Journal of Algorithms and Computation

journal homepage: http://jac.ut.ac.ir

Fréchet and Hausdorff Queries on x-Monotone Trajectories

Zeinab Saeidi*[1] and Mohammad Farshi†[2]

[1,2]Combinatorial and Geometric Algorithms Lab, Department of Mathematical Science, Yazd University, Yazd, Iran.

ABSTRACT

In this paper, we design a data structure for the following problem. Let π be an x-monotone trajectory with n vertices in the plane and $\epsilon > 0$. We show how to preprocess π and ϵ into a data structure such that for any horizontal query segment Q in the plane, one can quickly determine the minimal continuous fraction of π whose Fréchet and Hausdorff distance to the horizontal query segment Q is at most some threshold value ϵ. We present a data structure for this query that needs $\mathcal{O}(n \log n)$ preprocessing time, $\mathcal{O}(n)$ space, and $\mathcal{O}(\log n)$ query time.

ARTICLE INFO

Article history:
Received 11, October 2018
Received in revised form 12, September 2019
Accepted 15 November 2019
Available online 31, December 2019

Keyword: Data structures, Trajectory data, Computational geometry , Fréchet distance , Hausdorff distance

AMS subject Classification: 05C78.

1 Introduction

Measuring the similarity among curves has been studied in computational geometry [2] and several other research areas, such as data mining [10, 12], image processing [13], and geographic information system (GIS) [11]. Numerous similarity measures have been

*zsaeidi2007@gmail.com
†Corresponding author: M. Farshi. Email: mfarshi@yazd.ac.ir

proposed to measure the distance between two curves such as the Hausdorff and the Fréchet distance.
Movement analysis is important in sports like soccer. The coach of the team may wish to find the shortest sub-path does a player run in a straight line from a specific position on the field to another position. Manual analysis is time-consuming for answering this type of the questions. We can create a system which can find the solution very quickly. The input for our system is a polygonal path. Each path describes the movements of one player. After the system has preprocessed the path, a coach defines a query segment, and the system can quickly find the shortest sub-path which is similar to the query. The system needs a similarity measure. The most common metric is Fréchet distance.
Fréchet distance is generally described as the dog-leash distance; considering a man standing at the starting point of one trajectory and the dog at the starting point of other trajectory. The leash is required to connect the dog and its owner. Both man and his dog are free to vary their speed, but they are not allowed to go backward. The cost of each walk is the maximum leash length required to connect the dog and its owner from the beginning to the end of the respective trajectories. The Fréchet distance is the minimum length of the leash that is needed over all possible walks. More formally, for two curves of A and B, the Fréchet distance between A and B is defined as:

$$\delta_F(A,B) = \inf_{\mu} \max_{a \in A} dist(a, \mu(a)) \tag{1}$$

where $dist(a,b)$ denotes Euclidean distance between point a and b and $\mu : A \to B$ is a continuous and non-decreasing function that maps every point in $a \in A$ to a point in $\mu(a) \in B$ and the start of A is mapped to the start of B to make sure that curves A and B are passed in the same direction. For the polygonal curves A and B with n and m vertices, the Fréchet distance can be computed in $\mathcal{O}(nm \log(nm))$ time [2].
The Hausdorff distance is generally used in computer visions to find all the locations in the image which are matched to the pattern in the input [14]. For the Hausdorff distance between two trajectories, we considered the maximum of the directed Hausdorff distance from trajectory A to trajectory B and vice verse. More precisely, for the two curves A and B, the Hausdorff distance is defined as follows:

$$\delta_H(A,B) = \max\{\delta_{\overrightarrow{h}}(A,B), \delta_{\overrightarrow{h}}(B,A)\}, \tag{2}$$

The directed Hausdorff distance from trajectory A to trajectory B is the maximum distance of every point of the trajectory A to its closest point in the trajectory B. More precisely, the directed Hausdorff distance from A to B is defined as follows:

$$\delta_{\overrightarrow{h}}(A,B) = \sup_{a \in A} \inf_{b \in B} dist(a,b), \tag{3}$$

In other words, the Hausdorff distance between two trajectories is defined as the maximum distance when every point of one is mapped to its closest point in the other. Alt [1] has shown that the Hausdorff distance can be computed in $\mathcal{O}((n+m)\log(n+m))$ time when A and B are polygonal curves with n and m vertices, respectively.

