Chapter 30

Fréchet distance: How to walk your dog

By Sariel Har-Peled, February 5, 2024[1] Version: 0.1

This chapter is part of a book. See here for more details: http://sarielhp.org/book/.

This is an early draft of a new chapter. Read at your own peril.

This chapter is available from here: http://sarielhp.org/book/chapters/frechet.pdf

> Polonius: My lord, I will use them according to their desert.
>
> Hamlet: God's bodykins, man, much better! Use every man after his desert, and who should scape whipping? Use them after your own honour and dignity. The less they deserve, the more merit is in your bounty.
>
> Hamlet, Shakespeare

Given two curves, we are interested in measuring how similar are they to each other. This can be useful in various domains, from handwriting recognition, clustering, etc. Here, we discuss one such measure – the ***Fréchet distance*** – it is an elegant measure of similarity between curves, similar conceptually to edit distance. In this chapter, we are going define it formally, present exact and approximation algorithms for its computation.

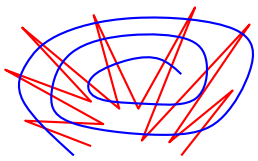
30.1. Introduction and solving the discrete case

The Hausdorff distance, and why it sucks. Assume we are in some continuous metric space (usually $\mathbb{R}^d$ in our case), with a distance metric $\mathrm{d}(\cdot,\cdot)$. For a point p and a set π, let $\mathrm{d}(p,\pi) = \min_{q\in\pi} \mathrm{d}(p,q)$, be the ***distance*** of p to π. For a set σ, the distance of σ to π is $\mathrm{d}(\sigma \to \pi) = \max_{p\in\sigma} \mathrm{d}(p,\pi)$ – this is the distance of the furthest[2] point of σ from a point of π. Note that usually $\mathrm{d}(\sigma \to \pi) \neq \mathrm{d}(\pi \to \sigma)$. To overcome this asymmetry, the ***Hausdorff distance*** between π and σ is

$$\mathrm{d}_H(\pi,\sigma) = \max\Big(\mathrm{d}(\sigma \to \pi),\, \mathrm{d}(\pi \to \sigma)\Big).$$

Unfortunately, for curves, the Hausdorff distance fails to capture the structure of curves, measuring two curves that are conceptually far from each other as close. See Figure 30.1 for examples. The main reason for this shortcoming is that the Hausdorff distance consider each point of a set on its own, and does not take into consideration how points of the same set interact with each other. Thus, we are interested in a distance measure that not only compute distances, but also matches the two given curves to each other, while respecting their structure. Thus emerges the Fréchet distance.

[2] Usually, the above quantities are defined using inf and sup but in our settings we work with compact sets, so it is not necessary.



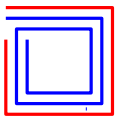


Figure 30.1: the above two examples, show curves that are conceptually far from each other, but are close under the Hausdorff distance.

30.1.1. The Fréchet distance

Informally, the Fréchet distance between two curves is the maximum distance a point on the first curve has to travel as this curve is being continuously deformed into the second curve. The more popular definition is via a person and its dog going on a walk – imagine you have two paths close to each other. The person would walk along the first path, while the dog[3] follows the second path, Assuming the dog is well behaved, how long does the leash between the person and the dog has to be?

This is a motion planning problem. Indeed, once given the curves and the starting points on the curves, we need to compute the optimal way for the person and the dog to move along their respective curves till they reach the endpoints of the curves. This optimal motion can be quite complicated, and a priori it is not even obvious why it can be described efficiently. One simplifying assumption we make throughout is that the two given curves are polygonal curves.

Before providing a formal definition, we first study the discrete version of the problem, which is conceptually easier, but would be quite insightful in studying the continuous variant.

30.1.2. The discrete Fréchet distance (or how to walk your frog, if you are a frog yourself)

So, let π and σ be the two given curves in the plane together with the regular Euclidean distance. Spread n points $p_1, \ldots, p_n$ along π, and m points $q_1, \ldots, q_n$ along σ, where p_1, q_1 are the starting points, and p_n, q_m are the end points. Imagine now that instead of the person and dog moving continuously, we only allow them to use the points marked. That is, the person and dog jump from one allowed point to the other. Initially, the starting configuration is (p_1, q_1). At time t, the jumping person is at point $p_{i(t)}$, and the jumping dog is at point $q_{j(t)}$. At each unit of time, one of them is allowed to jump forward (and only forward!) to the next point. Thus, in time $t+1$ we have that either

(i) $i(t+1) = i(t) + 1$ and $j(t+1) = j(t)$, or
(ii) $i(t+1) = i(t)$ and $j(t+1) = j(t) + 1$.

We usually do not allow for both person and dog to jump simultaneously, but this is not important. The leash length at time t is $\ell(t) = \|p_{i(t)} - q_{j(t)}\|$. In particular, the Fréchet distance provided by the functions $i(\cdot)$ and $j(\cdot)$ is $\max_t \ell(t)$. Here $i(1) = j(1) = 1$. Since exactly one of the entities jumps at every unit of time, and there are $n-1+m-1$ needed jumps, it follows that the motion is done at time $t_{\text{end}} = 1 + n - 1 + m - 1 = n + m - 1$.

The challenge is that the desired Fréchet distance asks for the parameterizations i and j that minimizes $\max_t \ell(t)$. This seems problematic since the number of such parametrizations is quite large. But fortunately, there is an elegant way to encode the parameterizations.

Consider the *configuration space* associated with this problem. The possible parametrizations are points (i, j) in the integer grid $[\![n]\!] \times [\![m]\!]$. The length of the leash of this configuration is the value

[3] Since this is a thought experiment, the reader can pick any other animal instead of a dog. Maybe a whale?

$\|p_i - q_j\|$. Thus, one can think about the configuration space as a two dimensional function. Depicting this function in three dimensions, the image of this function is a discrete terrain, where the value at (i, j) is the ***height*** of this configuration, which is the point $(i, j, \|p_i - q_j\|)$.

From a configuration (i, j) the system is allowed to move to either the configuration $(i + 1, j)$ or $(i, j + 1)$. Namely, a valid parameterization is a monotone grid path from $(1, 1)$ to (n, m). There are $\binom{n-1+m-1}{n-1}$ such paths (you have to make $n + m - 2$ decisions of either moving right or up, with $n - 1$ right moves and $m - 1$ up moves). We are looking for the path in this grid from $(1, 1)$ to (n, m) that minimizes the maximum height of any point along the path.

Giving in to our monstrous pedantic temptations, we next provide a formal definition – the reader that feels comfortable with the above description can skip these formalities directly to Section 30.1.2.1.

Definition 30.1.1. Let $\Pi(n, m)$ be the set of all monotone grid paths from $(1, 1)$ to (n, m). Formally, such a ***monotone grid path*** is pair of functions $(i, j) \in \Pi(n, m)$, such that:

(A) $i : [\![n + m - 1]\!] \to [\![n]\!]$ and $j : [\![n + m - 1]\!] \to [\![m]\!]$.
(B) $i(1) = j(1) = 1$.
(C) For all t, $i(t + 1) \geq i(t)$ and $j(t + 1) \geq j(t)$.
(D) For all t, $\Big(i(t + 1) - i(t)\Big) + \Big(j(t + 1) - j(t)\Big) = 1$.
(E) $i(n + m - 1) = n$ and $j(n + m - 1) = m$.

We refer to such a path as a ***walk***.

Definition 30.1.2. Given two sequences of points $\pi \equiv p_1, \ldots, p_n$ and $\sigma \equiv q_1, \ldots, q_m$, and a monotone path $(i, j) \in \Pi(n, m)$, the ***leash length*** of this parameterization of the two sequences encoded by (i, j) is

$$\ell(i, j) = \max_t \ell(t) = \max_t \|p_{i(t)} - q_{j(t)}\|.$$

The ***discrete Fréchet distance*** between the two sequences, is the distance

$$\mathsf{d}_{\mathcal{F}}(\pi, \sigma) = \min_{(i,j)\in\Pi(n,m)} \ell(i, j) = \min_{(i,j)\in\Pi(n,m)} \max_t \|p_{i(t)} - q_{j(t)}\|.$$

30.1.2.1. Computing the discrete Fréchet distance using dynamic programming

Given two sequences $\pi \equiv p_1, \ldots, p_n$ and $\sigma \equiv q_1, \ldots, q_m$ of points in $\mathbb{R}^d$, computing their (discrete) Fréchet distance is similar in spirit to edit distance. Indeed, let $f(x, y)$ be the Fréchet distance between $p_1, \ldots, p_x$ and $q_1, \ldots, q_y$. Consider the last move in the optimal path. It either was $(x - 1, y) \to (x, y)$ or $(x, y - 1) \to (x, y)$. As such, we have that

$$f(x, y) = \max\Big(\|p_x - q_y\|, \min\Big(f(x - 1, y), f(x, y - 1)\Big)\Big).$$

Using dynamic programming/memoization, the value of $f(n, m)$ and the parameterization/grid-path realizing it can be computed in $O(nm)$ time. We thus get the following result.

Lemma 30.1.3. *Given two sequences $\pi \equiv p_1, \ldots, p_n$ and $\sigma \equiv q_1, \ldots, q_m$ of points in $\mathbb{R}^d$, one can compute their discrete Fréchet distance in $O(nm)$ time.*

Naively the space used by the above algorithm is $O(nm)$ (it can be reduced by being more careful about the dynamic programming, but this not relevant for our purpose here).

30.1.2.2. Computing the weak discrete Fréchet distance

An interesting variant of the Fréchet distance is when we allow the person or the dog to go backward, or jump backward in the discrete case. As before we are looking for the path in the grid of lowest maximum value, except that the path might now go back and forth, and is potentially much longer. Fortunately, this problem can be reduced to the problem of bottleneck edge of the minimum spanning tree.

Given two curves as above with n and m vertices, for an edge e of the integer grid $[\![n]\!] \times [\![m]\!]$, that connects (x, y) to (x', y') (i.e., $|x-x'|+|y-y'| = 1$), assign it the value $\ell(e) = \max(\|p_x - q_y\|, \|p_{x'} - q_{y'}\|)$. Now, we are looking for the path in the grid connecting $(1, 1)$ with (n, m) that minimizes the maximum edge on the path (i.e., the bottleneck edge).

