

Geodesic Fréchet Distance Inside a Simple Polygon

Atlas F. Cook IV

University of Texas at San Antonio

and

Carola Wenk

University of Texas at San Antonio

We present an alternative to parametric search that applies to both the non-geodesic and geodesic Fréchet optimization problems. 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.

We introduce the first algorithm to compute the geodesic Fréchet distance between two polygonal curves A and B inside a simple bounding polygon P . The geodesic Fréchet *decision* problem is solved almost as fast as its non-geodesic sibling in $O(N^2 \log k)$ time and $O(k + N)$ space after $O(k)$ preprocessing, where N is the larger of the complexities of A and B and k is the complexity of P . The geodesic Fréchet *optimization* problem is solved by a randomized approach in $O(k + N^2 \log k N \log N)$ expected time and $O(k + N^2)$ space. This runtime is only a logarithmic factor larger than the standard non-geodesic Fréchet algorithm [Alt and Godau 1995]. Results are also presented for the geodesic Fréchet distance in a polygonal domain with obstacles and the geodesic Hausdorff distance for sets of points or sets of line segments inside a simple polygon P .

Categories and Subject Descriptors: F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems—*Geometrical Problems and Computations*

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Fréchet Distance, Geodesic, Shortest Path, Simple Polygon

1. INTRODUCTION

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 is a similarity metric for continuous shapes such as curves or surfaces which is defined using reparametrizations of the shapes. Since it takes the continuity of the shapes into account, it is generally a more appropriate distance measure than the often used Hausdorff distance. The Fréchet distance for curves is commonly illustrated by a person walking a dog on a leash [Alt and Godau 1995]. The person walks forward on one curve, and the dog walks forward on the other curve. As the person and dog move along their

Author Address: 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. Email: acook@cs.utsa.edu, carola@cs.utsa.edu
This work has been supported by the National Science Foundation grant NSF CAREER CCF-0643597.

Preliminary versions of this work appeared in [Cook IV and Wenk 2007a; 2007b; 2008].

Permission to make digital/hard copy of all or part of this material without fee for personal or classroom use provided that the copies are not made or distributed for profit or commercial advantage, the ACM copyright/server notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior specific permission and/or a fee.

© 2008 ACM 0000-0000/2008/0000-0001 \$5.00

respective curves, a leash is maintained to keep track of the separation between them. The Fréchet distance is the length of the *shortest* leash that makes it possible for the person and dog to walk from beginning to end on their respective curves without breaking the leash. See section 2 for a formal definition of the Fréchet distance.

Most previous work assumes an obstacle-free environment where the leash connecting the person to the dog has its length defined by an L_p metric. In [Alt and Godau 1995] the Fréchet distance between polygonal curves A and B is computed in arbitrary dimensions for obstacle-free environments in $O(N^2 \log N)$ time, where N is the larger of the complexities of A and B . Rote [Rote 2005] computes the Fréchet distance between piecewise smooth curves. Buchin et al. [Buchin et al. 2006] show how to compute the Fréchet distance between two simple polygons. Fréchet distance has also been used successfully in the practical realm of map matching [Wenk et al. 2006]. All these works assume a leash length that is defined by an L_p metric.

This paper's contribution is to measure the leash length by its geodesic distance inside a simple polygon P (instead of by its L_p distance). A few related works have also measured the leash by its geodesic distance. Maheshwari and Yi [Maheshwari and Yi 2005] compute the Fréchet distance for polygonal curves A and B on the surface of a convex polyhedron in $O(N^3 k^4 \log(kN))$ time. Chambers et al. [Chambers et al. 2008] compute the Fréchet distance in $O(N^4 k^3 \log kN)$ time by restricting the leash to different homotopy classes. The Fréchet distance has even been applied to morphing by considering the polygonal curves A and B to be obstacles that the leash must go around [Bespamyatnikh 2002; Efrat et al. 2002]. The morphing method works in $O(N^2)$ time but only applies when A and B are disjoint and lie on the boundary of a simple polygon. Our work can handle both this case and more general cases. We consider a simple polygon P to be the only obstacle and the curves, which may intersect each other or self-intersect, both lie inside P .

A core insight of this paper is that the free space in a geodesic cell (see section 2) is x -monotone, y -monotone, and connected. We show how to quickly compute the cell boundary and how to propagate reachability information through a cell in constant time. This is sufficient to solve the geodesic Fréchet decision problem. To solve the geodesic Fréchet optimization problem, we replace the standard parametric search approach by a novel and asymptotically faster (in the expected case) randomized algorithm that is based on red-blue intersection counting. Palazzi and Snoeyink [Palazzi and Snoeyink 1994] have previously explored red-blue intersections for line segments using a slab-based approach, but they require that all red segments are disjoint and all blue segments are disjoint (see also [Chazelle et al. 1994]). Our approach is more general because it applies to functions instead of line segments, and we have no disjointness requirement. Other randomized alternatives to parametric search have been mentioned in [Agarwal et al. 1994; Matoušek 1991].

We show that the geodesic Fréchet distance between two polygonal curves inside a simple bounding polygon can be computed in $O(k + N^2 \log kN \log N)$ expected time, where N is the larger of the complexities of A and B and k is the complexity of the simple polygon. The expected runtime is almost a quadratic factor in k faster than the straightforward approach, similar to [Efrat et al. 2002], of partitioning each cell into $O(k^2)$ subcells with combinatorially distinct shortest paths. It is notable that the randomized algorithm also applies to the non-geodesic Fréchet distance in arbitrary dimensions. We also present algorithms to compute the geodesic Fréchet distance in a polygonal domain with obstacles

and the geodesic Hausdorff distance for sets of points or sets of line segments inside a simple polygon.

2. PRELIMINARIES

Let k be the complexity of a simple polygon P that contains polygonal curves A and B in its interior. In general, a *geodesic* is a path that avoids all obstacles and cannot be shortened by slight perturbations [Mitchell et al. 1987]. Inside a simple polygon, a geodesic is a unique shortest path between two points. Let $\pi(a, b)$ denote the geodesic inside P between points a and b . The *geodesic distance* $d(a, b)$ is the length of a shortest path between a and b that does not cross the boundary of the obstacle polygon P , where the length of the path is measured by L_2 distance.

A *bitonic* function has at most one change in monotonicity. A function H is “ $\downarrow\uparrow$ -bitonic” when it decreases monotonically and then increases monotonically.

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, usually the L_2 distance, and in our setting the geodesic distance. For a given $\varepsilon > 0$ the *free space* is defined as $FS_\varepsilon(A, B) = \{(s, t) \mid d'(A(s), B(t)) \leq \varepsilon\} \subseteq [0, 1]^2$. 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 $FS_\varepsilon(\overline{ab}, \overline{cd}) = FS_\varepsilon(A, B) \cap C$.

The decision problem to check whether the Fréchet distance is at most a given $\varepsilon > 0$ is solved by Alt and Godau [Alt and Godau 1995] 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 setting, each point (s, t) in the free space diagram has an associated geodesic distance and a binary classification. The geodesic distance for (s, t) equals $d(f(s), g(t))$ for points $f(s) \in A, g(t) \in B$. The binary classification for (s, t) is decided by $d(f(s), g(t)) \leq \varepsilon$ and indicates whether (s, t) is a point in the free space. Unless otherwise indicated, geodesic distance is used throughout this paper as the underlying distance measure.

2.1 Funnels and Hourglasses

The associated geodesic distances for points in a free space cell C can be described by either the funnel or hourglass structure of [Guibas et al. 1986]. A funnel $\mathcal{F}_{p, \overline{cd}}$ describes all shortest paths between an apex point p and a line segment \overline{cd} , so it represents a horizontal (or vertical) line segment in a cell C . The boundary of $\mathcal{F}_{p, \overline{cd}}$ is $\overline{cd} \circ \pi(d, p) \circ \pi(p, c)$, where $\pi(d, p)$ and $\pi(p, c)$ are shortest path chains and \circ denotes concatenation.

An hourglass $\mathcal{H}_{\overline{ab}, \overline{cd}}$ describes all shortest paths between two line segments \overline{ab} and \overline{cd} and represents all distances in a cell C . For non-crossing segments, the boundary of the hourglass $\mathcal{H}_{\overline{ab}, \overline{cd}}$ is $\overline{ab} \circ \pi(a, c) \circ \overline{cd} \circ \pi(d, b)$, and for crossing line segments the boundary is $\pi(a, c) \circ \pi(c, b) \circ \pi(b, d) \circ \pi(d, a)$ (see Figure 1). Both funnel and hourglass boundaries have $O(k)$ complexity because shortest paths inside a simple polygon P are simple, polygonal, and only have corners at vertices of P [Guibas et al. 1987].

