

Geodesic Fréchet Distance Inside a Simple Polygon*

Atlas F. Cook IV[†] and Carola Wenk[†]

Department of Computer Science, University of Texas at San Antonio
One UTSA Circle, San Antonio, TX 78249-0667

Abstract—We present the first algorithm for the geodesic Fréchet distance between two polygonal curves A and B inside a simple polygon P . If A and B have total complexity N and P has complexity k , then the algorithm runs in $O(k + N^2 \log k N \log N)$ expected time and $O(k + N^2)$ space. This runtime is quite good as it is only a logarithmic factor larger than the non-geodesic Fréchet algorithm [2].

We also unveil an alluring alternative to parametric search that applies to both the non-geodesic and geodesic Fréchet distance algorithms. This randomized approach is based on a variant of red-blue intersections and is appealing due to its elegance and practical efficiency when compared to parametric search.

Motivation

The Fréchet distance [2] is a similarity metric used to compare continuous shapes such as curves or surfaces. It is important because the comparison of geometric shapes is essential in various applications including computer vision, computer aided design, robotics, medical imaging, and drug design. The Fréchet distance for two curves $A, B : [0, 1] \rightarrow \mathbb{R}^l$ is defined as $\delta_F(A, B) = \inf_{f, g: [0, 1] \rightarrow [0, 1]} \sup_{t \in [0, 1]} d(A(f(t)), B(g(t)))$, where f and g range over continuous non-decreasing reparametrizations and d is a distance metric for points. Most previous work on the Fréchet distance assumes an obstacle-free environment where distances are measured using an L_p metric. Our work is different because we measure all distances using geodesics inside a simple polygon P , where a *geodesic* is a path that avoids all obstacles and cannot be shortened by slight perturbations [9]. Note that P is the only obstacle (A and B are *not* obstacles).

Red-Blue Intersections

Red-blue intersections are interesting in their own right and are useful for computing the Fréchet distance. This section shows how to efficiently count and report a certain type of red-blue intersections in the plane inside a search domain $x \in [\alpha, \beta]$. Let $R = \{r_1(x), r_2(x), \dots, r_m(x)\}$ be a set of m continuous “red” curves in the plane such that each red curve is monotone *decreasing* and has $O(k)$ complexity. Let $B = \{b_1(x), b_2(x), \dots, b_n(x)\}$ be a set of n continuous “blue” curves in the plane where each blue curve is monotone *increasing* and has $O(k)$ complexity. Let $V(k)$ be the time to compute the *value* of any red or blue curve at a given x -position, and let $I(k)$ be the time to find the *intersection* of any $r_i(x)$ and $b_j(x)$.

Assume all red and blue curves have $\min(r_i(x)) = y_{\min}$ for all $1 \leq i \leq m$ and $\max(b_j(x)) = y_{\max}$ for all $1 \leq j \leq n$. In addition, assume the (open) domain (α, β) contains no left endpoint of any red or blue curve.

Theorem 1: The number of intersections in the domain $[\alpha, \beta]$ between every red curve $r_i(x) \in R$ and all blue curves $b_j(x) \in B$ can be counted in $O(N(V(k) + \log N))$ total time, where $N = \max(m, n)$.

Proof Sketch: A key idea is that if $r_i(\alpha) \geq b_j(\alpha)$ at $x = \alpha$, then the curves $r_i(x)$ and $b_j(x)$ will intersect for some $x \geq \alpha$. This follows because $\min(r_i(x)) \leq \max(b_j(x))$. Intersections in the domain $[\alpha, \beta]$ can be counted by taking “snapshots” of the curve positions at the endpoints α and β . The α -snapshot is found by computing the values of all red and blue curves at α in $O(N * V(k))$ time and sorting these values to create the list L_α in $O(N \log N)$ time. The list L_β can be found similarly in $O(N(V(k) + \log N))$ time.