Definition 30.1.4. For a graph G with weights $\omega(\cdot)$ on the edges, a ***bottleneck edge*** in a path π is the edge realizing $\omega_{\max}(\pi) = \max_{e\in\pi} \omega(e)$. A ***bottleneck path*** between vertices s, t in G is a path minimizing $\omega_{\max}(\cdot)$ over all paths connecting s to t in G.

Claim 30.1.5. *Given a graph G with n vertices and m edges with weights on the edges, a bottleneck path between two vertices s, t of G can be computed in $O(n + m)$ time.*

Proof: For the sake of simplicity of exposition assume all the weights on the edges of G are unique. Let $\mathcal{T}$ be the MST of graph.

Let π be the unique path between s and t in $\mathcal{T}$ – we claim this is a bottleneck path. So, let σ be a bottleneck path connecting s and t. If σ is contained in $\mathcal{T}$ then we are done. Otherwise, the path σ must contain an edge e that is not in $\mathcal{T}$. But then $\mathcal{T} + e$ contains a cycle C such that e is the most expensive edge in C (otherwise, $\mathcal{T}$ would not be the MST). We replace e in σ, by the unique path between its endpoints in $\mathcal{T}$. Clearly, we got a new path σ' such that $\omega_{\max}(\sigma') \le \omega_{\max}(\sigma)$. Repeating this replacement processes, over all edges outside $\mathcal{T}$, we end up with a path σ'' that is fully contained in $\mathcal{T}$ connecting s and t such that $\omega_{\max}(\sigma'') \le \omega_{\max}(\sigma)$. The edges of σ'' must cover all the edges of π, and it thus follows that $\omega_{\max}(\pi) \le \omega_{\max}(\sigma'') \le \omega_{\max}(\sigma)$, and since σ is a bottleneck path it follows that these inequalities hold with equality. It follows that π is a bottleneck path.

As for the algorithm, one can compute the MST in randomized linear time, and then compute the unique path in $\mathcal{T}$ in linear time. In this case there is a beautiful and simple linear time deterministic algorithm for computing the path. We leave it as an exercise for the reader (See Exercise 30.1). ■

30.1.3. A formal definition of the Fréchet distance

Some notations, unfortunately. Let π be a ***curve*** in $\mathbb{R}^d$; that is, a continuous mapping from $[0, 1]$ to $\mathbb{R}^d$. Namely, a curve is parameterized by a point in the interval $[0, 1]$. We identify π with its range $\pi([0, 1]) \subseteq \mathbb{R}^d$. We usually deal with polygonal curves which are formed by a sequence of segments. For a curve π, its ***length*** is denoted by $\|\cdot\|$. For two points q and s on a curve π, let $\pi[q, s]$ denote the portion of the curve between the two points.

The Fréchet distance. A ***reparameterization*** is a bijective and continuous function $f : [0, 1] \to [0, 1]$ such that $f(0) = 0$ and $f(1) = 1$. A pair (f, g) of reparameterizations of the two curves π and σ, respectively, is a ***walk***. The ***height*** of the walk is

$$\hbar_{\pi,\sigma}(f, g) = \max_{s\in[0,1]} \|\pi(f(s)) - \sigma(g(s))\| .$$

This can be interpreted as the maximum length of a leash one needs to walk a dog, where the dog walks monotonically along π according to f, while the handler walks monotonically along σ according to g. In this analogy, the Fréchet distance is the shortest possible leash admitting such a walk.

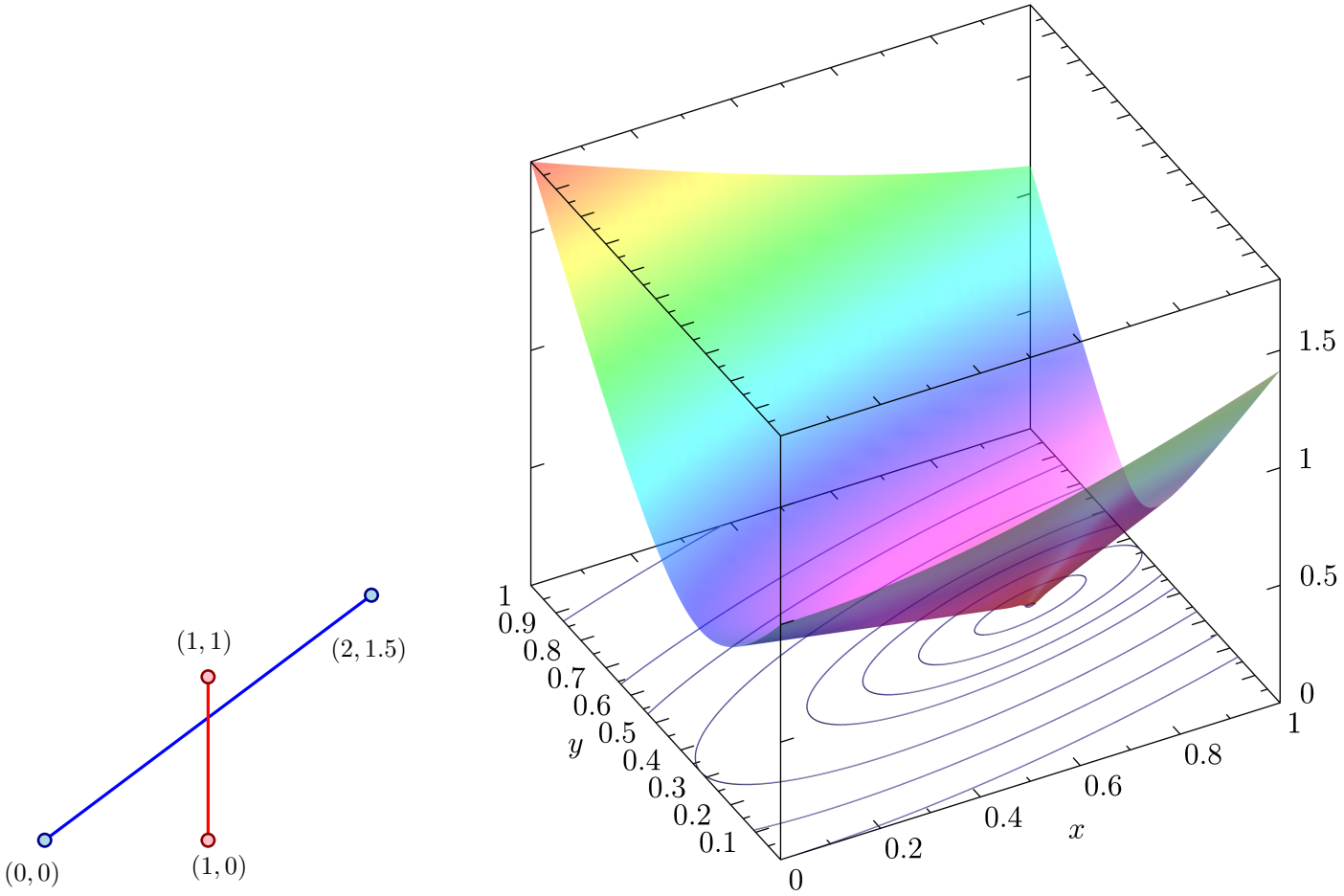


Figure 30.2: The elevation function for two segments.

Definition 30.1.6. Given two curves π and σ in $\mathbb{R}^d$, the ***Fréchet distance*** between them is

$$\mathsf{d}_{\mathcal{F}}(\pi, \sigma) = \inf_{\substack{f:[0,1]\to[0,1] \\ g:[0,1]\to[0,1]}} \hbar_{\pi,\sigma}(f, g),$$

where f and g are reparameterizations of the curves π and σ, respectively.

It is easy to verify that the Fréchet distance complies with the triangle inequality; that is, for any three curves π, σ and τ we have that $\mathsf{d}_{\mathcal{F}}(\pi, \tau) \leq \mathsf{d}_{\mathcal{F}}(\pi, \sigma) + \mathsf{d}_{\mathcal{F}}(\sigma, \tau)$.

30.2. Computing the Fréchet distance

30.2.1. The Fréchet distance between two segments

We start our investigation with the simplest case – the two curves are straight segments. So let $\pi = pp'$ and $\sigma = qq'$, where $p, p', q, q \in \mathbb{R}^d$. Furthermore, assume that these segments are parameterized ***uniformly***; that is, for $t \in [0, 1]$, we have $\pi(t) = (1 - t)p + tp'$. Here, a the person starts at p and the dog at q, and in the end of the walk, the person is at p' and the dog is at q'. A configuration is a