There are three types of hourglasses: open, closed, and intersecting. An *open hourglass* is defined by non-crossing \overline{ab} and \overline{cd} and two disjoint shortest path chains $\pi(a, c)$ and $\pi(d, b)$. A *closed hourglass* has non-crossing \overline{ab} and \overline{cd} and a collapsed interior such that $\pi(a, c)$ and $\pi(d, b)$ share a common polygonal sub-path. An *intersecting hourglass* has crossing \overline{ab} and \overline{cd} and four shortest path chains $\pi(a, c)$, $\pi(c, b)$, $\pi(b, d)$, and $\pi(d, a)$. Open, closed, and intersecting hourglasses are illustrated in Figure 1.

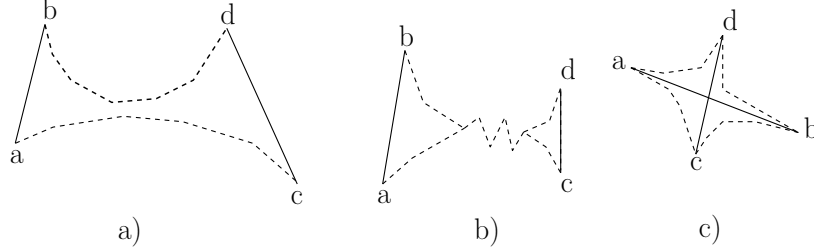


Fig. 1. a) An open hourglass, b) a closed hourglass, and c) an intersecting hourglass

Any horizontal or vertical line segment in C is associated with a funnel's distance function $F_{p, \overline{cd}} : [c, d] \rightarrow \mathbb{R}$ with $F_{p, \overline{cd}}(q) = d(p, q)$. The below three results are generalizations of Euclidean properties and will prove useful in section 3 for analyzing the structure of a free space cell.

LEMMA 2.1. $F_{p, \overline{cd}}$ is piecewise hyperbolic with $O(k)$ complexity and is $\downarrow\uparrow$ -bitonic.

PROOF. The shortest path $\pi(p, q)$ in a simple polygon between a fixed point p and any point $q \in \overline{cd}$ can be described by a funnel $\mathcal{F}_{p, \overline{cd}}$ with convex chains $\pi(d, p)$, $\pi(p, c)$ [Guibas et al. 1986]. By extending all line segments on these convex chains into lines and intersecting these lines with \overline{cd} , a partition of \overline{cd} into $O(k)$ intervals I_1, I_2, \dots, I_R is obtained (see Figure 2a).

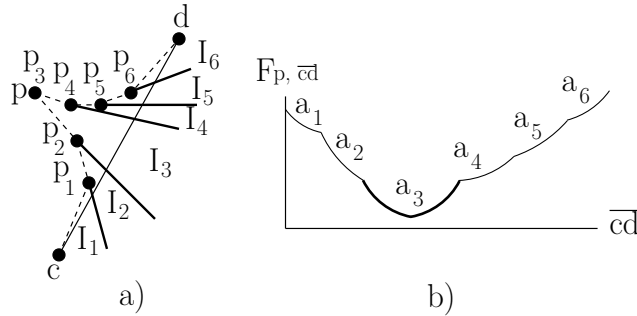


Fig. 2. a) A funnel $\mathcal{F}_{p, \overline{cd}}$ can be partitioned into $O(k)$ intervals such that chain vertex p_j defines interval I_j . b) The funnel's distance function $F_{p, \overline{cd}}$ is piecewise hyperbolic.

All shortest paths $\pi(p, q)$ from p to any point $q \in I_j$ are polygonal and have the form $p, p_i, p_{i+1}, \dots, p_j, q$ or $p, p_i, p_{i-1}, \dots, p_j, q$ where p_i, \dots, p_j are the funnel chain vertices visited by $\pi(p, q)$. For example, all shortest paths from p to $q \in I_1$ in Figure 2 have the

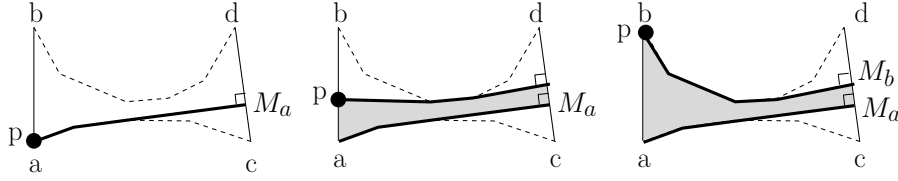


Fig. 3. Shortest paths in the hourglass $\mathcal{H}_{\overline{ab}, \overline{cd}}$ define $H_{\overline{ab}, \overline{cd}}$.

form p, p_2, p_1, q . Let L be the length of the path from p to q so that $d(p, q) = L + ||p_j - q||$. As q varies along I_j , L and p_j are fixed, so $d(p, q)$ equals the L_2 -distance from a point to a line segment (plus the constant L). The graph of $d(p, q)$ for $q \in I_j$ therefore defines a hyperbolic arc α_j . Hence, $F_{p, \overline{cd}}$ is piecewise hyperbolic with $O(k)$ complexity.

Observe that the *slopes* of the line segments defining the funnel chains $\pi(c, p)$ and $\pi(p, d)$ in order from c to p to d form a monotone sequence. This follows from the convexity of each chain. We now show that at most one arc of $F_{p, \overline{cd}}$ is bitonic, and the remaining arcs are monotone.

Each interval I_j for $1 \leq j \leq R$ is defined by two rays R_{j-1} and R_j that originate from the chain vertex p_j and intersect \overline{cd} . Arc α_j is bitonic if and only if the perpendicular from p_j to the line ζ through \overline{cd} lies strictly in the interior of I_j . Otherwise, α_j is monotone. Note that the slope μ of any perpendicular to ζ is a constant. Since I_j is defined by two rays R_{j-1}, R_j from p_j to ζ , a perpendicular from p_j will only intersect ζ in I_j when the slope μ lies between the slopes of R_{j-1} and R_j . Since the ray slopes are monotone through the intervals $I_{1..R}$, at most one bitonic arc α_v for $1 \leq v \leq R$ exists, and the remaining arcs are monotone. The ray slopes also ensure that the arcs $\alpha_{1..(v-1)}$ are monotone decreasing and the arcs $\alpha_{(v+1)..R}$ are monotone increasing. Hence, $F_{p, \overline{cd}}$ is $\downarrow\uparrow$ -bitonic. \square

COROLLARY 2.2. *Any horizontal (or vertical) line segment in a geodesic¹ free space cell C has at most one connected set of free space values.*

PROOF. A horizontal (or vertical) line segment in a cell C is associated with a funnel's distance function $F_{p, \overline{cd}}$, and free space consists of all values less than or equal to a given distance ε . Since Lemma 2.1 ensures that $F_{p, \overline{cd}}$ is $\downarrow\uparrow$ -bitonic, $F_{p, \overline{cd}}$ has at most one connected set of free space values. \square

Consider the hourglass $\mathcal{H}_{\overline{ab}, \overline{cd}}$ in Figure 3. As p varies from a to b , the *minimum* distance from p to \overline{cd} defines a function $H_{\overline{ab}, \overline{cd}} : [a, b] \rightarrow \mathbb{R}$ with $H_{\overline{ab}, \overline{cd}}(p) = \min_{q \in [c, d]} d(p, q)$.

LEMMA 2.3. $H_{\overline{ab}, \overline{cd}}$ is $\downarrow\uparrow$ -bitonic.

PROOF. Let the *shortest* distance from a to any point on \overline{cd} occur at $M_a \in \overline{cd}$. Similarly, let the shortest distance from b to any point on \overline{cd} occur at $M_b \in \overline{cd}$. Observe that $H_{\overline{ab}, \overline{cd}}$ and $H_{\overline{ab}, \overline{M_a M_b}}$ are identical functions (see Figure 3). We now show that $H_{\overline{ab}, \overline{M_a M_b}}$ is $\downarrow\uparrow$ -bitonic regardless of whether the hourglass $\mathcal{H}_{\overline{ab}, \overline{M_a M_b}}$ is open, closed, or intersecting (cf. Figure 1).

