo Tabei, Keisuke Fujii, Ken Yoda, Masakazu Ishihata, and Takuya Maekawa. Fréchet kernel for trajectory data analysis. In *Proceedings of the 29th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL'21)*, pages 221–224, 2021. doi:10.1145/3474717.3483949.

13 Stephen J. Taylor. *Modelling financial time series*. World cientific, 2008.

14 Mees van de Kerkhof, Irina Kostitsyna, Maarten Löffler, Majid Mirzanezhad, and Carola Wenk. Global curve simplification. In *Proceedings of the 27th Annual European Symposium on Algorithms (ESA'19)*, volume 144 of *LIPIcs*, pages 67:1–67:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICS.ESA.2019.67.

15 Weizeng Wang, Gaofan Lyu, Yuliang Shi, and Xun Liang. Time series clustering based on dynamic time warping. In *Proceedings of the 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS'18)*, pages 487–490. IEEE, 2018. doi:10.1109/ICSESS.2018.8663857.

16 Öz Yilmaz. *Seismic data analysis: Processing, inversion, and interpretation of seismic data*. Society of Exploration Geophysicists, 2001. doi:10.1190/1.9781560801580.fm.

17 Jianbin Zheng, Xiaolei Gao, Enqi Zhan, and Zhangcan Huang. Algorithm of on-line handwriting signature verification based on discrete Fréchet distance. In *Proceedings of the 3rd International Symposium on Intelligence Computation and Applications: Advances in Computation and Intelligence (ISICA'08)*, volume 5370 of *LNCS*, pages 461–469. Springer, 2008. doi:10.1007/978-3-540-92137-0_51.