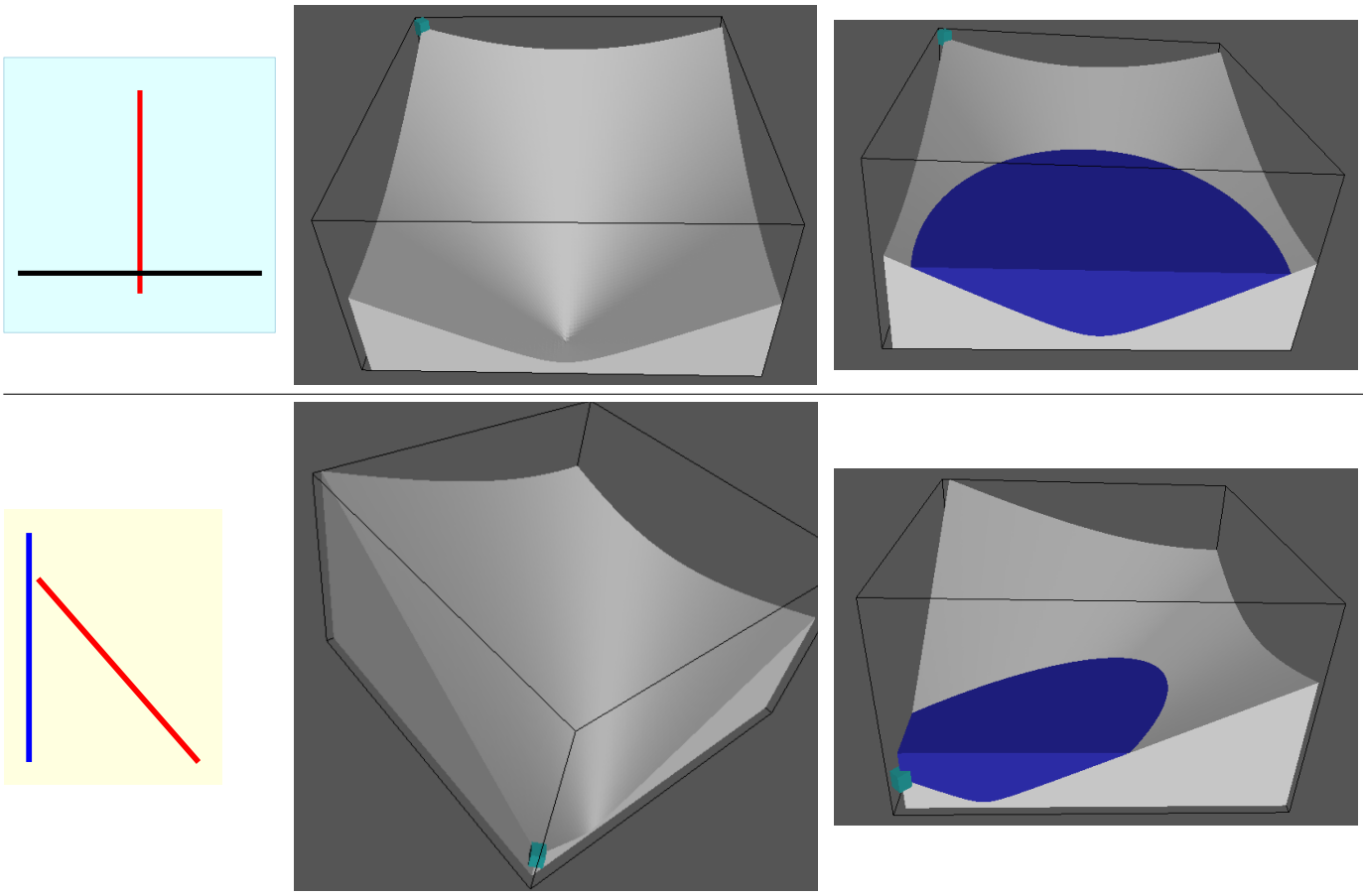


Figure 30.3: Two examples of the elevation function for two segments, together with an example of the level set.

point $(x, y) \in [0, 1]^2$ corresponding to the person being at $\pi(x)$ and the dog at $\sigma(y)$. The *price* of this configuration (x, y) is specified by the ***elevation*** function

$$\operatorname{elev}\Big(x, y\Big) = \operatorname{elev}\Big((x, y)\Big) = \|\pi(x) - \sigma(y)\| \tag{30.1}$$

(i.e., this is the length of leash needed for this configuration). See Figure 30.2 and Figure 30.3 for examples. Under this interpretation, the ***configuration space*** is the unit square $[0, 1]^2$, and the task of computing the Fréchet distance between the curve is equivalent to computing a (monotone) path with minimum maximum elevation that connects $(0, 0)$ to $(1, 1)$.

Lemma 30.2.1. *For a fixed $\delta \geq 0$, the sublevel set of the elevation function*

$$D_{\leq \delta} = \Big\{(x, y) \in [0, 1]^2 \;\Big|\; \operatorname{elev}(x, y) \leq \delta\Big\}$$

is an ellipse clipped to the unit square.

Proof: Consider the mapping $f : (x, y) \to \pi(x) - \sigma(y)$ is an affine function, and assuming general position of the segments, it is also one-to-one. The elevation function is $\operatorname{elev}(x, y) = \|f(x, y)\|$. As such, all the configurations with leash of length $\leq \delta$ are mapped into the ball b centered at the origin with radius δ. Let $\mathcal{F}$ be the two dimensional affine subspace that is the image of f; that is, $\mathcal{F} = f(\mathbb{R}^2)$. Clearly, the desired set is $f^{-1}(\mathcal{F} \cap \mathsf{b}) \cap [0, 1]^2$. The inverse of an affine function is an affine function (i.e., f^{-1}), and the set $\mathcal{F} \cap \mathsf{b}$ is a disk. The affine image of a disk is an ellipse, and as such so is $f^{-1}(\mathcal{F} \cap \mathsf{b})$.■

We need the following helper lemma which follows by standard calculations. The proof is included, but the reader is encouraged to skip it.

Lemma 30.2.2. *Let $f : \mathbb{R}^k \to \mathbb{R}^d$ be an affine function. Then, for $s \in \mathbb{R}^k$, the function $\|f(s)\|$ is convex.*

Proof: Fix any two points $\hat{p}.\hat{q} \in \mathbb{R}^k$, and consider the segment $\hat{p}\hat{q}$. We need to prove that $h(t) = \left\|f((1-t)\hat{p} + t\hat{q})\right\|$ is convex. Let $p = f(\hat{p})$ and $q = f(\hat{q})$. Since f is affine, we have that $h(t) = \left\|f((1-t)\hat{p} + t\hat{q})\right\| = \left\|(1-t)f(\hat{p}) + tf(\hat{q})\right\| = \|(1-t)p + tq\| = \sqrt{\sum_{i=1}^d g_i(t)}$, where $g_i(t) = ((1-t)p_i + tq_i)^2 = \alpha_i t^2 + \beta_i t + \gamma_i$ and $\alpha_i, \beta_i, \gamma_i$ are constants, for $i = 1, \ldots, d$. For any i, the function $g_i(t)$ is nonnegative. If $\alpha_i = 0$ then $g_i(t) = \gamma_i$ and then $p_i = q_i$. Otherwise, $\alpha_i > 0$ and $g_i(t)$ is a parabola. Let $\alpha = \sum_i \alpha_i$, $\beta = \sum_i \beta_i$ and $\gamma = \sum_i \gamma_i$, and consider the function $g(t) = \sum_{i=1}^d g_i(t) = \alpha t^2 + \beta t + \gamma$. If, for all i, $p_i = q_i$, then $g(t) = \gamma$, and the claim trivially holds. Otherwise, $g(t)$ is a non-negative parabola with $\alpha > 0$, and since it has at most a single root, we have that $\beta^2 - 4\alpha\beta \leq 0$.

Now, we have

$$h'(t) = \frac{2\alpha t + \beta}{2\sqrt{\alpha t^2 + \beta t + \gamma}} = \frac{h_1(t)}{h(t)} \qquad \text{for } h_1(t) = \alpha t + \beta/2,$$

$$\text{and} \qquad h''(t) = \frac{h(t)g'(t) - h'(t)h_1(t)}{(h(t))^2} = \frac{\alpha h(t) - (h_1(t))^2/h(t)}{(h(t))^2} = \frac{(h(t))^2 - (h_1(t))^2/\alpha}{(h(t))^3/\alpha}.$$

This implies that

$$\begin{aligned}\operatorname{sign}(h''(t)) &= \operatorname{sign}\Big((h(t))^2 - (h_1(t))^2/\alpha\Big) = \operatorname{sign}\Big(\alpha t^2 + \beta t + \gamma - \alpha t^2 - \beta t - \beta^2/4\alpha\Big) \\ &= \operatorname{sign}\Big(\gamma - \beta^2/4\alpha\Big) = \operatorname{sign}\Big(4\alpha\gamma - \beta^2\Big) \geq 0,\end{aligned}$$

since $\alpha > 0$ and $\beta^2 - 4\alpha\gamma \leq 0$. Which implies that $h(\cdot)$ is convex, and so is $\|f(\cdot)\|$. ■

Since $\pi(x) - \sigma(y)$ is an affine function, we readily get the following.

Corollary 30.2.3. *The elevation function* $\operatorname{elev}(\cdot, \cdot)$*, see Eq. (30.1), is convex.*

Observation 30.2.4. *Given values x_0 and α, computing the maximum interval $[y_1, y_2] \subseteq [0, 1]$ such that $\operatorname{elev}\Big(x_0, y\Big) \leq \alpha$ can be done in constant time. Indeed, consider the ball of radius α centered at the point $\pi(x_0)$, and observe that the desired interval corresponds to portion of σ in this ball. The symmetric claim holds for computing the interval of values of x for which $\operatorname{elev}\Big(x, y_0\Big) \leq \alpha$.*

Lemma 30.2.5. *The Fréchet distance between two given segments π and σ is realized by the uniform parameterization that matches $\pi(t)$ with $\sigma(t)$, for $t \in [0, 1]$.*

Proof: The Fréchet distance between π and σ is at least

$$\alpha = \max\Big(\|\pi(0) - \sigma(0)\|, \|\pi(1) - \sigma(1)\|\Big) = \max\Big(\operatorname{elev}(0,0), \operatorname{elev}(1,1)\Big)$$

since the $(0, 0)$ and $(1, 1)$ are the starting and ending configurations, respectively, of the desired matching. Lemma 30.2.3 implies that any configuration (t, t) has elevation at most

$$\operatorname{elev}(t,t) \leq (1-t)\operatorname{elev}(0,0) + t\operatorname{elev}(1,1) \leq \max(\operatorname{elev}(0,0), \operatorname{elev}(1,1)) = \alpha.$$