Suppose that $\mathcal{H}_{\overline{ab}, \overline{M_a M_b}}$ is an open hourglass. If $M_a = M_b$, then observe that the

¹A *geodesic* refers to a shortest path in a simple polygon.

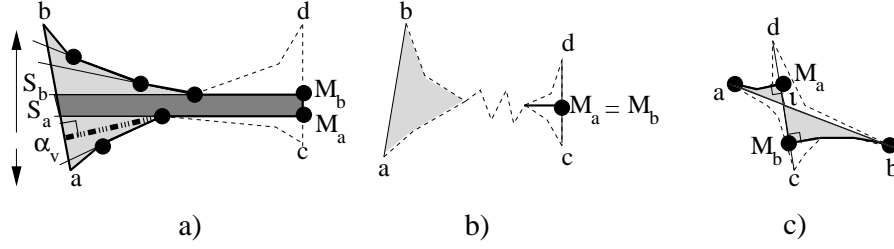


Fig. 4. a) An *open* hourglass $\mathcal{H}_{\overline{ab}, \overline{M_a M_b}}$. $\mathcal{F}_{\overline{a S_a}, M_a}$ and $\mathcal{F}_{\overline{S_b b}, M_b}$ are lightly shaded; \mathcal{L} is heavily shaded. b) A *closed* hourglass always has $M_a = M_b$. c) An *intersecting* hourglass can be split into two (shaded) open hourglasses.

hourglass distance function $H_{\overline{ab}, \overline{M_a M_b}}$ equals the funnel distance function $\mathcal{F}_{\overline{ab}, M_a}$ ² and is $\downarrow\uparrow$ -bitonic by Lemma 2.1. When $M_a \neq M_b$ the convexity and disjointness of the open hourglass chains ensures that perpendiculars to \overline{cd} through M_a and M_b intersect \overline{ab} in two points S_a and S_b (see Figure 4a). Using S_a and S_b , the hourglass $\mathcal{H}_{\overline{ab}, \overline{M_a M_b}}$ can be split into three parts: two funnels $\mathcal{F}_{\overline{a S_a}, M_a}$, $\mathcal{F}_{\overline{S_b b}, M_b}$, and an L_2 -section \mathcal{L} that lies in-between the two funnels. These three structures together form a $\downarrow\uparrow$ -bitonic sequence for any open hourglass by the proof of Lemma 2.1 because the line segment slopes on the chains form a monotone sequence from a to M_a along $\pi(a, M_a)$ and continuing from M_b to b along $\pi(M_b, b)$ (see Figure 4a). Since the three structures define $H_{\overline{ab}, \overline{M_a M_b}}$, it is also $\downarrow\uparrow$ -bitonic.

Suppose that $\mathcal{H}_{\overline{ab}, \overline{M_a M_b}}$ is a closed hourglass. All shortest paths from $p \in \overline{ab}$ to the closest $q \in \overline{cd}$ will end at the same point M_a as shown in Figure 4b. Hence, $H_{\overline{ab}, \overline{M_a M_b}}$ equals $\mathcal{F}_{\overline{ab}, M_a}$ and is $\downarrow\uparrow$ -bitonic by Lemma 2.1.

When $\mathcal{H}_{\overline{ab}, \overline{M_a M_b}}$ is an intersecting hourglass, \overline{ab} and \overline{cd} will cross at the point ι as illustrated in Figure 4c. $H_{\overline{ab}, \overline{M_a M_b}}$ has the form $H_{\overline{a\iota}, \iota M_a} \circ H_{\overline{\iota b}, M_b \iota}$, where $H_{\overline{a\iota}, \iota M_a}$ and $H_{\overline{\iota b}, M_b \iota}$ are distance functions for *open* hourglasses. By the above arguments on open hourglasses, $H_{\overline{a\iota}, \iota M_a}$ and $H_{\overline{\iota b}, M_b \iota}$ are each (at worst) $\downarrow\uparrow$ -bitonic. It follows from $d(\iota, \iota) = 0$ that $H_{\overline{ab}, \overline{cd}}$ is $\downarrow\uparrow$ -bitonic. \square

3. GEODESIC CELL PROPERTIES

Consider a geodesic cell C for polygonal curves A and B inside a simple polygon. Let $\overline{ab} \in A$ and $\overline{cd} \in B$ be the two line segments defining C . A subset $S \subseteq \mathbb{R}^2$ is x -monotone if every vertical line intersects R in at most one connected interval. $S \subseteq \mathbb{R}^2$ is y -monotone if every horizontal line intersects R in at most one connected interval.

LEMMA 3.1. *For any $\varepsilon \geq 0$, the free space in a geodesic cell C is connected, x -monotone, and y -monotone.*

PROOF. The monotonicity of the free space in C follows from Corollary 2.2. For connectedness, choose any two free space points (p_1, q_1) , (p_2, q_2) , and construct a path connecting them in the free space as follows: move vertically from (p_1, q_1) to the minimum distance in C on the vertical line defined by p_1 . Similarly, move vertically from (p_2, q_2) to the minimum associated distance in C on the vertical line defined by p_2 . By Lemma

²Notice that the funnel $\mathcal{F}_{\overline{ab}, M_a}$ uses the *second* subscript for the apex. This emphasizes that the apex occurs on \overline{cd} instead of on \overline{ab} .

2.1, this movement causes the distance to decrease monotonically. By Lemma 2.3, any two minimum points are connected by a $\downarrow\uparrow$ -bitonic distance function $H_{\overline{ab}, \overline{cd}}$ (cf. section 2.1), but as the starting points are in the free space – and therefore have distance at most ε – all points on this constructed path lie in the free space. \square

The boundary of a free space cell consists of vertical and horizontal line segments. Each boundary segment can be associated with a funnel $\mathcal{F}_{p, \overline{cd}}$ that has a $\downarrow\uparrow$ -bitonic distance function $F_{p, \overline{cd}}$ (cf. Lemma 2.1). Given $\varepsilon \geq 0$, computing the free space on a boundary segment requires finding the (at most two) values t_1, t_2 such that $F_{p, \overline{cd}}(t_1) = F_{p, \overline{cd}}(t_2) = \varepsilon$ (see Figure 5).

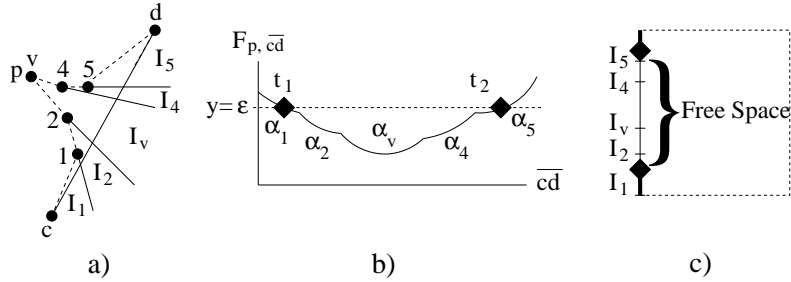


Fig. 5. a) A funnel $\mathcal{F}_{p, \overline{cd}}$. b) The funnel’s bitonic distance function $F_{p, \overline{cd}}$. c) Free space on a boundary segment of a free space cell. At most two values t_1, t_2 with $F_{p, \overline{cd}}(t_1) = F_{p, \overline{cd}}(t_2) = \varepsilon$ define the free space on this boundary segment.

LEMMA 3.2. *Both the minimum value of $F_{p, \overline{cd}}$ and the (at most two) values t_1, t_2 such that $F_{p, \overline{cd}}(t_1) = F_{p, \overline{cd}}(t_2) = \varepsilon$ can be found for any $\varepsilon \geq 0$ and any $F_{p, \overline{cd}}$ in $O(\log k)$ time after $O(k)$ preprocessing. Hence, the free space on the boundary of a cell can be computed in $O(\log k)$ time.*

PROOF. Although an explicit construction of $F_{p, \overline{cd}}$ would take $O(k)$ time, realize that it is not necessary to explicitly construct *all* the arcs of $F_{p, \overline{cd}}$. Instead, a binary search can find the intersections t_1 and t_2 of $\alpha_{1..R}$ with $y = \varepsilon$ by examining only $O(\log k)$ arcs.