There are a lot of papers that propose data structures for querying a trajectory to answer a specific question. For example compute exact Fréchet distance between trajectory and a given query trajectory or find subtrajectories whose Fréchet distance to a given query trajectory is at most some threshold value.

de Berg *et al.* [4] proposed an algorithm to preprocess a given trajectory π with n vertices and a threshold value ϵ in a data structure such that, given any horizontal segment Q with $length(Q) > 6\epsilon$ as query, counts the number of subtrajectories of π whose Fréchet distance to Q are at most ϵ. If there is no such subtrajectory, their data structure is done correctly, reports zero. Otherwise their data structure may count extra subtrajectories whose Fréchet distance to Q can be up to $(2+3\sqrt{2})$ times larger than ϵ. For a fixed value of ϵ and the parameter s with $n \leq s \leq n^2$, their data structure can be constructed in $\mathcal{O}(n^2+s\,\mathrm{polylog}n)$ with size $\mathcal{O}(s\,\mathrm{polylog}n)$. The query algorithm uses $\mathcal{O}((n/\sqrt{s})\,\mathrm{polylog}n)$ time. In the same paper, they proposed another data structure for the case when ϵ is part of the query. The space and query time is the same as previous case but construction time is $\mathcal{O}(n^3 \log n)$.

Gudmundsson and Smid [9] studied a special case when the input is a geometric tree and the query is a polygonal path. More precisely, for any fixed value of Δ, any geometric tree T that is c-packed and the length of each edge is $\Omega(\Delta)$, and polygonal path Q with m vertices whose edges have the length of $\Omega(\Delta)$, a data structure of size $\mathcal{O}(n\,\mathrm{polylog}n)$ can be built in $\mathcal{O}(n\,\mathrm{polylog}n)$ time that can decide if T contains a path P whose Fréchet distance to Q is at most Δ in $\mathcal{O}(m\,\mathrm{polylog}n)$. If there is no such path P, their data structure is done correctly, reports zero. If Q is a line segment, their data structure reports a path whose Fréchet distance to Q is at most $(1+\epsilon)\Delta$. Otherwise if $m > 1$, a path whose Fréchet distance to Q is at most $3(1+\epsilon)\Delta$ may be reported.

Driemel and Har-peled [7] proposed a data structure for preprocessing a trajectory π such that, given a trajectory Q with k vertices and two points p and q on the π, one can quickly compute approximate Fréchet distance between Q and the subtrajectory of π between p and q. Their data structure can be constructed in $\mathcal{O}(n \log^3 n)$ with $\mathcal{O}(n \log n)$ space. The time complexity of the query algorithm is $\mathcal{O}(k^2 \log n \log (k \log n))$ for computing Fréchet distance with constant approximation factor. In the same paper, they presented another data structure for the case $k = 2$ (line segment) with $\mathcal{O}(\chi^2 n \log^2 n)$ construction time and $\mathcal{O}(\chi^2 n)$ space where $\chi = \epsilon^{-d} \log (1/\epsilon)$. The data structure computes $(1+\epsilon)$-approximation Fréchet distance between a line segment and the subtrajectory of π between p and q in $\mathcal{O}(\epsilon^{-2} \log n \log \log n)$ time.

de Berg *et al.* [6] studied how to preprocess trajectory π and build a data structure such that, given any horizontal query segment Q, one can quickly compute the exact Fréchet distance between the trajectory π and Q. Their data structure can be constructed in $O(n^2 \log n)$ time with $O(n^2)$ size and $O(\log^2 n)$ query time. They proposed another data structure that can determine the exact Fréchet distance between Q and a subtrajectory of π between two vertices that are part of the query. Their data structure needs $O(n^2 \log^2 n)$ construction time and size, and $O(\log^2 n)$ query time.