We conclude that the $\mathsf{d}_{\mathcal{F}}(\pi, \sigma) = \alpha$. ■

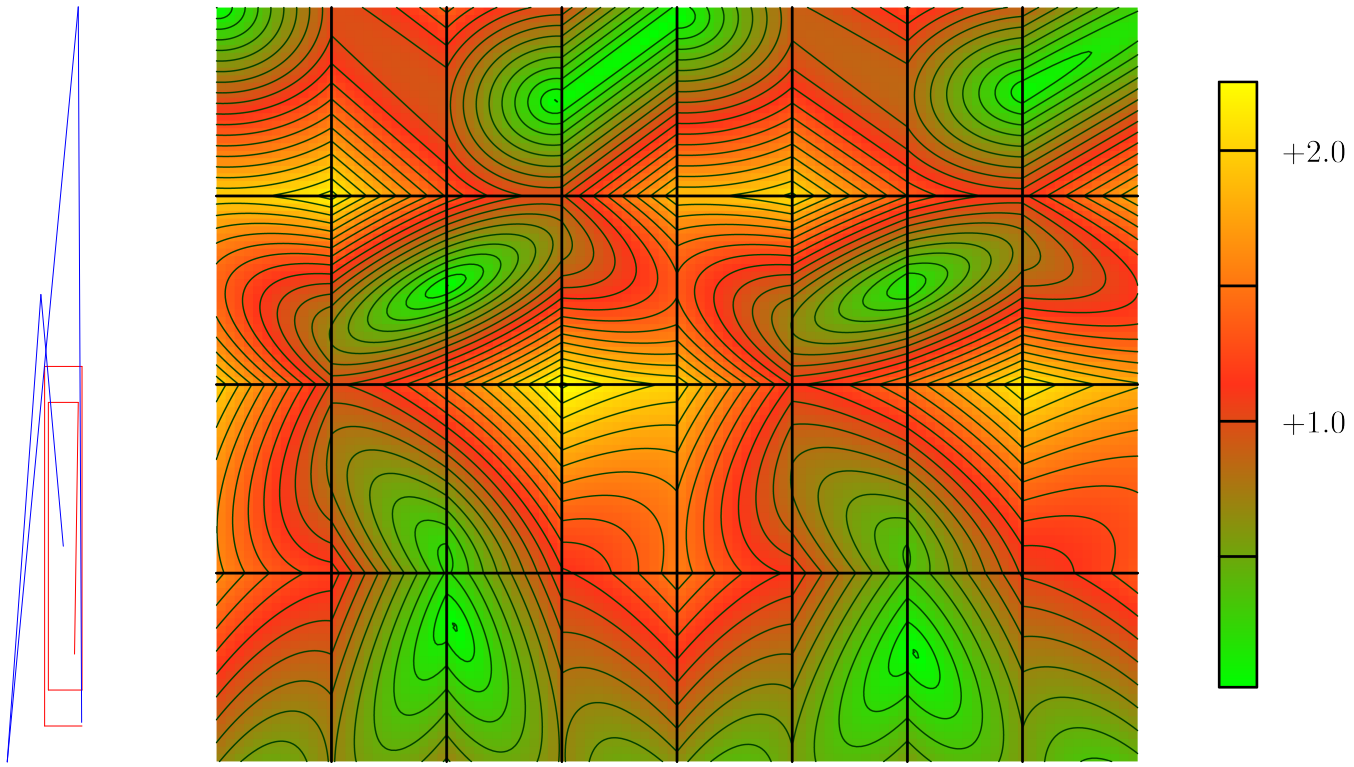


Figure 30.4: Two polygonal curves and their elevation function.

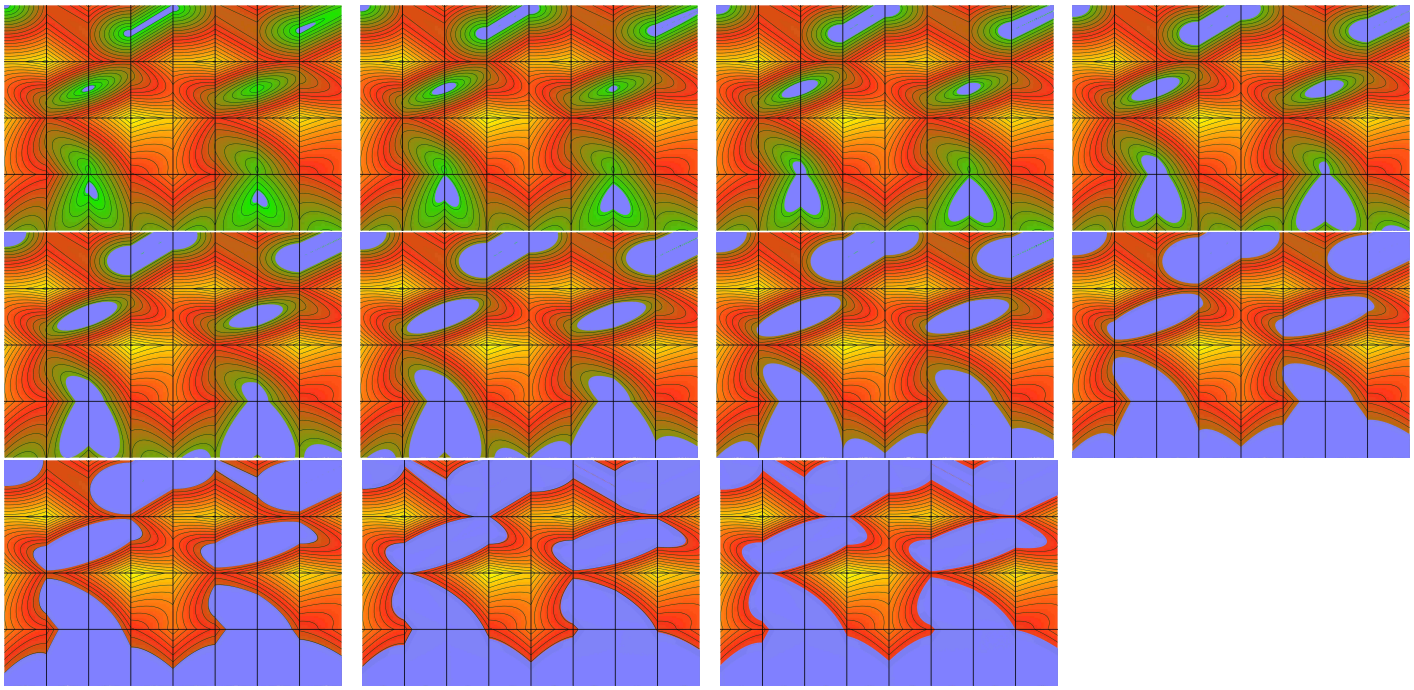


Figure 30.5: The water is rising – we must learn how to swim. As the water level rises, the connectivity of the free space grows, till a Frechet path becomes feasible.

30.2.2. The free space diagram

We are given two polygonal curves π and σ in $\mathbb{R}^d$, and the task is to compute the Fréchet distance between them. Assume that π has vertices $p_0, p_1, \ldots, p_n$ (in this order along the curve), and similarly $\sigma = q_0 q_1 \ldots q_m$. For $i = 1, \ldots, n$, it would be convenient to parameterize the ith segment π, that is $p_{i-1}p_i$ uniformly on the interval $[i-1, i]$. As such, π is parameterized by the interval $[0, n]$. Similarly, σ is parameterized by the interval $[0, m]$.

The ***free space diagram***[4] is the rectangle $D = [0, n] \times [0, m]$, A configuration $(x, y) \in D$, corresponds to the two points $\pi(x)$ and $\sigma(y)$. The elevation function value (i.e., the length of the leash needed for this confirmation) at (x, y), as before, is simply the distance between $\pi(x)$ and $\sigma(y)$. In particular, the domain D is a rectangle divided into unit squares by a $n \times m$ grid, where inside each grid cell, the elevation function is the one defined for a specific segment of π against a specific segment of σ. As such, going back to my childhood, the elevation function looks a bit like a 5×6 tray of eggs (except that it might not be that regular).

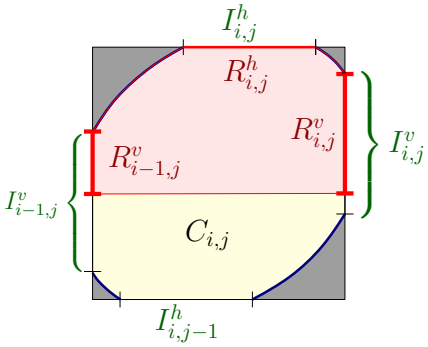
An example of the elevation function of two polygonal paths is depicted in Figure 30.4. It is somewhat convenient to think about the image of the elevation function as an implicitly defined terrain.

A reparameterization of the two polygonal curves corresponds to an (x, y)-monotone curve τ in D from $(0, 0)$ to (n, m). The price of such a path τ is the maximum value of the elevation function on τ. Going back to the terrain analogy, the Fréchet distance is the path on the terrain with its highest point being minimal (i.e., we are trying to use the lowest passes in the terrain to get from the source to the destination).

Let $\delta \geq 0$ be parameter, the ***free space*** of π and σ of radius δ is defined as

$$D_{\leq \delta}(\pi, \sigma) = \left\{ (s, t) \in [0, 1]^2 \;\middle|\; \|\pi(s) - \sigma(t)\| \leq \delta \right\}.$$

By Lemma 30.2.1, this set clipped to a grid cell of D, is a clipped ellipse. Indeed, such a grid cell corresponds to two segments of π and σ, respectively. In particular, let $\square_{i,j} = \square_{i,j}(\pi, \sigma)$ denote the ***free space cell*** that corresponds to the ith edge of π and the jth edge of σ. The cell $\square_{i,j}$ is located in the ith column and jth row of the grid of D.



The free space inside such a cell is depicted on the right. Let $I^h_{i,j}$ denote the horizontal ***free space interval*** at the top boundary of $\square_{i,j}$, and $I^v_{i,j}$ denote the vertical free space interval at the right boundary.

The Fréchet distance between π and σ is at most δ if and only if there is an (x, y)-monotone path in the free space diagram between $(0, 0)$ and $(1, 1)$ that is fully contained in $D_{\leq \delta}(\pi, \sigma)$. Let the ***reachability intervals*** $R^h_{i,j} \subseteq I^h_{i,j}$ and $R^v_{i,j} \subseteq I^v_{i,j}$ consist of the points (x, y) on the boundary that are reachable by a monotone path from $(0, 0)$ to (x, y).

30.2.2.1. Characterizing the optimal solution

Lemma 30.2.6. *The (shortest) optimal walk is a polygonal path, that crosses each unit cell of the grid of D by a straight segment.*

[4] It is just a configuration space, but this is the term used in the literature and we follow suit. We had resisted the temptation to introduce our own term here. Candidates included: *doggy* space, and *Via Dolorosa* space.