Any shortest path chain $\pi(p, q)$ can be represented as a balanced binary tree \mathcal{T} in $O(\log k)$ time (after $O(k)$ preprocessing) by the algorithms of Guibas and Hershberger [Guibas and Hershberger 1989; Hershberger 1991]. \mathcal{T} supports binary searches because it stores chain edges at its nodes such that “the edges along the chain, taken in order, are the same as the edges stored in the nodes, taken in symmetric order” [Hershberger 1991]. Even though \mathcal{T} can have $O(k)$ complexity, it can be constructed in only $O(\log k)$ time by merging preconstructed trees together at query time [Guibas and Hershberger 1989].

Construct the binary search trees \mathcal{T}_c and \mathcal{T}_d for the two shortest path chains $\pi(d, p)$ and $\pi(p, c)$, respectively. These chains together define all arcs α_j of the piecewise hyperbolic distance function $F_{p, \overline{cd}}$, where $1 \leq j \leq R$. The bitonic arc α_v of $F_{p, \overline{cd}}$ can be found by searching \mathcal{T}_c and \mathcal{T}_d for the arc α_j with the smallest value among $\alpha_{1..R}$. At most one bitonic arc exists because \mathcal{T}_c and \mathcal{T}_d together represent the chains of the funnel $\mathcal{F}_{p, \overline{cd}}$ (cf. Lemma 2.1). After finding α_v , t_1 and t_2 can be found by binary searches over at most three monotone sequences of arcs defined by \mathcal{T}_c and \mathcal{T}_d . \square

COROLLARY 3.3. *$F_{p, \overline{cd}}$ can be evaluated at any $\varepsilon \geq 0$ in $O(\log k)$ time.*

PROOF. This follows immediately from the balanced binary tree \mathcal{T} representation for $F_{p, \overline{cd}}$ that is due to Guibas and Hershberger [Guibas and Hershberger 1989; Hershberger 1991]. \square

3.1 Propagating Reachability Information Through a Cell in $O(1)$ Time

Reachability information determines which points on the cell boundary are reachable by a *monotone* path that originates from the bottom-left corner of the free space diagram and travels only through the free space. Dynamic programming is traditionally used to propagate reachability information through each cell to solve the Fréchet decision problem [Alt and Godau 1995].

LEMMA 3.4. *Given the free space on the boundaries of a cell C and given reachability information for the bottom and left boundaries of C , reachability information can be propagated through C in constant time.*

PROOF. The standard argument used for convex free space [Alt and Godau 1995] still applies. By Lemma 3.1, the free space is x -monotone, y -monotone, and connected. Hence, if some point on the left boundary is reachable, then the whole top boundary is reachable. If no point on the left boundary is reachable, then project the reachable points from the bottom boundary onto the top boundary. Analogous arguments hold for the bottom and right boundaries. \square

4. RED-BLUE INTERSECTIONS

In this section we show how to efficiently count and report a certain type of red-blue intersections in the plane. This problem is interesting both from theoretical and applied stances and will prove useful in section 5.2 for the Fréchet optimization problem.

Let $R = \{r_1, \dots, r_m\}$ and $B = \{b_1, \dots, b_n\}$ be sets of continuous functions such that $r_i, b_j : [\alpha, \beta] \rightarrow \mathbb{R}$ for $1 \leq i \leq m$ and $1 \leq j \leq n$. Assume every “red” function r_i is strictly *decreasing* and every “blue” function b_j is strictly *increasing*. Let $I(k)$ be the time to find the at most one intersection of the functions r_i and b_j ,³ and assume r_i, b_j can be evaluated at any position in $E(k)$ time.

THEOREM 4.1. *The number of red-blue intersections between R and B in the slab $[\alpha, \beta] \times \mathbb{R}$ can be counted in $O(N(E(k) + \log N))$ total time, where $N = \max(m, n)$. These intersections can be reported in $O(N(E(k) + \log N) + K \cdot I(k))$ total time, where K is the total number of intersections reported. After $O(N(E(k) + \log N))$ preprocessing time, a random red-blue intersection in $[\alpha, \beta] \times \mathbb{R}$ can be returned in $O(\log N + I(k))$ time, and the red function involved in the most red-blue intersections in $[\alpha, \beta] \times \mathbb{R}$ can be returned in $O(1)$ time. All operations require $O(N)$ space.*

PROOF. Figure 6 illustrates the key idea. Suppose a red function r_3 lies *above* a blue function b_2 at $x = \alpha$. If it is also true that r_3 lies *below* b_2 at $x = \beta$, then r_3 and b_2 must intersect in the slab $[\alpha, \beta] \times \mathbb{R}$ due to the continuity and monotonicity of r_3 and b_2 . All red-blue intersections in the slab $[\alpha, \beta] \times \mathbb{R}$ can be counted by taking *snapshots* of the functions at $x = \alpha$ and $x = \beta$. Let the α -snapshot L_α be the list of functions in the order they intersect the vertical line $x = \alpha$. Let L_β be the β -snapshot at $x = \beta$. These snapshots

³There is at most one intersection due to the monotonicities of the red and blue functions.

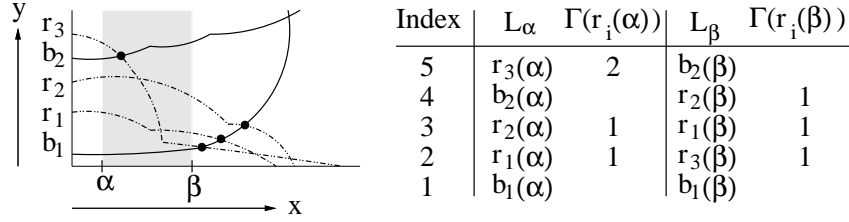


Fig. 6. r_3 lies above *two* blue functions at $x = \alpha$ but only lies above *one* blue function at $x = \beta$. Subtraction reveals that r_3 has one intersection in the slab $[\alpha, \beta] \times \mathbb{R}$.

can be computed in $O(N(E(k) + \log N))$ time by computing and sorting red and blue values.⁴

Let $\Gamma(r_i(x))$ be the number of $b_j \in B$ such that $r_i(x) > b_j(x)$ at a given value of x . The number of red-blue intersections for each r_i in the slab $[\alpha, \beta] \times \mathbb{R}$ is simply $\Gamma(r_i(\alpha)) - \Gamma(r_i(\beta))$ due to the continuity and monotonicity of the red and blue functions (see Figure 6). All $\Gamma(r_i(\alpha))$ (resp. $\Gamma(r_i(\beta))$) values can be computed by a linear scan over L_α (resp. L_β). Intersection counting simply sums up the number of intersections over all red functions, and this process also reveals which red function is involved in the most red-blue intersections in the slab $[\alpha, \beta] \times \mathbb{R}$.

We now *report* intersections instead of *counting* them. Note that we avoid enumerating all possible intersections and instead use the idea of Figure 6 to only compute intersections that are actually reported. An intersection in the slab $[\alpha, \beta] \times \mathbb{R}$ occurs when $r_i(\alpha) \geq b_j(\alpha)$ and $r_i(\beta) \leq b_j(\beta)$. To test these conditions, *incrementally* construct a balanced binary search tree T . Let $I_\beta(r_i)$ denote the rank of r_i in the list L_β , and let $I_\beta(b_i)$ denote the rank of b_i in L_β . Begin with an empty tree T , and march through L_α in ascending order (i.e., bottom-to-top in Figure 6). During this traversal, process each b_j by inserting $I_\beta(b_j)$ into T in $O(\log N)$ time. Process each r_i by querying the tree T with $I_\beta(r_i)$. At query time for r_i , T contains only indices for b_j such that $r_i(\alpha) \geq b_j(\alpha)$. All b_j with indices in T greater than the query index must intersect r_i because both conditions $r_i(\alpha) \geq b_j(\alpha)$ and $r_i(\beta) \leq b_j(\beta)$ are satisfied.