For counting queries, the sorted list L_α is preprocessed in $O(N)$ time by one linear scan over L_α . For each $r_i(\alpha) \in L_\alpha$, let $\Gamma(r_i(\alpha))$ be the

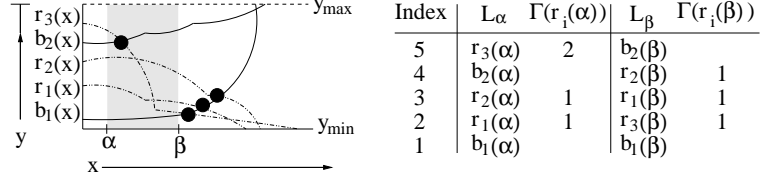


Fig. 1. Intersection counting calculates that $r_3(x)$ intersects two blue curves for $x \geq \alpha$ but only intersects one blue curve for $x \geq \beta$. Subtracting these quantities and accounting for $x = \beta$ intersections reveals that $r_3(x)$ must have one intersection in the domain $[\alpha, \beta]$.

number of $b_j(\alpha)$ such that $r_i(\alpha) > b_j(\alpha)$. Define $\Gamma(r_i(\beta))$ similarly (see Figure 1). In essence, each red curve keeps track of how many blue curves lie below it at $x = \alpha$ (resp. $x = \beta$) because these curves will intersect for some $x \geq \alpha$ (resp. $x \geq \beta$). For counting purposes, the number of $x \geq \alpha$ intersections minus the number of $x \geq \beta$ intersections yields the number of intersections for $\alpha \leq x < \beta$. Special situations such as $x = \beta$ intersections and curves with right endpoints inside (α, β) are not difficult to handle. \square

Theorem 2: The intersections in the domain $[\alpha, \beta]$ between every red curve $r_i(x) \in R$ and all $b_j(x) \in B$ can be reported in $O(N(V(k) + \log N + I(k)) + K)$ total time, where K is the total number of intersections reported.

Proof Sketch: The goal is to compute for every $r_i(x)$ a list of all $b_j(x)$ such that $r_i(x) \cap b_j(x)$ for $\alpha \leq x \leq \beta$. This can be done by building a balanced binary search tree from L_α and L_β . \square

Geodesic Fréchet Decision Problem

The non-geodesic decision problem solved by Alt and Godau [2] determines whether the Fréchet distance is at most a given $\varepsilon > 0$. The *free space* is defined as $FS_\varepsilon(A, B) = \{(s, t) \mid d(A(s), B(t)) \leq \varepsilon\} \subseteq [0, 1]^2$ where A and B are two polygonal curves. A free space cell $C \subseteq [0, 1]^2$ is the parameter space defined by two line segments $\overline{ab} \in A$ and $\overline{cd} \in B$, and the free space inside the cell is $FS_\varepsilon(\overline{ab}, \overline{cd}) = FS_\varepsilon(A, B) \cap C$.

Alt and Godau [2] solve the non-geodesic decision problem using a *free space diagram* which consists of all free space cells for all pairs of line segments of A and B . Their dynamic programming algorithm checks for the existence of a monotone path in the free space from $(0, 0)$ to $(1, 1)$ by propagating reachability information cell by cell through the free space.

In our technical report [5], we show that the free space inside a *geodesic* free space cell is x -monotone, y -monotone, and connected. As such, only the boundaries of a cell need to be computed to propagate reachability, and we show how to compute a cell boundary in $O(\log k)$ time using the algorithms of Guibas and Hershberger [6], [7].

Theorem 3: After a one-time preprocessing step of $O(k)$ time [6], the geodesic Fréchet decision problem for polygonal curves A and B inside a simple polygon P can be solved for any $\varepsilon \geq 0$ in $O(N^2 \log k)$ time and $O(k + N)$ space.

Proof Sketch: There are $O(N^2)$ cells in the free space diagram. Compute all cell boundaries in $O(N^2 \log k)$ time and propagate reachability information through all cells in $O(N^2)$ time. Return true if the upper right corner of the free space diagram is reachable. Return false otherwise.

The space bounds follow because only $O(1)$ space is needed per cell and dynamic programming only requires that two rows of cells reside in memory at any one time. The $O(k)$ term comes from storing the preprocessing structures of [6] throughout the algorithm’s execution. \square

*The full version of this paper is available as a technical report [5].

[†]This work has been supported by NSF grant CCF-0643597.