Proof: By the monotonicity requirement, the optimal walk w_{opt} can enter a cell of the grid only once (and thus leave it only once). So consider the entrance point $t = (x, y)$ and the exit point $t' = (x', y')$ of the walk for a cell $\square = [i, i+1] \times [j, j+1]$ of D. The elevation function in $\square$ is the elevation function between two segments of the two paths and by Lemma 30.2.3 it is convex. In particular, the elevation for any point on the segment tt' is bounded by $\max(\text{elev}(t), \text{elev}(t'))$. As such, we can replace the subwalk $w_{\text{opt}}[t, t']$ by the segment tt', which potentially shorten it. Doing this process for all the cells of the domain implies the claim. ■

30.2.3. A decider for the Fréchet distance via the reachable free space

For a fixed length of the leash δ, the ***reachable free space***, denoted by $F_{\leq\delta} = F_{\leq\delta}(\pi, \sigma)$, is the set of all the points of $D_{\leq\delta}(\pi, \sigma)$ that are reachable from $(0, 0)$ by an (x, y)-*monotone* path. Here we present a decision procedure that decides if $(n, m) \in F_{\leq(}\delta)$ – if it is then $\mathsf{d}_{\mathcal{F}}(\pi, \sigma) \leq \delta$ (otherwise, $\mathsf{d}_{\mathcal{F}}(\pi, \sigma) > \delta$).

The set $F_{\leq\delta}$ has finite descriptive complexity inside each grid cell, and we need to describe it only for the grid cells that have non-empty intersection with $F_{\leq\delta}$. Clearly, generating only those grid cells is sufficient to decide if there is a monotone path between $(0, 0)$ and (n, m), which is equivalent to deciding if the Fréchet distance between π and σ is smaller or equal to δ. In particular, to fully describe $F_{\leq\delta}$, we compute the reachability intervals $R^h_{i,j} \subseteq I^h_{i,j}$ and $R^v_{i,j} \subseteq I^v_{i,j}$ for each cell $\square_{i,j}$, which describe the intersection of $F_{\leq\delta}$ with the top and right boundary of $\square_{i,j}$. These intervals contain all the needed information, since $F_{\leq\delta} \cap \square_{i,j}$ is convex.

Definition 30.2.7. The ***complexity*** of the reachable free space, for distance δ, denoted by $N_{\leq\delta}(\pi, \sigma)$, is the total number of grid cells which have non-empty intersection with $F_{\leq\delta}$.

Observe, that naively, we have that $N_{\leq\delta}(\pi, \sigma) = O(nm)$, where n and m are the number of vertices of π and σ, respectively.

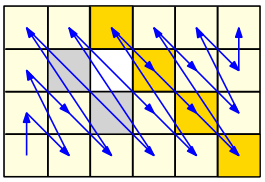
One can compute this set of cells and extract an existing monotone path in $O(N_{\leq\delta}(\pi, \sigma))$ time, by performing a BFS of the grid cells that visits only the reachable cells.

Lemma 30.2.8. *Given two polygonal curves π and σ in $\mathbb{R}^d$, and a parameter $\delta \geq 0$, one can compute a representation of $F_{\leq\delta}(\pi, \sigma)$ in $O(N_{\leq\delta}(\pi, \sigma))$ time. Furthermore, one can decide if $\mathsf{d}_{\mathcal{F}}(\pi, \sigma) \leq \delta$. If so, one can compute, in $O(N_{\leq\delta}(\pi, \sigma))$ time, a walk that realize it.*

Proof: We create a directed graph G that has a node $v(i, j)$ for every reachable free space cell $\square_{i,j}$. With each node $v(i, j)$ we store the free space intervals $I^h_{i,j}$ and $I^v_{i,j}$ as well as the reachability intervals $R^h_{i,j} \subseteq I^h_{i,j}$ and $R^v_{i,j} \subseteq I^v_{i,j}$.

Each node $v(i, j)$ can have an outgoing edge to its right and top neighbor; an edge between these vertices exists if and only if the corresponding reachability interval between them is nonempty. In particular, a monotone path from $(0, 0)$ to a point $(x, y) \in \square_{i,j}$ in $F_{\leq\delta}$ corresponds to a monotone path in the graph G from $v(1, 1)$ to $v(i, j)$. Furthermore, any such monotone path has exactly $k = i + j - 2$ edges on it.

We compute the graph G on the fly by performing a BFS on it, starting from $v(1, 1)$, and keeping the invariant that when the BFS visits a node $v(i, j)$ it enqueues the vertices $v(i, j + 1)$ and $v(i + 1, j)$, in this order, to the BFS queue (if they are connected to $v(i, j)$, naturally).



This implies that at any point in time, and for any k, the BFS queue contains the nodes on the kth diagonal (i.e., all nodes $v(i,j)$ such that $i+j=k-1$) of the diagram sorted from left to right. However, the same node might appear twice (consecutively) in this queue.

In every iteration, the BFS dequeues the one or two copies of the same node $v(i,j)$ and merges the two copies of the same vertex into one if necessary. Now, the one or two vertices (i.e., $v(i-1,j)$ and $v(i,j-1)$) that have incoming edges to $v(i,j)$ are known, as are their reachability intervals. Therefore one can compute the reachability intervals for $v(i,j)$ in constant time. Now, $v(i,j+1)$ is enqueued if and only if the top side of the cell $\square_{i,j}$ is reachable by a monotone path (i.e., $R^h_{i,j} \neq \emptyset$), and $v(i+1,j)$ is enqueued if and only if the right side of the cell $\square_{i,j}$ is reachable by a monotone path (i.e., $R^v_{i,j} \neq \emptyset$). Since $F_{\leq\delta}(\pi,\sigma) \cap \square_{i,j}$ is convex and of constant complexity, this can be done in constant time.

Clearly, the BFS takes time linear in the size of G and it computes the reachability information for all reachable free space cells of $F_{\leq\delta}(\pi,\sigma)$. Now, one can check if (n,m) is reachable by inspecting the reachability intervals for $\square_{n-1,m-1}$, and checking if the top right corner of this cell is monotonically reachable from the origin, where n and m are the number of vertices of π and σ, respectively. The monotone path realizing this can be extracted in linear time, by introducing backward edges in the graph and tracing a path back to the origin. ■

30.2.4. Parametric search and the critical events

Once we have the above setup, the optimization problem can be reduced to the decision problem solved above – given a leash length α, is there a valid reparametrization that uses a leash of length at most α. This is the decision procedure described in Lemma 30.2.8.

Given such a ***decision procedure*** that can decide if the optimal solution is equal, larger or smaller than the given parameter, what remains is to perform a search for the optimal value. Naively, one would start with an interval $[0,t]$ that must contain the optimal solution, and then use binary search to repeatedly find a smaller interval containing the optimal solution. This approach works quite well in practice, but it has the drawback that if we use exact arithmetic, it might never find the optimal solution, continuously searching in a rapidly shrinking interval.

Theoretically, this is not satisfying, and our purpose is to find an algorithm that is guaranteed to stop in finite time. To this end, the basic idea is to compute (implicitly) all the values where the decision procedure might change its mind (i.e., the value it returns). If they number of such critical events is small, then we can sort them, and perform a binary search on the critical values to find the minimum value that is feasible.

Usually, to get an efficient algorithm, one has to do this search implicitly on the critical values without computing them explicitly. There is a rather powerful technique, called ***parametric search*** that enables one to solve such problems efficiently. See the bibliographical notes for details.

So, what are the critical events for the Fréchet distance? It would be convenient to think about the parameter specifying the leash length as being the sublevel set in the terrain defined by the elevation function (i.e., the free space). In particular, one can think about the situation as slowly filling up the terrain with water – as the sea rises, critical events happen (two disjoint connected components merge, etc).

We emphasize that we only care about critical events that changes the structure of the reachable space. There would events that changes the free space that we are going to ignore.

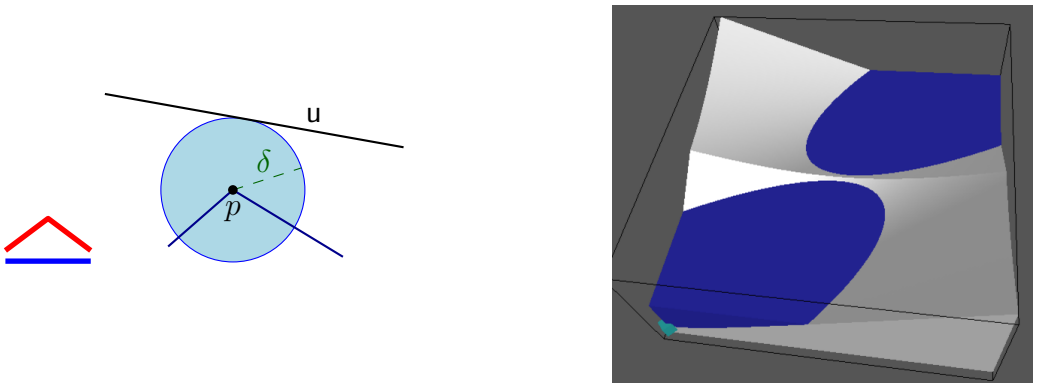


Figure 30.6: Vertex-edge event: This is the distance of a vertex of one curve and an edge of the other curve. Here, as water level rises, two adjacent connected components get connected through a minimum in the middle of an edge of the terrain.

30.2.4.1. Vertex-edge event

Consider a segment u of π and a vertex q of σ, a ***vertex-edge event*** corresponds to the minimum value δ such that u is tangent to $\text{ball}(q, \delta)$. In the free space diagram, this corresponds to the event that a free space interval that consists of only one point was just created. The line supporting this boundary edge corresponds to the vertex, and the other dimension corresponds to the edge. Naturally, the event could happen at a vertex of u. This event is depicted in Figure 30.6.

30.2.4.2. Monotonicity event

The second type of event, a ***monotonicity event***, corresponds to a value δ for which a monotone subpath inside $D_{\leq\delta}$ becomes feasible, see Figure 30.8. Geometrically, this corresponds to two vertices q and s on one curve and a directed segment u on the other curve such that: (1) u passes through the intersection $\mathbb{S}(q, \delta) \cap \mathbb{S}(s, \delta)$, and (2) u intersects $\text{ball}(s, \delta)$ first and $\text{ball}(q, \delta)$ second, where q comes before s in the order along the curve π.