To find a *random* red-blue intersection in the slab $[\alpha, \beta] \times \mathbb{R}$, count the number κ of red-blue intersections in $[\alpha, \beta] \times \mathbb{R}$, and pick a random integer ρ between 1 and κ . To find the red curve r_i that is involved in the ρ th red-blue intersection, preprocess each r_i in L_α to store the total number λ of red-blue intersections for *all* r_j that lie below r_i in L_α and perform a binary search. Once r_i is identified, we must find the $(\rho - \lambda)$ th blue curve that intersects r_i in the slab $[\alpha, \beta] \times \mathbb{R}$. Recall that the reporting structure T allows finding all intersections involving r_i if queries are made at an appropriate time during an incremental construction. Sarnak and Tarjan [Sarnak and Tarjan 1986] have shown how to build a *persistent* tree in $O(N \log N)$ time such that any previous version of a tree after i' insertions is available in $O(\log N)$ time. Precompute T as a persistent tree. At query time, find the tree T' that defines T after $\Gamma(r_i(\alpha))$ insertions. Search T' to identify the subtree such that every node in the subtree represents a red-blue intersection involving r_i

⁴During the sorting process, if a red function has the same value as a blue function at $x = \alpha$, then the red function should be considered greater than the blue function. By contrast, if a red function has the same value as a blue function at $x = \beta$, then the red function should be considered less than the blue function. This ensures that intersections directly at $x = \alpha$ or $x = \beta$ are counted and reported properly.

in $[\alpha, \beta] \times \mathbb{R}$. The $(\rho - \lambda)$ th node in this subtree is the desired intersection. Hence, a persistent [Sarnak and Tarjan 1986] version of the reporting structure allows a random red-blue intersection in the slab $[\alpha, \beta] \times \mathbb{R}$ to be returned in $O(\log N + I(k))$ query time after $O(N(E(k) + \log N))$ preprocessing time. \square

5. GEODESIC FRÉCHET ALGORITHM

5.1 Geodesic Fréchet Decision Problem

THEOREM 5.1. *After preprocessing a simple polygon P for shortest path queries in $O(k)$ time [Guibas and Hershberger 1989], the geodesic Fréchet decision problem (see section 2) for polygonal curves A and B inside P can be solved for any $\varepsilon \geq 0$ in $O(N^2 \log k)$ time and $O(k + N)$ space.*

PROOF. Following the standard dynamic programming approach of [Alt and Godau 1995], compute all cell boundaries in $O(N^2 \log k)$ time (cf. Lemma 3.2), and propagate reachability information through all cells in $O(N^2)$ time (cf. Lemma 3.4). $O(k)$ space is needed for the preprocessing structures of [Guibas and Hershberger 1989], and only $O(N)$ space is needed for dynamic programming if two rows of the free space diagram are stored at a time. \square

5.2 Geodesic Fréchet Optimization Problem

For a given ε , a cell boundary segment of a cell C_{ij} has at most one free space interval (cf. Lemma 2.1). Denote the lower boundary of this interval as $a_{ij}(\varepsilon)$ and the upper boundary as $b_{ij}(\varepsilon)$ (see Figure 7). Let $\varepsilon^* = \delta_F(A, B)$ be the minimum value of ε such that the Fréchet decision problem returns true. Parametric search is a technique commonly used to find ε^* (see [Agarwal et al. 1994; Alt and Godau 1995; Cole 1987; van Oostrum and Veltkamp 2002]).⁵ Using parametric search, ε^* is found by sorting all the functions $a_{ij}(\varepsilon), b_{ij}(\varepsilon)$ 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” [Alt and Godau 1995] that could potentially define ε^* . Traditionally, such a sort operates on functions of constant complexity. The geodesic case is different because $a_{ij}(\varepsilon), b_{ij}(\varepsilon)$ have $O(k)$ complexity (cf. Lemma 2.1). A straightforward parametric search based on sorting $O(kN^2)$ constant complexity hyperbolic functions would require $O(kN^2 \log kN)$ time even when using Cole’s [Cole 1987] optimization.⁶ We present a randomized algorithm with expected runtime $O(k + N^2 \log kN \log N)$.

The seminal work of Alt and Godau [Alt and Godau 1995] defines three types of critical values which are candidate values of ε for ε^* , and these candidate values also apply to the setting for a simple polygon. There are exactly two type (a) critical values associated with distances between the starting points of A and B and the ending points of A and B . Type (b) critical values occur $O(N^2)$ times when $a_{ij}(\varepsilon) = b_{ij}(\varepsilon)$. See Figure 7a. The $O(N^2)$ type (a) and (b) critical values can be handled in $O(N^2 \log k \log N)$ total time by computing the critical values in $O(N^2 \log k)$ time,⁷ sorting the values in $O(N^2 \log N)$

⁵A simple alternative to parametric search is to run the decision problem once for every bit of accuracy that is desired. This approach runs in $O(BN^2 \log k)$ time and $O(k + N)$ space, where B is the desired number of bits of accuracy [van Oostrum and Veltkamp 2002].

⁶A variation of the general sorting problem called the “nuts and bolts” problem (see [Kömlös et al. 1996]) is tantalizingly close to an acceptable $O(N^2 \log N)$ sort but does not apply to our setting.

⁷Both of the type (a) critical values can be computed in $O(\log k)$ time by computing shortest paths between the

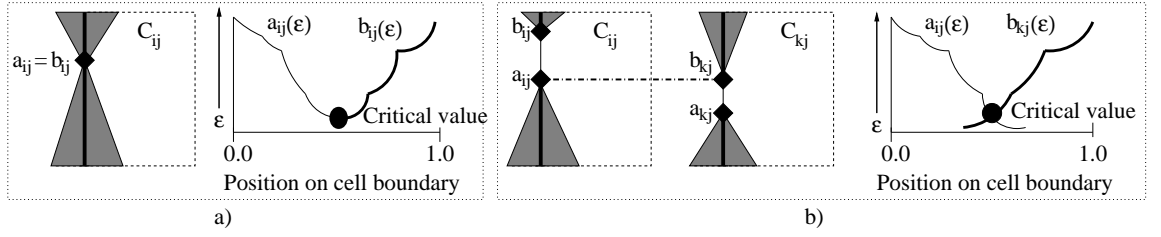


Fig. 7. a) Free space diagram and distance function for a type (b) critical value of ε . b) Free space diagram and distance function for a type (c) critical value.

time, and finally performing a binary search using the decision problem. Resolving the type (a) and (b) critical values as a first step defines a search interval for ε^* that simplifies the randomized algorithm for the type (c) critical values.

Alt and Godau [Alt and Godau 1995] show that type (c) critical values occur when the position of $a_{ij}(\varepsilon)$ in a cell C_{ij} equals the position of $b_{kj}(\varepsilon)$ in cell C_{kj} in the free space diagram. See Figure 7b. As ε increases, Lemma 2.1 ensures that $a_{ij}(\varepsilon)$ is monotone decreasing on the cell boundary segment and $b_{ij}(\varepsilon)$ is monotone increasing (see Figure 7b). This means that $a_{ij}(\varepsilon)$ and $b_{kj}(\varepsilon)$ intersect at most once. Hence, there are $O(N^2)$ intersections of $a_{ij}(\varepsilon)$ and $b_{kj}(\varepsilon)$ in row j and a total of $O(N^3)$ type (c) critical values over all rows. There are also $O(N^2)$ intersections of $a_{ij}(\varepsilon)$ and $b_{ik}(\varepsilon)$ in column i and a total of $O(N^3)$ additional type (c) critical values over all columns.

LEMMA 5.2. *The intersection of $a_{ij}(\varepsilon)$ and $b_{kl}(\varepsilon)$ can be found for any $\varepsilon \geq 0$ in $O(\log k)$ time after preprocessing.*

PROOF. Recall that $a_{ij}(\varepsilon)$ is defined by the monotone decreasing component of a distance function $F_{p, \overline{cd}}$ and is composed of arcs that correspond to a subset of $\pi(d, p) \cup \pi(p, c)$. Similarly, $b_{kl}(\varepsilon)$ is defined by the monotone increasing component of $F_{p, \overline{cd}}$. Using the approach of section 3, construct the balanced binary search trees \mathcal{T}_a and \mathcal{T}_b in $O(\log k)$ time that are, respectively, associated with the shortest path chains for $a_{ij}(\varepsilon)$ and $b_{kl}(\varepsilon)$. Recall that these trees support binary searches because they store chain edges at their nodes such that “the edges along the chain, taken in order, are the same as the edges stored in the nodes, taken in symmetric order” [Hershberger 1991]. We show that a logarithmic search over \mathcal{T}_a and \mathcal{T}_b is sufficient to find the intersection of $a_{ij}(\varepsilon)$ and $b_{kl}(\varepsilon)$ or report that no intersection exists.