Gudmundsson *et al.* [8] proposed a data structure for preprocessing a trajectory π with n vertices and threshold value Δ such that, given a trajectory Q with m vertices, one

can quickly decide whether Fréchet distance between π and Q is at most Δ. If each length edge of π is greater than 2Δ and the trajectory Q has m vertices such that the length of each edge is greater than $(1+\sqrt{2})\Delta$, a data structure with $\mathcal{O}(n \log n)$ size and construction time can be built such that one can answer the query in $\mathcal{O}(m \log^2 n)$ time.
Recently, de Berg *et al.* [5] studied how to preprocess a set of trajectories S that can answer similarity queries with respect to the Fréchet distance. More precisely, for any trajectory Q with k vertices, the data structure can answer the following queries: the nearest neighbor queries (report the trajectory in S that is most similar to Q), TOP-j queries (report the j trajectories in S with minimum Fréchet distance to Q), similarity range queries (given a query trajectory Q and a number δ_{max}, report all trajectories in S with Fréchet distance at most δ_{max} to Q), and similarity queries (given a query trajectory Q and a trajectory $\tau \in S$, report the Fréchet distance between Q and τ). Their data structure uses $\mathcal{O}(n/\varepsilon^{2k})$ space and can answer the nearest neighbor queries in $\mathcal{O}(1)$, the TOP-j queries in $\mathcal{O}(j)$, the similarity range queries in $\mathcal{O}(1+|answers|)$, and the similarity queries in $\mathcal{O}(1)$ with approximation ratio $\epsilon.reach(Q)$ for a fixed constant $\epsilon > 0$, where $reach(Q)$ is the maximum euclidean distance between the start vertex of Q and any other vertex of Q.
In this paper, we want to preprocess an x-monotone trajectory π with n vertices and $\epsilon > 0$ into a data structure that one can quickly determine the minimal continuous fraction of the trajectory π whose Fréchet and Hausdorff distance to a horizontal query segment Q is at most a threshold value ϵ. The reported minimal continuous fraction of π is a subpath of π whose endpoints can lie in the interior of some edge of π. Our main result states that an x-monotone trajectory π can be preprocessed in $\mathcal{O}(n \log n)$ time into a data structure with $\mathcal{O}(n)$ storage. A given horizontal query segment Q uses $\mathcal{O}(\log n)$ time for determining the minimal continuous fraction of π whose Fréchet and Hausdorff distance to Q is at most ϵ.
Compared to the other data structure, the main drawback is that we require the input trajectory to be x-monotone. However, the advantages are that: 1) our data structure has linear space, 2) its query time is logarithmic, 3) our data structure can report the exact path not the approximation path.

2 Preliminaries

Let $\pi = (p_0, p_1, ..., p_n)$ be a trajectory in the plane with n vertices and $\epsilon > 0$ as a real number. We assume that query is a horizontal segment $Q = (q_0, q_1)$ with $q_0 = (x_0, y)$, $q_1 = (x_1, y)$, and $x_0 \leq x_1$. We also assume $|Q| > 2\epsilon$. The $\pi[p, p']$ denotes the subpath of π that starts at p and ends at p'.
Let $R(q_0, q_1)$ be the rectangle whose top and bottom sides are parallel to the query Q of the length $|Q|$ and whose left and right sides are orthogonal to the query Q of length 2ϵ; see Figure 1. Let B_1 be the disk with center q_0 and radius ϵ, and let $C_{q_0,\epsilon}$ be the part of the boundary of B_1 contains $R(q_0, q_1)$. Define B_2 and $C_{q_1,\epsilon}$ similarly with respect to q_1. It is easy to see that a path is located in ϵ-distance from Q, if it completely lies in the

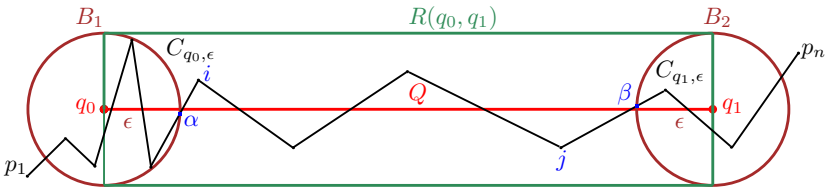

Figure 1: The trajectory π, the horizontal segment Q, the rectangle $R(q_0, q_1)$, two disks B_1 and B_2 and the circular arc $C_{q_0,\epsilon}$ and $C_{q_1,\epsilon}$