30.2.4.3. Vertex-vertex event

Other values of δ that would be relevant to our algorithm are the distances between any pair of points of $V(\pi) \cup V(\sigma)$. Technically, apart from the two single events that the endpoints of the curves are being matched to each other, these ***vertex-vertex*** events are vertex-edge events when they are relevant to the connectivity of the reachable space.

30.2.4.4. Edge-edge event

This event happens when δ is equal to the distance between two segments, and the two points realizing this distance are in the middle of the two respective edges (which can happen only in three and higher dimensions or if $\delta = 0$ in two dimensions). This event means that the free space had gained a new connected component, but it does not effect the reachable space. As such, this event is irrelevant for describing the algorithm, but we mention it for the sake of completeness.

Observation 30.2.9. *One can compute all relevant vertex-edge events with radius $\leq \delta$ in $O(N_{\leq\delta}(\pi, \sigma))$ time as follows. We compute the graph representation of $F_{\leq\delta}(\pi, \sigma)$ using Lemma 30.2.8. Next, for each reachable cell consider the vertex-edge events at its top and right boundaries and compute their event*

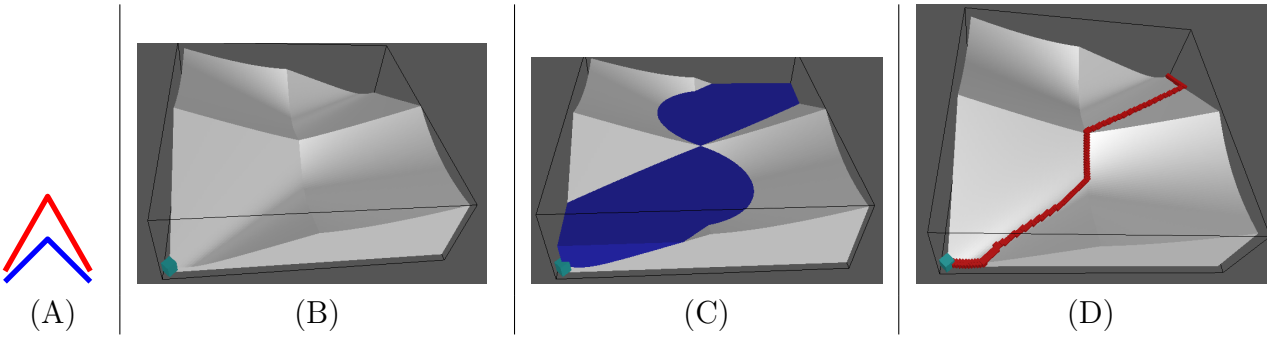


Figure 30.7: (A) Vertex-vertex event – the distance between two vertices. (B) Corresponds to a saddle point in the terrain. (C) Once the water level is high enough two connected components of the level set merge. (D) A walk realizing this distance.

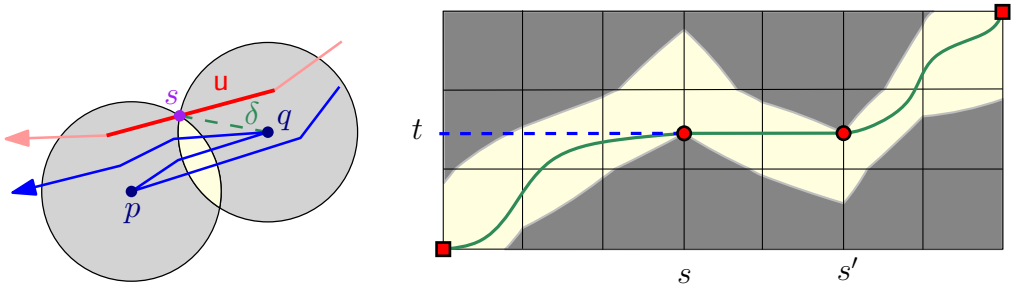


Figure 30.8: Two curves π and σ and their free space diagram $D_{\leq\delta}(\pi,\sigma)$, where $q=\pi(s), s=\pi(s')$ and $c=\sigma(t)$. Here, δ is the minimal free space parameter, such that a monotone path exists, i.e., in this example $\mathsf{d}_{\mathcal{F}}(\pi,\sigma)$ coincides with a monotonicity event.

radii. Recall that a cell boundary corresponds to an edge from the one curve and a vertex from the other curve. Clearly, any cell boundary can be used by the reparameterization of width $\leq\delta$, if and only if the corresponding event radius is smaller or equal δ.

30.2.5. Computing the optimal Fréchet walk/distance

here, we present a simple algorithm for computing exactly the strong Fréchet distance between two polygonal curves. This algorithm has running time $O(n^2\log n)$, and uses randomization and the decision procedure describe above. We need the following easy definition.

Definition 30.2.10. Given a set of numbers $U\subseteq\mathbb{R}$, an ***atomic interval*** of U is a (possibly infinite) maximal interval on the real line that does not contain any point of U in its interior. Let $\mathcal{D}(P)$ be the set of all pairwise distances of points in P.

In the following, let π and σ be the two polygonal curves under consideration, and we assume that the total number of their vertices is n. We outline a randomized algorithm to compute the value of the Fréchet distance between π and σ, that start and end at their respective start and end vertices.

The algorithm needs to search over the critical values when the decision procedure changes its behavior. These critical values were described in Section 30.2.4.

In the following, let δ^* denote the actual minimum value of the Fréchet distance. Given a parameter δ, let decider(δ) be the decision procedure described in Lemma 30.2.8. Let extract(a,b) be a procedure that returns all critical values determined by π and σ whose radius is in the interval $[a,b]$. Suppose, for the time being, that the following subroutines have the following running times:

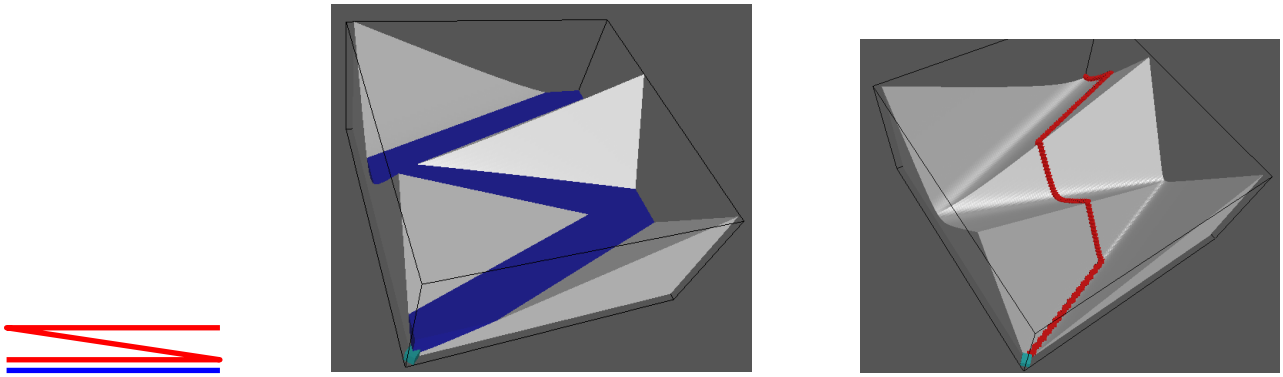


Figure 30.9: A monotonicity event.

```
compFr(π, σ, s1, s2, t1, t2):
    R: random sample of μ = 4n^2 critical values
    Sort R
    Perform a binary search over R using decider
    J = [a, b] ← Atomic interval of R containing δ*
    S ← extract(a, b)
        // S:  all critical values in [a, b]
    Sort S
    x ← Smallest value in S for which decider accepts
        // Computed using a binary search
    Return x
```

Figure 30.10: The algorithm for computing the Fréchet distance between two DAG complexes.

(A) decider(δ) runs in $O(n^2)$ time (Lemma 30.2.8).
(B) extract(a, b) runs in $O(n^2 \log n + k \log n)$ time, where k is the number of critical values with radius in the interval $[a, b]$.
(C) One can uniformly sample a critical value from the set of all critical values in $O(1)$ time per sample.

The new algorithm is depicted in Figure 30.10.

30.2.5.1. Computing the Critical Values in an Interval

To complete the description of the algorithm, we need to describe how to implement extract(a, b). For the interval $\mathcal{I} = [a, b]$, we need to compute all the critical values with radius in $\mathcal{I}$. We can explicitly compute all the radii of vertex-vertex and vertex-edge events in this interval and sort them in $O(n^2 \log n)$ time, where n is the number of edges (since there are $O(n^2)$ such events in total and each radius can be computed in $O(1)$ time). Indeed, for a vertex-vertex event, its radius is the distance between the two vertices that define it. Similarly, the radius of a vertex-edge event is the distance between a vertex and an edge. Both types of radii can be computed in constant time, given the two elements that define them.

In order to compute the radii of monotonicity events in $\mathcal{I}$, we apply a variant of the standard line sweeping algorithm (i.e KDS). Specifically, for π and σ, consider finding all monotonicity events between an edge e of π, and pairs of vertices from $V = V(\sigma)$. To this end, place a sphere of radius δ at each point of V with radius $\delta = a$. We now increase the radius δ until it reaches b. The algorithm maintains

an ordered list L of the intersections of the spheres with the edge e. The events in this growing process are:

(A) The first time a sphere intersects e (this will create two intersections, if the intersection happens internally on e, since after this point the sphere will intersect e in two places).
(B) When the intersection point of a sphere with e grows past an endpoint of e.
(C) When two different spheres intersect at the same point on e. At this point, the algorithm exchanges the order of these two intersections along e. The value of δ when such an event happens is the radius of a monotonicity event.

At any point in time, the algorithm maintains a heap of future events. Whenever a new intersection point is introduced, or two intersections change their order along e, the algorithm computes the next time of an event involving these intersections with the intersections next to them along e.