Start at the roots of both trees and build the arcs α_a, α_b for the current nodes in $\mathcal{T}_a, \mathcal{T}_b$ in $O(1)$ time. If α_a and α_b intersect, then we are done. Otherwise, we need to move left or right in one of the trees such that no intersection is missed. The monotone decreasing nature of $a_{ij}(\varepsilon)$ ensures that all arcs left of α_a in the tree \mathcal{T}_a must lie in the space defined by an infinite rectangular wedge r_{a1} , where r_{a1} is defined by the upper left quadrant with origin at the left endpoint of α_a . Similarly, r_{a2} is defined by the lower right quadrant with origin at the right endpoint of α_a , r_{b1} is defined by the lower left quadrant with origin at the left endpoint of α_b , and r_{b2} is defined by the upper right quadrant with origin at the right endpoint of α_b (see Figure 8). Let \mathcal{A} be the “red” wedge-arc-wedge sequence defined by

starting and ending points of A and B . Each type (b) critical value occurs at the minimum value of $F_{p, \overline{cd}}$ and can be computed in $O(\log k)$ time by Lemma 3.2.

r_{a1} , α_a , and r_{a2} . Let \mathcal{B} be the “blue” wedge-arc-wedge sequence defined by r_{b1} , α_b , and r_{b2} . The monotonicities of \mathcal{A} and \mathcal{B} ensure that at least one of the wedges will not have its boundary crossed by the other color. Hence, at least one of the wedges is guaranteed not to involve an intersection and can be discarded, so a binary search is sufficient to find an intersection if it exists.

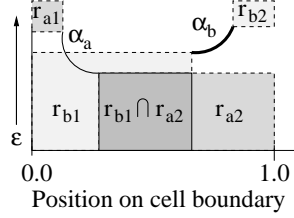


Fig. 8. The intersection of $a_{ij}(\varepsilon)$ and $b_{kl}(\varepsilon)$ does not involve r_{a1} , α_b , and r_{b2} .

The runtime follows because each step performs constant work on four wedges and two arcs to perform a binary search. Since the binary search trees \mathcal{T}_a and \mathcal{T}_b have $O(\log k)$ height, the total number of steps is $O(\log k)$. Hence, the intersection of $a_{ij}(\varepsilon)$ and $b_{kl}(\varepsilon)$ can be found (or determined not to exist) in $O(\log k)$ time. \square

The below randomized algorithm uses Theorem 4.1 to count type (c) critical values for the $a_{ij}(\varepsilon)$ and $b_{kl}(\varepsilon)$ functions and solves the geodesic Fréchet optimization problem in $O(k + N^2 \log k N \log N)$ expected time. This is faster than the standard parametric search approach which requires $O(kN^2 \log kN)$ time. The idea is to examine a *random* critical value to achieve a fast expected runtime. To improve the worst-case time, we also use Cole’s [Cole 1987] pool idea to batch unresolved comparisons in step (4). Although the a_{ij} and b_{kl} functions have $O(k)$ complexity, Corollary 3.3 ensures that these functions can be evaluated at any ε in $O(\log k)$ time, and Lemma 5.2 ensures that the (at most one) intersection of a_{ij} and b_{kl} can also be found in $O(\log k)$ time.

ALGORITHM 5.3. *Randomized Fréchet Distance*

- (1) Precompute and sort all type (a) and type (b) critical values in $O(N^2 \log kN)$ time (cf. Corollary 3.3 and Lemma 3.2). Run the decision problem $O(\log N)$ times to resolve these $O(N^2)$ values and shrink the search interval for ε^* down to $[\alpha, \beta]$ in $O(N^2 \log k \log N)$ time.
- (2) Count the number κ_j of type (c) critical values for each row j in the slab $[\alpha, \beta] \times \mathbb{R}$ using Theorem 4.1. Let C_j be the resulting counting data structure for row j .
- (3) Pick a random intersection ϑ_j for each row using C_j (see Theorem 4.1).⁸ Find the median Ψ of the $O(N)$ randomly selected ϑ_j in $O(N)$ time using a *weighted* median algorithm, where weights are based on the number of critical values κ_j for each row j .

⁸Picking a critical value at random is related to the distance selection problem [Bespamyatnikh and Segal 2004] and is mentioned in [Agarwal et al. 1994], but to our knowledge this alternative to parametric search has never been applied to the Fréchet distance.

- (4) Use C_j to find the curve $a_{M_j}(\varepsilon)$ in each row that has the most intersections (see Theorem 4.1). Add all intersections in $[\alpha, \beta] \times \mathbb{R}$ that involve $a_{M_j}(\varepsilon)$ to a global pool \mathcal{P} of unresolved critical values⁹ and delete $a_{M_j}(\varepsilon)$ from any future consideration. Find the median Ξ of the values in \mathcal{P} in $O(N^2)$ time using the standard median algorithm mentioned in [Koslós et al. 1996].
- (5) Run the decision problem twice: once on Ξ and once on Ψ . This shrinks the search interval for ε^* and allows *at least* half the values in \mathcal{P} to be discarded because the result of the decision problem is known for them. Repeat steps 2 through 5 until all *row*-based type (c) critical values have been resolved.
- (6) Resolve all *column*-based type (c) critical values in the same spirit as steps 2 through 5 and return the smallest critical value that satisfied the decision problem as the value of the geodesic Fréchet distance.

THEOREM 5.4. *The 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 used.*

PROOF. Preprocess P once for shortest path queries in $O(k)$ time [Guibas and Hershberger 1989]. In the expected case, each execution of the decision problem will eliminate a constant fraction of the remaining type (c) critical values due to the proof of Quicksort's expected runtime and the weighted median approach for Ψ . Consequently, the expected number of iterations of the algorithm is $O(\log N^3) = O(\log N)$.

In the worst-case, each of the $O(N)$ $a_{ij}(\varepsilon)$ in a row will be picked as $a_{M_j}(\varepsilon)$. Therefore, each row can require at most $O(N)$ iterations. Since *all* rows are processed in each iteration, the entire algorithm requires at most $O(N)$ iterations for *row*-based critical values. By a similar argument, *column*-based critical values also require at most $O(N)$ iterations.

The size of the pool \mathcal{P} is expressed by the inequality $S(x) \leq \frac{S(x-1) + O(N^2)}{2}$, where $S(0) = 0$ and x is the iteration number for the loop involving steps 2 through 5. Intuitively, each iteration adds $O(N^2)$ values to \mathcal{P} and then at least half of the values in \mathcal{P} are always resolved using the median Ξ . It is not difficult to show that $S(x) \in O(N^2)$ for any $x \geq 0$.

Each iteration of the algorithm requires intersection counting and intersection calculations for $O(N)$ rows (or columns) at a cost of $O(N^2 \log kN)$ time. In addition, the global pool \mathcal{P} has its median calculated in $O(N^2)$ time, and the decision problem is executed in $O(N^2 \log k)$ time. 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 [Guibas and Hershberger 1989] for geodesics. The preprocessing structures use $O(k)$ space that must remain allocated throughout the algorithm, and the pool \mathcal{P} uses $O(N^2)$ additional space. \square

Although the (non-geodesic) Fréchet distance is normally computed in $O(N^2 \log N)$ time using parametric search (see [Alt and Godau 1995]), parametric search is often regarded as impractical because it is difficult to implement¹⁰ and involves enormous constant factors

⁹The idea of a global pool is similar to Cole's optimization for parametric search [Cole 1987].

¹⁰Quicksort-based parametric search has been implemented by van Oostrum and Veltkamp [van Oostrum and Veltkamp 2002] using a framework approach.

[Cole 1987]. Algorithm 5.3 provides a practical alternative to parametric search for solving the non-geodesic Fréchet optimization problem in \mathbb{R}^l (see also [Agarwal et al. 1994]).

THEOREM 5.5. *The (non-geodesic) Fréchet distance between two polygonal curves A and B in \mathbb{R}^l can be computed in $O(N^2 \log^2 N)$ expected time, where N is the larger of the complexities of A and B . $O(N^2)$ space is required.*

PROOF. The argument is almost identical to the proof of Theorem 5.4. The main difference is that non-geodesic distances can be computed in $O(1)$ time (instead of $O(\log k)$ time). \square

6. GEODESIC FRÉCHET DISTANCE IN A POLYGONAL DOMAIN WITH OBSTACLES

Consider the situation of a person walking a dog in a park. If the person and dog walk on opposite sides of a group of trees, then the leash will wrap around the trees. More formally, suppose the two polygonal curves A and B lie in a planar polygonal domain \mathcal{D} [Mitchell 1998] of complexity k . The leash is required to change continuously, i.e., it must stay inside \mathcal{D} and may not pass through or jump over an obstacle. It may, however, cross itself. Let δ_C be the geodesic Fréchet distance for this scenario when the leash length is measured geodesically.