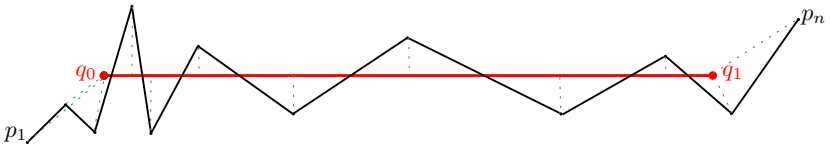

Figure 2: The trajectory π, the horizontal segment Q, and the mapping for proofing the lemma 2.1

area $R(q_0, q_1) \cup B_1 \cup B_2$ as shown in Figure 1.

Lemma 2.1. *For an x-monotone trajectory $\pi = (p_0, p_1, ..., p_n)$ with n vertices and a horizontal segment $Q = (q_0, q_1)$ with $q_0 = (x_0, y)$, $q_1 = (x_1, y)$, and $x_0 \leq x_1$, the Fréchet distance between π and Q can be computed from the following formula:*

$$\delta_F(\pi, Q) = \max\{\|p_0 - q_0\|, \|p_n - q_1\|, \delta_{\overrightarrow{h}}(\pi, Q)\} \tag{4}$$

Proof. Recall that the Fréchet distance between π and Q is the minimum leash length required to connect the dog and its owner. If all points of π are mapped to its closest point on the Q, the minimum length of the leash is attained. This kind of the mapping leads to the directed Hausdorff distance from π to Q. In addition, the start and end points of the π must be mapped to the start and end points of the Q, respectively. The mapping is shown in the Figure 2. Therefore, the following formula is attained: $\delta_F(\pi, Q) \leq \max\{\|p_0 - q_0\|, \|p_n - q_1\|, \delta_{\overrightarrow{h}}(\pi, Q)\}$.
For all of the mapping, the point p_0 is mapped to q_0, and p_n is mapped to q_1. Therefore, $\delta_F(\pi, Q) \geq \|p_0 - q_0\|$ and $\delta_F(\pi, Q) \geq \|p_n - q_1\|$. All point of the π mapped to a point of Q, hence $\delta_F(\pi, Q) \geq \delta_{\overrightarrow{h}}(\pi, Q)$. Combining this, we attain $\delta_F(\pi, Q) \geq \max\{\|p_0 - q_0\|, \|p_n - q_1\|, \delta_{\overrightarrow{h}}(\pi, Q)\}$. □

Lemma 2.2. *Let π be an x-monotone trajectory with n vertices and ϵ is a positive real number. Let $Q = (q_0, q_1)$ be a horizontal segment of length more than 2ϵ. If there exist two points α and β on π, such that $\delta_F(\pi[\alpha, \beta], Q) \leq \epsilon$, then*

1. *The points α and β are inside the disks B_1 and B_2, respectively.*
2. *The path $\pi[\alpha, \beta]$ completely lies in the region $R(q_0, q_1) \cup B_1 \cup B_2$.*

Proof. According to the formula 4

$$\delta_F(\pi[\alpha,\beta],Q) = \max\{\|\alpha - q_0\|, \|\beta - q_1\|, \delta_{\overrightarrow{h}}(\pi[\alpha,\beta],Q)\} \leq \epsilon,$$

Therefore, $\|\alpha - q_0\| \leq \epsilon$, and $\|\beta - q_1\| \leq \epsilon$. We can conclude the points α and β are inside the disks B_1 and B_2.
$\delta_{\overrightarrow{h}}(\pi[\alpha,\beta],Q) \leq \epsilon$, the path $\pi[\alpha,\beta]$ must lie in the ϵ-distance from Q. So, the path $\pi[\alpha,\beta]$ is completely in the area that is indicated in the Figure 1. □

Lemma 2.3. *Let π be an x-monotone trajectory with n vertices and ϵ is a real positive number. Let $Q = (q_0, q_1)$ be a horizontal segment of length more than 2ϵ. If there exist two points of α and β on π, such that $\delta_H(\pi[\alpha,\beta],Q) \leq \epsilon$, then*