It is clear that between such events the ordering of the intersections of the spheres with e does not change. Similarly, for a monotonicity event to happen on e, there must be a point in time in which the corresponding spheres are neighbors along e. Hence, this algorithm will correctly find all the monotonicity events.

It takes $O((n+k)\log n)$ time to compute all the relevant monotonicity events involving e and V, where k is the number of such events. We must do this for all edges of π and hence it takes $O((n^2+k')\log n)$ time to compute all the monotonicity events between edges of π and vertices of σ, where $k' = \sum_i k_i$ and k_i is the number of monotonicity events in the interval $[a,b]$ involving the ith edge of π. Therefore it takes $O((n^2+k'')\log n)$ time to compute all the relevant monotonicity events between π and σ, where k'' is the number of such events (i.e. both those involving edges of π and those involving edges of σ).

30.2.5.2. Sampling Critical Values

We can uniformly sample critical values in $O(1)$ time, as follows. A vertex-edge event is determined by sampling a vertex and an edge, a vertex-vertex event is determined by sampling a pair of vertices, and a monotonicity event is determined by sampling a pair of vertices and an edge. Since we can easily uniformly sample vertices and edges in $O(1)$ time, we can therefore do so for critical events. In general, the decision of which type of critical event to sample would have to be weighted by the respective number of such events.

30.2.5.3. Analysis

Let R be the random sample of critical values, of size $O(n^2)$, and let $[a,b]$ be the interval computed by compFr that contains δ^*. The call to extract(a,b) takes $O(n^2\log n + k\log n)$ time, where k is the number of monotonicity events. The following lemma shows that $k = O(n^2)$.

Lemma 30.2.11. *Let $\mathcal{I} = [a,b]$ be the interval computed by compFr, and let c be some positive constant. Then,*

$$\mathbb{P}[\textit{number of critical events in } [a,b] > 2cn\ln n] \leq \frac{1}{n^c}.$$

Proof: There are $2\binom{n}{2}n \leq n^3$ possible monotonicity events, $2n^2$ possible vertex-edge events, and n^2 possible vertex-vertex events. As such, the total number of critical events is bounded by $Z = n^3 + 2n^2 + n^3 \leq 2n^3$.

Consider the position of δ^* on the real line. Let C be the set of the radii of all these critical events, and let U^- (resp. U^+) be the set of $M = cn\ln n$ values of C that are closest to δ^* that are smaller (resp. larger) than it, and let $U = U^- \cup U^+$.

If the number of values in C smaller than δ^* is at most M, then there could be at most M critical values smaller than δ^* in $[a, b]$. The same holds if the C contains less than M values larger than δ^*. As such, in the following, assume that both quantities are larger than M.

The probability that the random sample R of size $\mu = 4n^2$ picked by the algorithm, does not contain a point of U^-, is at most

$$\left(1 - \frac{|U^-|}{|C|}\right)^{\mu} \leq \left(1 - \frac{c \ln n}{2n^2}\right)^{4n^2} \leq \exp(-2c \ln n) \leq \frac{1}{2n^c}.$$

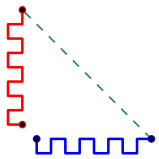
This also bounds the probability that R does not contain a value of U^+. As such, with high probability, $[a, b]$ contains only events in the set U. Namely, $[a, b]$ contains the radii of at most $|U^-| + |U^+| \leq 2M$ monotonicity events, with probability $\geq 1 - 1/n^c$. ■

Combining the above, we thus the following.

Theorem 30.2.12. *For two polygonal curves, π and σ, of total complexity n, with start and end vertices $s_1, t_1 \in \pi, s_2, t_2 \in \sigma$, the algorithm compFr$(\pi, \sigma, s_1, t_1, s_2, t_2)$ computes, in $O(n^2 \log n)$ time, the (strong) Fréchet distance between π and σ. The running time bound holds with high probability.*

30.3. Curve simplification, relative complexity, and c-packed curve

There are reasons to believe that the Fréchet distance can not be well approximated in subquadratic time – see bibliographical notes. In particular, even for two curves defined on the line, we do not know an exact subquadratic algorithm (or even a good approximation algorithm). Of course, such curves that go back and fourth over the same region are not that natural.



The example on the right shows that the situation is even worse – even for two curves that are relatively simple and well behaved, the free space diagram and relevant portion of it to the Fréchet distance between the two curves can have quadratic complexity (specifically, for the Fréchet distance the free space complexity (see Definition 30.2.7) is quadratic. On the other hand, small wiggles of the curves should not matter that much for approximating the Fréchet distance. This naturally raises the idea of simplification – how to simplify the two input curves such that we can compute the Fréchet distance on the simplified curve, and still get a good approximation for the Fréchet distance.

It is thus natural to ask the following question: Are there natural subclass of polygonal curves for which the Fréchet distance can be approximated quickly? By the above, such a class has to be simplification friendly – enabling use to dramatically reduce the complexity of the free space diagram by simplifying the curves, while preserving approximately the Fréchet distance.

To this end, we first introduce an algorithm for simplifying polygon curves. We then define a measure to quantify the benefit of such a simplification as far as the Fréchet distance is concerned. We then introduce the family of c-packed curves which has linear free space complexity in any resolution.

30.3.1. Curve Simplification

The key idea for the new algorithm for approximating the Fréchet distance, is to aggressively simplify the curve as the algorithm runs. Here, we suggest a straightforward greedy algorithm for curve simplification, which is sufficient for our purposes.

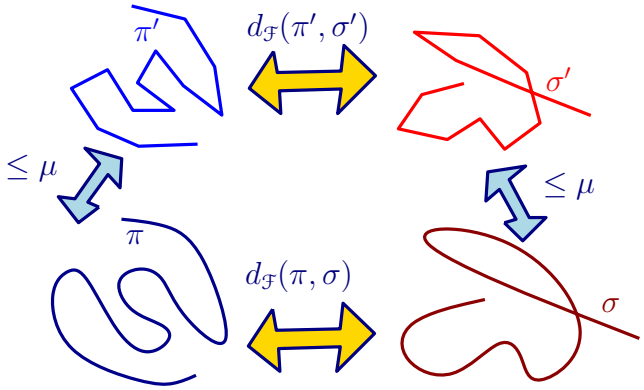


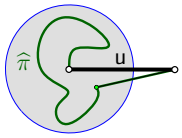
Figure 30.11: The idea of the fuzzy decision procedure using simplification.

Algorithm 30.3.1. Given a polygonal curve $\pi = q_1 q_2 q_3 \ldots q_k$ and a parameter $\mu > 0$, consider the following simplification algorithm: First mark the initial vertex q_1 and set it as the current vertex. Now scan the polygonal curve from the current vertex until it reaches the first vertex q_i that is in distance at least μ from the current vertex. Mark q_i and set it as the current vertex. Repeat this until reaching the final vertex of the curve, and also mark this final vertex. Consider the curve that connects only the marked vertices, in their order along π. We refer to the resulting curve $\pi' = \text{simpl}(\pi, \mu)$ as the ***μ-simplification*** of π. Note, that this simplification can be computed in linear time.

Remark. The simplified curve has the useful property that all its segments are of length at least μ, except for the last edge that might be shorter. For the sake of simplicity of exposition, we assume that the last segment in the simplified curve also has length at least μ.

Lemma 30.3.2. *For any polygonal curve π in $\mathbb{R}^d$, and $\mu \geq 0$, it holds $\mathsf{d}_{\mathcal{F}}(\pi, \text{simpl}(\pi, \mu)) \leq \mu$.*

Proof: Consider a segment u of $\text{simpl}(\pi, \mu)$ and the portion $\widehat{\pi}$ of π that corresponds to it. Clearly, all the vertices of $\widehat{\pi}$ are contained inside a ball of radius μ centered at the first endpoint of u visited by π, except the last vertex of $\widehat{\pi}$. As such, one can parameterize u and $\widehat{\pi}$, such that initially the point stays on the vertex of u while visiting all vertices of $\widehat{\pi}$ (except the last one), and then simultaneously move in sync on u and the last segment of $\widehat{\pi}$, in such a way that the distance is always at most μ. ■



30.3.2. The relative complexity of two curves

The idea underlying the approximate decision procedure is depicted in Figure 30.11. We simplify the two input curves to a resolution that is (roughly) an ε-fraction of the radius we care about (i.e., δ), and we then use the exact decision procedure on these two simplified curves. Since the Fréchet distance complies with the triangle inequality and by Lemma 30.3.2, we can infer the original distance from this information. In order for this approach to work, the complexity of the reachable free space for the two simplified curves has to be small. This notion of complexity is captured by the following definition.

Definition 30.3.3. For two curves π and σ, let

$$\mathsf{N}(\varepsilon, \pi, \sigma) = \max_{\delta \geq 0} N_{\leq \delta}\Big(\text{simpl}(\pi, \varepsilon\delta), \text{simpl}(\sigma, \varepsilon\delta)\Big)$$

be the maximum complexity of the reachable free space for the simplified curves, see Definition 30.2.7. We refer to $\mathsf{N}(\varepsilon, \pi, \sigma)$ as the ***ε-relative complexity*** of π and σ.

Figure 30.12: A few examples of c-packed curves.

Assumption 30.3.4. *We assume that for any $0 < \varepsilon < 1$ the following properties hold for $\mathsf{N}(\cdot,\cdot,\cdot)$:*

(P1) For any constant $c' \geq 1$, it holds $\mathsf{N}(\varepsilon/c', \pi, \sigma) = O(\mathsf{N}(\varepsilon, \pi, \sigma))$.
(P2) $\mathsf{N}(\varepsilon, \pi, \sigma) \leq \mathsf{N}(\varepsilon/2, \pi, \sigma)/2$.

30.3.3. c-packed curves

Intuitively, for a c-packed curve the constant c measures how "*unrealistic*" the input is. formally, a curve π is *c-packed* if the total length of π inside any ball is bounded by c times the radius of the ball. The family of c-packed curves is quite natural. For example, a c-packed curve might self cross and revisit the same location several times, and the class of c-packed curves is closed under concatenation. Intuitively, c-packed curves behave reasonably in any resolution.