Due to the continuity of the leash’s motion, the free space inside a geodesic cell is represented by an hourglass – just as it was for the geodesic Fréchet distance inside a simple polygon. Hence, free space in a cell is x -monotone, y -monotone, and connected (cf. Lemma 3.1), and reachability information can be propagated through a cell in constant time (cf. Lemma 3.4). We compute δ_C for a single initial leash that can be defined by computing a shortest path between the start points of the polygonal curves A and B in $O(k \log k)$ time [Mitchell 1998]. This initial leash defines a *homotopy class* that represents the set of all paths that can be continuously deformed into each other without crossing any obstacles [Hershberger and Snoeyink 1991].¹¹

The main task in computing δ_C is to construct funnels for all cell boundaries. Once these funnels are known, the decision and optimization problems can be solved by the algorithms for the geodesic Fréchet distance inside a simple polygon (cf. Theorems 5.1 and 5.4). We use Hershberger and Snoeyink’s homotopic shortest paths algorithm [Hershberger and Snoeyink 1991] to incrementally construct all funnels needed to compute δ_C . To use the homotopic algorithm, the polygonal domain \mathcal{D} should be triangulated in $O(k \log k)$ time [Mitchell 1998], and all obstacles should be replaced by their vertices.

LEMMA 6.1. *After precomputing a homotopic shortest path for the bottom-left corner of a δ_C -cell \mathcal{C} , all four funnels representing the boundary segments of \mathcal{C} and the homotopic shortest paths for the bottom-left corners of cells adjacent to \mathcal{C} can be computed in $O(k)$ time.*

PROOF. The funnels representing cell boundaries are constructed *incrementally*. The idea is to extend the initial homotopic shortest path into a homotopic “sketch” that describes how the shortest path should wind through the obstacles and then to “snap” this sketch into a homotopic shortest path (see Figures 9a and 9b).

¹¹We recently learned that δ_C has been independently explored in [Chambers et al. 2008]. They repeatedly calculate δ_C $O(N^2)$ times to find the optimal value for δ_C under all possible homotopy classes.

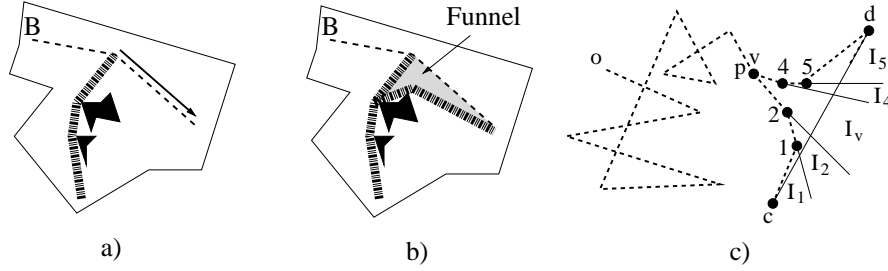


Fig. 9. a) A funnel for a δ_C -cell can be found by first extending an initial homotopic shortest path along one segment to create a path sketch and then b) snapping this sketch into a homotopic shortest path. c) Such a funnel $\mathcal{F}_{o, \overline{cd}}$ has $O(kN)$ complexity, but the distance function $F_{o, \overline{cd}}$ has only $O(k)$ complexity because $d(o, p)$ is a constant.

Homotopic shortest paths have increased complexity over normal shortest paths because they can loop around obstacles. For example, if the person walks in a triangular path around all the obstacles, then the leash follows a homotopic shortest path that can have $O(k)$ complexity in a single cycle around the obstacles. By repeatedly winding around the obstacles $O(N)$ times a path achieves $O(kN)$ complexity.

To avoid spending $O(kN)$ time per cell, we extend a previous homotopic shortest path into a sketch by appending a single line segment to the previous path (see Figure 9a). Adding this single segment can unwind at most one loop over a subset of obstacles, so only the most recent $O(k)$ vertices of the sketch will need to be updated when the sketch is snapped into the true homotopic shortest path. A turning angle is used to identify these $O(k)$ vertices by backtracking on the sketch until the angle is at least 2π different from the final angle.

Putting all this together, a funnel for a boundary segment of a free space cell can be computed in $O(k)$ time by starting with an initial shortest path L_I of $O(kN)$ complexity, constructing a homotopic “sketch” by appending a single segment to L_I , backtracking with a turning angle to find $O(k)$ vertices that are eligible to be changed, and finally “snapping” these $O(k)$ vertices to the true homotopic shortest path using Hershberger and Snoeyink’s algorithm [Hershberger and Snoeyink 1991]. The result is a funnel that describes one cell boundary segment (see Figures 9a and 9b).

Extending and snapping L_I is sufficient to define funnels for all four boundary segments of a free space cell \mathcal{C} that is defined by \overline{ab} and \overline{cd} . For example, extending and snapping L_I along the current $\overline{ab} \in A$ segment defines the funnel for the left cell boundary segment. Similarly, extending and snapping L_I along the $\overline{cd} \in B$ segment defines the funnel for the bottom cell boundary segment. Extending and snapping L_I in this fashion also conveniently defines the initial homotopic shortest paths for cells that are adjacent to \mathcal{C} . \square

THEOREM 6.2. *The δ_C decision problem can be solved in $O(kN^2)$ time and $O(k+N)$ space.*

PROOF. Each cell boundary segment is associated with a funnel $\mathcal{F}_{o, \overline{cd}}$ with $O(kN)$ complexity [Duncan et al. 2006]. However, this high complexity is a result of looping over obstacles, and most of these points do not affect the funnel’s distance function $F_{o, \overline{cd}}$. As illustrated in Figure 9c, $F_{o, \overline{cd}}$ has only $O(k)$ complexity because only vertices $\pi(d, p) \cup \pi(p, c)$ contribute arcs to $F_{o, \overline{cd}}$. To solve the decision problem, compute all cell boundary

funnels in $O(kN^2)$ time using Lemma 6.1 and apply the approach of Theorem 5.1 (since all funnels once again have $O(k)$ complexity). \square

THEOREM 6.3. *The δ_C optimization problem can be solved in $O(kN^2 + N^2 \log kN \log N)$ expected time and $O(kN^2)$ space.¹²*

PROOF. If the funnels are precomputed in $O(kN^2)$ time and space, then Theorem 5.4 applies directly. \square

7. GEODESIC HAUSDORFF DISTANCE

Hausdorff distance is a similarity metric commonly used to compare sets of points or sets of line segments. The *directed* geodesic Hausdorff distance can be formally defined as $\tilde{\delta}_H(A, B) = \sup_{a \in A} \inf_{b \in B} d(a, b)$, where A and B are sets and $d(a, b)$ is the geodesic distance between a and b (see [Alt and Godau 1995; Alt et al. 2003]). The *undirected* geodesic Hausdorff distance is the larger of the two directed distances: $\delta_H(A, B) = \max(\tilde{\delta}_H(A, B), \tilde{\delta}_H(B, A))$.

THEOREM 7.1. *The Euclidean geodesic Hausdorff distance $\delta_H(A, B)$ for point sets A, B inside a simple polygon P can be computed in $O((k + N) \log(k + N))$ time and $O(k + N)$ space, where N is the larger of the complexities of A and B and k is the complexity of P . If A and B are sets of line segments, $\delta_H(A, B)$ can be computed in $O((k + N) \log(k + N))$ time and space.*

PROOF. The directed Hausdorff distance $\tilde{\delta}_H(A, B)$ is calculated by finding for each point $a \in A$ the distance to its nearest neighbor point in B and returning the maximum of these distances. To find nearest neighbors efficiently, precompute the geodesic Voronoi diagrams V_A, V_B for the point sets A and B inside the simple polygon P in $O((k + N) \log(k + N))$ time and $O(k + N)$ space using the algorithm of [Papadopoulou and Lee 1998]. For each point $a \in A$, find its nearest neighbor $a' \in B$ in $O(\log k)$ time and the distance $d(a, a')$ via point location in V_B . Return the maximum of these distances as the value of $\tilde{\delta}_H(A, B)$. $\tilde{\delta}_H(B, A)$ can be computed in a similar manner. For *line segment* sets A and B , the algorithm of Hershberger and Suri [Hershberger and Suri 1999] can be used to build a geodesic Voronoi diagram in $O((k + N) \log(k + N))$ time and space. This Voronoi diagram can be used to identify and resolve $O(N)$ candidate values for the Hausdorff distance using a planesweep algorithm described in [Alt et al. 2003]. \square

8. CONCLUSION

To compute the geodesic Fréchet distance between two polygonal curves inside a simple polygon, we have proven that the free space inside a geodesic cell is x -monotone, y -monotone, and connected. By extending the shortest path algorithms of [Guibas and Hershberger 1989; Hershberger 1991], the boundary of a single free space cell can be computed in logarithmic time, and this leads to an efficient algorithm for the geodesic Fréchet decision problem.