Let ε^* be the minimum value of ε such that the Fréchet decision problem returns true. That is, ε^* equals the Fréchet distance $\delta_F(A, B)$. Parametric Search is a technique commonly used to find ε^* (see [2], [4], and [10]). The typical approach to find ε^* is to sort all the cell boundary functions based on the unknown parameter ε^* . The comparisons performed during the sort guarantee that the result of the decision problem is known for all “critical values” [2] that could potentially define ε^* .

Previous sorting algorithms have operated on cell boundaries of constant complexity. The geodesic case is different because each cell boundary has $O(k)$ complexity. As a result, a straightforward parametric search based on sorting these values would require $O(kN^2 \log kN)$ time even when using Cole’s [4] optimization.¹

The seminal work of Alt and Godau [2] defines three types of critical values that are useful for computing the exact geodesic Fréchet distance. These critical values depend on the (at most one) free space interval on a cell boundary. The upper boundary of this interval is a monotone increasing function $b_{ij}(\varepsilon)$, and the lower boundary of this interval is a monotone decreasing function $a_{ij}(\varepsilon)$.

There are exactly two type (a) critical values between the start points and end points of the polygonal curves A and B . Type (b) critical values occur $O(N^2)$ times when $a_{ij}(\varepsilon) = b_{ij}(\varepsilon)$. Type (c) critical values occur $O(N^3)$ times when $a_{ij}(\varepsilon) = b_{kj}(\varepsilon)$.

The (at most one) intersection of $a_{ij}(\varepsilon)$ and $b_{kj}(\varepsilon)$ can be found in $O(\log k)$ time using monotonicity properties and the algorithms of Guibas and Hershberger [6], [7]. Hence, any type of critical value can be calculated in $O(\log k)$ time by using shortest paths and intersecting two monotone functions.

Randomized Algorithm: The below randomized algorithm solves the geodesic Fréchet optimization problem in $O(k + N^2 \log kN \log N)$ expected time. This is faster than the standard parametric search approach which requires $O(kN^2 \log kN)$ time. Resolving the type (a) and (b) critical values as a first step permits using red-blue intersection counting (cf. Theorem 1) to resolve the type (c) critical values.

- 1) Precompute and sort all type (a) and type (b) critical values. Run the decision problem $O(\log N)$ times to resolve these values. This shrinks the search domain for ε^* down to $[\alpha, \beta]$ in $O(N^2 \log k \log N)$ total time.
- 2) Count the number κ_j of type (c) critical values for each row j in the domain $[\alpha, \beta]$ using Theorem 1. Intersection counting requires $O(N \log kN)$ time per row for a total of $O(N^2 \log kN)$ time for all rows. Let C_j be the counting data structure for row j .
- 3) To achieve a fast *expected* runtime, pick a random intersection ϑ_j for each row j using C_j .²
- 4) To achieve a fast *worst-case* runtime, use C_j to find the curve $a_{Mj}(\varepsilon)$ in each row that has the most intersections. Add all intersections in $[\alpha, \beta]$ that involve $a_{Mj}(\varepsilon)$ to a global pool P of unresolved critical values³ and delete $a_{Mj}(\varepsilon)$ from any future consideration.
- 5) Find the median Ξ of the values in P . Also find the median Ψ of the $O(N)$ randomly selected ϑ_j .
- 6) Run the decision problem twice: once on Ξ and once on Ψ . This shrinks the search domain $[\alpha, \beta]$ and *always* halves the size of P . Repeat steps 2 through 6 until all *row*-based type (c) critical values have been resolved.
- 7) Resolve all *column*-based type (c) critical values in the same spirit as steps 2 through 6 and return the smallest critical value that satisfied the decision problem as the value of the geodesic Fréchet distance.

¹A variation of the general sorting problem called the “nuts and bolts” problem (see [8]) is tantalizingly close to an acceptable $O(N^2 \log N)$ sort, but it is not solvable in the general case.

²Picking a critical value at random is related to the distance selection problem [3] and is mentioned in [1], but to our knowledge, this alternative to parametric search has never been applied to the Fréchet distance.

³The idea of a global pool is similar to Cole’s [4] optimization for parametric search.