1. *The path $\pi[\alpha,\beta]$ is completely inside in the region $R(q_0,q_1) \cup B_1 \cup B_2$.*
2. *The path $\pi[\alpha,\beta]$ contains at least a point in the disk B_1 and a point in the disk B_2.*

Proof. Since $\delta_H(\pi[\alpha,\beta],Q) = max\{\delta_{\overrightarrow{h}}(\pi[\alpha,\beta],Q), \delta_{\overrightarrow{h}}(Q,\pi[\alpha,\beta])\} \leq \epsilon$, we find

1. $\delta_{\overrightarrow{h}}(\pi[\alpha,\beta],Q) \leq \epsilon$: the path $\pi[\alpha,\beta]$ must lie in ϵ-distance from Q. So, it is located in the region $R(q_0,q_1) \cup B_1 \cup B_2$.
2. $\delta_{\overrightarrow{h}}(Q,\pi[\alpha,\beta]) \leq \epsilon$: the start and end point of Q must be mapped to the points whose distance is at most ϵ from q_1 and q_2, respectively. Therefore, the points should be in B_1 and B_2. Hence, the path $\pi[\alpha,\beta]$ contains at least a point in the disk B_1 and a point in the disk B_2.

□

3 Algorithm

According to Lemma 2.2 and Lemma 2.3, the key idea for finding the minimal continuous fraction of the π whose Fréchet and Hausdorff distance to the horizontal query segment Q is at most some threshold value ϵ is to efficiently compute the first hit between π and $C_{q_1,\epsilon}$ as well as, the last hit between π and $C_{q_0,\epsilon}$, and then to check whether the path between two hit points is completely in the region $R(q_0,q_1)$. We computed the subpath using the steps shown in Algorithm 1. We preprocess π as follows:
CircularRayShooting: We compute the points α and β in Figure 1 by constructing the data structure of Theorem 9 designed by Cheng *et al.* [3], which considered a simple polygon of size n and $r > 0$. For any circular query C with radius r and the starting point c', it determines the first intersection point between the boundary of the input polygon and the circular arc C starting at c'. This data structure needs $\mathcal{O}(n \log n)$ preprocessing time, $\mathcal{O}(n)$ space, and answers queries in $\mathcal{O}(\log n)$ time.
RectangleIntersection: We construct a balanced binary search tree storing the points $p_1, p_2, ..., p_n$ in its leaves. At each node of the tree, we store the top and bottom points

between of all points stored in its subtree. Using this structure, we can answer the following type of the query: given any interval $\mathbb{Y}$ and the two indices of i and j with $1 \leq i \leq j \leq n$, we decide if the subpath $\pi[p_i, p_j] = (p_i, p_{i+1}, ..., p_j)$ is completely in $\mathbb{Y}$. To answer a query, we searched the node v_{split} for p_i and p_j where the path to the p_i and p_j splits. The $\pi[p_i, p_j]$ is completely in $\mathbb{Y}$ if and only if the top and the bottom points of v_{split} are in $\mathbb{Y}$. This data structure needs $\mathcal{O}(n \log n)$ preprocessing time, $\mathcal{O}(n)$ space ,and answers queries in $\mathcal{O}(\log n)$ time.

Binary Search: We compute the indices of i and j in Figure 1 via binary search on the edges of π. We start from p_1 and perform a binary search on the edges of π to find the edge that intersects with the desired point. For the point α, we report the endpoint of the edge that has bigger x-coordinate and for the point β, the endpoint with the smaller x-coordinate is reported.

Algorithm 1 Find subpath such that $\delta_h(\pi', Q) \leq \epsilon$

```
1: α ⟵ CircularRayShooting(π, ϵ, C_{q0,ϵ}, [x0 + ϵ, y]);
2: if (α is Null) then
3:    Return Null and Stop.
4: end if
5: β ⟵ CircularRayShooting(π, ϵ, C_{q1,ϵ}, [x1 − ϵ, y]);
6: if (β is Null) then
7:    Return Null and Stop.
8: end if
9: i ⟵ BinarySearch(π, α);
10: j ⟵ BinarySearch(π, β);
11: if (RectangleIntersection ( π[pi, pj] is True ) then
12:    Return π[α, β];
13: end if
14: Return Null;
```