¹²If space is at a premium, the algorithm can also run with $O(k + N^2)$ space and $O(kN^2 \log N + N^2 \log kN \log N)$ expected time by recomputing the funnels each time the decision problem is computed (instead of precomputing the funnels). Note that $O(N^2)$ storage is required for the red-blue intersections algorithm (cf. Theorem 5.4).

We present a randomized algorithm based on red-blue intersections that solves the geodesic Fréchet optimization problem in lieu of the standard parametric search approach. The randomized algorithm is also a practical alternative to parametric search for the non-geodesic Fréchet distance in arbitrary dimensions.

We can compute the geodesic Fréchet distance between two polygonal curves A and B inside a simple bounding polygon P in $O(k + N^2 \log kN \log N)$ expected time, where N is the larger of the complexities of A and B and k is the complexity of P . In the expected case, the randomized optimization algorithm is an order of magnitude faster than a straightforward parametric search that uses Cole's [Cole 1987] optimization to sort $O(kN^2)$ values.

We define the geodesic Fréchet distance in a polygonal domain with obstacles by enforcing a homotopy on the leash. It can be computed in the same manner as the geodesic Fréchet distance inside a simple polygon after computing cell boundary funnels using Hershberger and Snoeyink's homotopic shortest paths algorithm [Hershberger and Snoeyink 1991]. An open question is whether it is possible to compute the funnels in $O(\log k)$ time instead of $O(k)$ time. The geodesic Hausdorff distance for point sets or line segment sets inside a simple polygon can be computed using geodesic Voronoi diagrams.

REFERENCES

- AGARWAL, P. K., SHARIR, M., AND TOLEDO, S. 1994. Applications of parametric searching in geometric optimization. *J. Algorithms* 17, 3, 292–318.
- ALT, H., BRASS, P., GODAU, M., KNAUER, C., AND WENK, C. 2003. Computing the Hausdorff distance of geometric patterns and shapes. In *Discrete and Computational Geometry. Algorithms and Combinatorics*, vol. 25. Springer, Berlin, 65–76. Special Issue: The Goodman-Pollack-Festschrift (B. Aronov, S. Basu, J. Pach, M. Sharir eds.).
- ALT, H. AND GODAU, M. 1995. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry and Applications* 5, 75–91.
- ALT, H., KNAUER, C., AND WENK, C. 2003. Comparison of distance measures for planar curves. *Algorithmica* 38, 1, 45–58.
- BESPAMYATNIKH, S. 2002. An optimal morphing between polylines. *Int. J. Comput. Geometry Appl.* 12, 3, 217–228.
- BESPAMYATNIKH, S. AND SEGAL, M. 2004. Selecting distances in arrangements of hyperplanes spanned by points. *J. Discrete Algorithms* 2, 3 (September), 333–345.
- BUCHIN, K., BUCHIN, M., AND WENK, C. 2006. Computing the Fréchet distance between simple polygons in polynomial time. *SoCG: 22nd Symposium on Computational Geometry*, 80–87.
- CHAMBERS, E. W., DE VERDIÈRE, E. C., ERICKSON, J., LAZARD, S., LAZARUS, F., AND THITE, S. 2008. Walking your dog in the woods in polynomial time. *SoCG: 24th Symposium on Computational Geometry*, 101–109.
- CHAZELLE, B., EDELSBRUNNER, H., GUIBAS, L. J., AND SHARIR, M. 1994. Algorithms for bichromatic line-segment problems and polyhedral terrains. *Algorithmica* 11, 2, 116–132.
- COLE, R. 1987. Slowing down sorting networks to obtain faster sorting algorithms. *J. ACM* 34, 1, 200–208.
- COOK IV, A. F. AND WENK, C. 2007a. Geodesic Fréchet and Hausdorff distance inside a simple polygon. Tech. Rep. CS-TR-2007-004, University of Texas at San Antonio. August.
- COOK IV, A. F. AND WENK, C. 2007b. Geodesic Fréchet distance inside a simple polygon. *17th Fall Workshop on Computational Geometry*.
- COOK IV, A. F. AND WENK, C. 2008. Geodesic Fréchet distance inside a simple polygon. *Proceedings of the 25th International Symposium on Theoretical Aspects of Computer Science (STACS), Bordeaux, France*.
- DUNCAN, C. A., EFRAT, A., KOBOUROV, S. G., AND WENK, C. 2006. Drawing with fat edges. *Int. J. Found. Comput. Sci.* 17, 5, 1143–1164.

- EFRAT, A., GUIBAS, L. J., HAR-PELED, S., MITCHELL, J. S. B., AND MURALI, T. M. 2002. New similarity measures between polylines with applications to morphing and polygon sweeping. *Discrete and Computational Geometry* 28, 4, 535–569.
- GUIBAS, L. J. AND HERSHBERGER, J. 1989. Optimal shortest path queries in a simple polygon. *J. Comput. Syst. Sci.* 39, 2, 126–152.
- GUIBAS, L. J., HERSHBERGER, J., LEVEN, D., SHARIR, M., AND TARJAN, R. E. 1986. Linear time algorithms for visibility and shortest path problems inside simple polygons. *SoCG: 2nd Symposium on Computational Geometry*, 1–13.
- GUIBAS, L. J., HERSHBERGER, J., LEVEN, D., SHARIR, M., AND TARJAN, R. E. 1987. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica* 2, 209–233.
- HERSHBERGER, J. 1991. A new data structure for shortest path queries in a simple polygon. *Inf. Process. Lett.* 38, 5, 231–235.
- HERSHBERGER, J. AND SNOEYINK, J. 1991. Computing minimum length paths of a given homotopy class (extended abstract). In *Workshop on Algorithms and Data Structures*. 331–342.
- HERSHBERGER, J. AND SURI, S. 1999. An optimal algorithm for Euclidean shortest paths in the plane. *SIAM Journal on Computing* 28, 6, 2215–2256.
- KOMLÓS, J., MA, Y., AND SZEMERÉDI, E. 1996. Matching nuts and bolts in $O(n \log n)$ time. *SODA: 7th ACM-SIAM Symposium on Discrete Algorithms*, 232–241.
- MAHESHWARI, A. AND YI, J. 2005. On computing Fréchet distance of two paths on a convex polyhedron. *European Workshop on Computational Geometry (EWCG)*, 41–4.
- MATOUŠEK, J. 1991. Randomized optimal algorithm for slope selection. *Inf. Process. Lett.* 39, 4, 183–187.
- MITCHELL, J. S. B. 1998. Geometric shortest paths and network optimization. *Handbook of Computational Geometry*.
- MITCHELL, J. S. B., MOUNT, D. M., AND PAPADIMITRIOU, C. H. 1987. The discrete geodesic problem. *SIAM Journal on Computing* 16, 4, 647–668.
- PALAZZI, L. AND SNOEYINK, J. 1994. Counting and reporting red/blue segment intersections. *CVGIP: Graph. Models Image Process.* 56, 4, 304–310.
- PAPADOPOULOU, E. AND LEE, D. T. 1998. A new approach for the geodesic Voronoi diagram of points in a simple polygon and other restricted polygonal domains. *Algorithmica* 20, 4, 319–352.
- ROTE, G. 2005. Computing the Fréchet distance between piecewise smooth curves. Tech. Rep. ECG-TR-241108-01. May.
- SARNAK, N. AND TARJAN, R. E. 1986. Planar point location using persistent search trees. *Commun. ACM* 29, 7, 669–679.
- VAN OOSTRUM, R. AND VELTKAMP, R. C. 2002. Parametric search made practical. *SoCG: 18th Symposium on Computational Geometry*, 1–9.
- WENK, C., SALAS, R., AND PFOSE, D. 2006. Addressing the need for map-matching speed: Localizing global curve-matching algorithms. *Proc. 18th International Conference on Scientific and Statistical Database Management (SSDBM)*, 379–388.