Theorem 4: The exact *geodesic* Fréchet distance between two polygonal curves A and B inside a simple bounding polygon P can be computed in $O(k + N^2 \log kN \log N)$ expected time and $O(k + N^3 \log kN)$ worst-case time, where N is the larger of the complexities of A and B and k is the complexity of P . $O(k + N^2)$ space is required.

Proof Sketch: The randomized algorithm executes $O(\log N)$ times in the expected case since it resolves a *random* critical value at each step. In the worst-case, each row can require at most $O(N)$ iterations (because each of the $O(N)$ $a_{ij}(\varepsilon)$ in a row can be picked as $a_{Mj}(\varepsilon)$). Since *all* rows are processed each iteration, the entire algorithm requires at most $O(N)$ iterations. A similar argument applies to *column*-based critical values.

The size of the pool P is expressed by the recurrence $S(x) = \frac{S(x-1) + O(N^2)}{2}$, where x is the current step number, and $S(0) = 0$. Intuitively, each step adds $O(N^2)$ values to P and then half of the values in P are always resolved. It is not difficult to see that $S(x) \in O(N^2)$ for any step number x .

Each iteration of the algorithm is dominated by the $O(N^2 \log kN)$ time spent on intersection counting. Consequently, the expected runtime is $O(k + N^2 \log kN \log N)$ and the worst-case runtime is $O(k + N^3 \log kN)$ including $O(k)$ preprocessing time [6] for geodesics inside P .

The preprocessing step of [6] requires $O(k)$ space, and this space must remain allocated throughout the algorithm. $O(N^2)$ additional space is sufficient for the remaining steps. \square

Non-Geodesic Fréchet Distance Runtime

Although the exact non-geodesic Fréchet distance is normally found in $O(N^2 \log N)$ time using parametric search (see [2]), parametric search is often regarded as impractical because it is difficult to implement⁴ and involves enormous constant factors [4]. To the best of our knowledge, our randomized algorithm is the first practical alternative to parametric search for solving the exact Fréchet optimization problem.

Theorem 5: The exact *non-geodesic* Fréchet distance between two polygonal curves A and B can be computed in $O(N^2 \log^2 N)$ expected time and $O(N^2)$ space, where N is the larger of the complexities of A and B .

Proof Sketch: The argument is very similar to the proof of Theorem 4. The main difference is that the randomized algorithm can compute distances in $O(1)$ time (instead of $O(\log k)$ time). \square

REFERENCES

- [1] Pankaj K. Agarwal and Micha Sharir. Efficient algorithms for geometric optimization. *ACM Comput. Surv.*, 30(4):412–458, 1998.
- [2] H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. volume 5, pages 75–91, 1995.
- [3] Sergei Bespamyatnikh and Michael Segal. Selecting distances in arrangements of hyperplanes spanned by points. volume 2, pages 333–345, September 2004.
- [4] Richard Cole. Slowing down sorting networks to obtain faster sorting algorithms. *J. ACM*, 34(1):200–208, 1987.
- [5] Atlas F. Cook IV and Carola Wenk. Geodesic Fréchet and Hausdorff distance inside a simple polygon. Technical Report CS-TR-2007-004, Department of Computer Science, University of Texas at San Antonio, August 2007.
- [6] L. J. Guibas and J. Hershberger. Optimal shortest path queries in a simple polygon. *J. Comput. Syst. Sci.*, 39(2):126–152, 1989.
- [7] John Hershberger. A new data structure for shortest path queries in a simple polygon. *Inf. Process. Lett.*, 38(5):231–235, 1991.
- [8] János Komlós, Yuan Ma, and Endre Szemerédi. Matching nuts and bolts in $(n \log n)$ time. In *SODA ’96: Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 232–241, Philadelphia, PA, USA, 1996. Society for Industrial and Applied Mathematics.
- [9] Joseph S. B. Mitchell, David M. Mount, and Christos H. Papadimitriou. The discrete geodesic problem. *SIAM J. Comput.*, 16(4):647–668, 1987.
- [10] René van Oostrum and Remco C. Veltkamp. Parametric search made practical. In *SCG ’02: Proceedings of the 18th Annual Symposium on Computational Geometry*, pages 1–9, New York, NY, USA, 2002. ACM Press.

⁴Quicksort-based parametric search has been implemented by van Oostrum and Veltkamp [10] using a complex framework.