Theorem 3.1. *Let π be an x-monotone trajectory with n vertices and ϵ be a positive constant. A data structure with $\mathcal{O}(n)$ space can be built in $\mathcal{O}(n \log n)$ time such that for any horizontal query segment Q it reports in $\mathcal{O}(\log n)$ time the minimal continuous fraction of π whose Fréchet and Hausdorff distance to the Q is at most ϵ.*

4 Conclusion

The main results of this paper is a query structure that determines the minimal continuous fraction of the x-monotone trajectory π in the plane within Fréchet and Hausdorff distance ϵ from a query horizontal segment Q. Our data structure needs linear space and its query time is logarithmic. We conclude the paper suggesting the following questions:

- In this paper, we only considered a special case in which π is an x-monotone trajectory. Can the same results hold true in case of a general trajectory?

- In this paper we only considered a special case in which Q is a straight-line segment. Can the same results hold true in the case of Q as a polygonal path?
- One other avenue is to report the maximal continuous fraction of π whose Fréchet and Hausdorff distance to the horizontal query segment Q is at most some threshold value ϵ. The naive solution to report the maximal fraction, we need to find these points: first point on the path from β to p_n that exits from B_2 and first point on the path from α to p_1 that exits from B_1. We can find these points with checking the intersection point between each edge in the mentioned paths with the disks. The query time is $\mathcal{O}(k)$ and k is the number of the edges that we need to check for finding the first intersection. For improving the query time, we need significant ideas.

References

References

[1] Alt, H. The computational geometry of comparing shapes. In Efficient Algorithms. Springer, (2009), 235–248.

[2] Alt, H., and Godau, M. Computing the Fréchet distance between two polygonal curves. International Journal of Computational Geometry & Applications 5, 2 (1995), 75–91.

[3] Cheng, S.-W., Cheong, O., Everett, H., and Van Oostrum, R. Hierarchical decompositions and circular ray shooting in simple polygons. Discrete & Computational Geometry 32, 3 (2004), 401–415.

[4] de Berg, M., Cook, A. F., and Gudmundsson, J. Fast Fréchet queries. Computational Geometry 46, 6 (2013), 747–755.

[5] de Berg, M., Gudmundsson, J., and Mehrabi, A. D. A dynamic data structure for approximate proximity queries in trajectory data. In Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (2017), ACM, 48.

[6] de Berg, M., Mehrabi, A. D., and Ophelders, T. Data structures for Fréchet queries in trajectory data. In Proceedings of the 29th Canadian Conference on Computational Geometry (CCCG 2017) (Carleton University, Ottawa, Canada, 2017).

[7] Driemel, A., and Har-Peled, S. Jaywalking your dog: computing the Fréchet distance with shortcuts. SIAM Journal on Computing 42, 5 (2013), 1830–1866.

[8] Gudmundsson, J., Mirzanezhad, M., Mohades, A., and Wenk, C. Fast Fréchet distance between curves with long edges. International Journal of Computational Geometry & Applications 29, 02 (2019), 161–187.

[9] Gudmundsson, J., and Smid, M. Fast algorithms for approximate Fréchet matching queries in geometric trees. Computational Geometry 48, 6 (2015), 479–494.

[10] Keogh, E. J., and Pazzani, M. J. Scaling up dynamic time warping to massive datasets. In European Conference on Principles of Data Mining and Knowledge Discovery (1999), 1–11.

[11] Meulemans, W. W. Similarity measures and algorithms for cartographic schematization. PhD thesis, Technische Universiteit Eindhoven, 2014.

[12] Ratanamahatana, C. A., and Keogh, E. Three myths about dynamic time warping data mining. In Proceedings of the 2005 SIAM International Conference on Data Mining (2005), 506–510.

[13] Sriraghavendra, E., Karthik, K., and Bhattacharyya, C. Fréchet distance based approach for searching online handwritten documents. In Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on (2007), vol. 1, 461–465.

[14] Zhu, L., and Zhu, C.-q. Application of Hausdorff distance in image matching. In Electronics, Computer and Applications, 2014 IEEE Workshop on (2014), IEEE, 97–100.