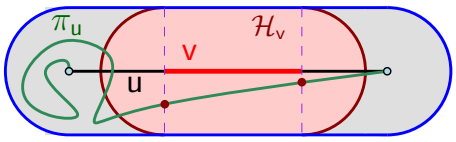
See Figure 30.12 for a few examples of c-packed curves.

30.3.3.1. Formal definition and basic properties

Definition 30.3.5. A curve π in $\mathbb{R}^d$ is ***c-packed*** if for any point q in $\mathbb{R}^d$ and any radius $r > 0$, the total length of π inside the ball $\mathrm{ball}(q, r)$ is at most cr.

Lemma 30.3.6. *Let π be a curve in $\mathbb{R}^d$, $\mu > 0$ be a parameter, and let $\pi' = \mathrm{simpl}(\pi, \mu)$ be the simplified curve. Then $\|\pi \cap \mathrm{ball}(q, r+\mu)\| \geq \|\pi' \cap \mathrm{ball}(q, r)\|$ for any ball $\mathrm{ball}(q, r)$.*

Proof: Let u be a segment of π' that intersects $\mathrm{ball}(q, r)$ and let $v = u \cap \mathrm{ball}(q, r)$ be this intersection. Let π_u be the portion of π that got simplified into u. Observe that π_u is a polygonal curve that lies inside a hippodrome of radius μ around u; that is, $\pi_u \subseteq \mathcal{H}_u = u \oplus \mathrm{ball}(0, \mu)$, where $\oplus$ denotes the Minkowski sum of the two sets, see the figure on the right.



In particular, erect two hyperplanes passing through the endpoints of v that are orthogonal to v, and observe that π_u must intersect both hyperplanes. Hence, we conclude that the portions of π_u in the hippodrome $\mathcal{H}_v = v \oplus \mathrm{ball}(0, \mu)$ are of length at least $\|v\|$. Clearly, $v \subseteq \mathrm{ball}(q, r)$ implies that $\mathcal{H}_v \subseteq \mathrm{ball}(q, r+\mu)$, which in turn implies that $\pi_u \cap \mathcal{H}_v \subseteq \mathrm{ball}(q, r+\mu)$ and thus $\|\pi_u \cap \mathrm{ball}(q, r+\mu)\| \geq \|v\|$.

Summing over all segments v in $\pi' \cap \mathrm{ball}(q, r)$ implies the claim. ■

Lemma 30.3.7. *Let π be a c-packed curve in $\mathbb{R}^d$, $\mu > 0$ be a parameter, and let $\pi' = \mathrm{simpl}(\pi, \mu)$ be the simplified curve. Then, π' is a $6c$-packed curve.*

Proof: Assume, for the sake of contradiction, that $\|\pi' \cap \text{ball}(q,r)\| > 6cr$ for some $\text{ball}(q,r)$ in $\mathbb{R}^d$. If $r \geq \mu$, then set $r' = 2r$ and Lemma 30.3.6 implies that $\|\pi \cap \text{ball}(q,r')\| \geq \|\pi \cap \text{ball}(q, r+\mu)\| \geq \|\pi' \cap \text{ball}(q,r)\| > 6cr = 3cr'$, which contradicts that π is c-packed.

If $r < \mu$ then let U denote the segments of π' intersecting $\text{ball}(q,r)$ and let $k = |U|$. Observe that $k > 6cr/2r = 3c$, as any segment can contribute at most $2r$ to the length of π' inside $\text{ball}(q,r)$. Therefore we have that $\|\pi' \cap \text{ball}(q,2\mu)\| \geq \|\pi' \cap \text{ball}(q, r+\mu)\| \geq \|U \cap \text{ball}(q, r+\mu)\| \geq k\mu$, since every segment of the simplified curve π' has a minimal length of μ. By Lemma 30.3.6, this implies that $\|\pi \cap \text{ball}(q,3\mu)\| \geq \|\pi' \cap \text{ball}(q,2\mu)\| \geq k\mu > 3c\mu$, which is a contradiction to the c-packedness of π. ■

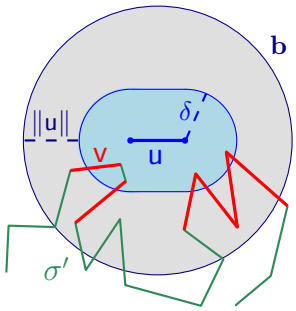
30.3.3.2. Bounding the relative complexity of c-packed curves

Lemma 30.3.8. *For any two c-packed curves π and σ in $\mathbb{R}^d$, with a total of n edges, and $0 < \varepsilon < 1$, we have that $\mathsf{N}(\varepsilon, \pi, \sigma) = O(cn/\varepsilon)$.*

Proof: Let $\delta \geq 0$ be an arbitrary number, $\mu = \varepsilon\delta$, $\pi' = \text{simpl}(\pi, \mu)$ and $\sigma' = \text{simpl}(\sigma, \mu)$.

We need to show that the complexity of $D_{\leq\delta}(\pi', \sigma')$ is $O(cn/\varepsilon)$. A free space cell of $D_{\leq\delta}(\pi', \sigma')$ corresponds to two segments $u \in \pi'$ and $v \in \sigma'$. The free space in this cell is non-empty if and only if there are two points $q \in u$ and $s \in v$ such that $\|q - s\| \leq \delta$. We charge this pair of points to the shorter of the two segments. We claim that a segment cannot be charged too many times.

Indeed, consider a segment $u \in \pi'$, and consider the ball h of radius $r = (3/2)\|u\| + \delta$ centered at the midpoint of u, see the figure on the right. Every segment $v \in \sigma'$ that participates in a close pair as above and charges u for it, is of length at least $\|u\|$, and the length of $v \cap \mathsf{h}$ is at least $\|u\|$. Since σ' is $6c$-packed, by Lemma 30.3.7, we have that the number of such charges is at most



$$c' = \frac{\|\sigma' \cap \mathsf{h}\|}{\|u\|} \leq \frac{6cr}{\|u\|} = \frac{6c((3/2)\|u\| + \delta)}{\|u\|} \leq 9c + \frac{6c\delta}{\mu} = O\left(\frac{c}{\varepsilon}\right),$$

since $\|u\| \geq \mu$.

We conclude that there are at most $c'n$ free space cells that contain a point of $D_{\leq\delta}$. The complexity of the free space inside a cell is a constant, thus implying the claim. ■

30.4. Approximation algorithm for the Fréchet distance

30.4.1. Approximate decider

Algorithm 30.4.1. The input is two polygonal curves π and σ, and parameters ε and δ.

Let $\mu = (\varepsilon/4)\delta$. The algorithm computes in linear time the curves $\pi' = \text{simpl}(\pi, \mu)$ and $\sigma' = \text{simpl}(\sigma, \mu)$ using Algorithm 30.3.1. Let $\delta' = \delta + 2\mu$ and observe that $\mu/\delta' = \varepsilon/(4 + 2\varepsilon)$. Using Lemma 30.2.8 the algorithm decides whether $\mathsf{d}_{\mathcal{F}}(\pi', \sigma') \leq \delta'$. If so the algorithm output the reparameterizations as a proof that the Fréchet distance is $\leq (1+\varepsilon)\delta$. Otherwise, the algorithm outputs "$\mathsf{d}_{\mathcal{F}}(\pi, \sigma) > \delta$".

Lemma 30.4.2. *Let π and σ be polygonal curves in $\mathbb{R}^d$, and let $\varepsilon > 0$ and $\delta > 0$ be two parameters. Then, Algorithm 30.4.1 output, in $O(\mathsf{N}(\varepsilon, \pi, \sigma))$ time, one of the following:*

(A) "$\mathsf{d}_{\mathcal{F}}(\pi, \sigma) \leq (1+\varepsilon)\delta$", and reparameterizations of π and σ of width $\leq (1+\varepsilon)\delta$, and this happens if $\mathsf{d}_{\mathcal{F}}(\pi, \sigma) \leq \delta$.

(B) "$\mathsf{d}_{\mathcal{F}}(\pi,\sigma) > \delta$" *if* $\mathsf{d}_{\mathcal{F}}(\pi,\sigma) > (1+\varepsilon)\delta$.
(C) If $\mathsf{d}_{\mathcal{F}}(\pi,\sigma) \in (\delta,(1+\varepsilon)\delta]$ *then the algorithm outputs either of the above outcomes.*
In either case, the statement returned is correct.

Proof: Using Lemma 30.2.8 we can decide whether $\mathsf{d}_{\mathcal{F}}(\pi',\sigma') \leq \delta'$ in

$$O(N_{\leq\delta'}(\pi',\sigma')) = O(\mathsf{N}(\mu/\delta',\pi,\sigma)) = O(\mathsf{N}(\varepsilon/(4+2\varepsilon),\pi,\sigma)) = O(\mathsf{N}(\varepsilon,\pi,\sigma))$$

time, by assumption (P(P1)). If so, we have that

$$\mathsf{d}_{\mathcal{F}}(\pi,\sigma) \leq \mathsf{d}_{\mathcal{F}}(\pi,\pi') + \mathsf{d}_{\mathcal{F}}(\pi',\sigma') + \mathsf{d}_{\mathcal{F}}(\sigma',\sigma) \leq \delta' + 2\mu = \delta + 4(\varepsilon/4)\delta = (1+\varepsilon)\delta.$$

On the other hand, if $\mathsf{d}_{\mathcal{F}}(\pi',\sigma') > \delta'$, then this implies, by the triangle inequality, that

$$\mathsf{d}_{\mathcal{F}}(\pi,\sigma) \geq \mathsf{d}_{\mathcal{F}}(\pi',\sigma') - \mathsf{d}_{\mathcal{F}}(\pi,\pi') - \mathsf{d}_{\mathcal{F}}(\sigma',\sigma) > \delta' - 2\mu = \delta.$$

Therefore, the algorithm outputs "$\mathsf{d}_{\mathcal{F}}(\pi,\sigma) > \delta$" in this case. ■

30.4.2. Approximating the Fréchet distance in an interval

How to use the approximate decider in a binary search. In order to use Lemma 30.4.2 to perform a binary search for the Fréchet distance, we can turn the "fuzzy" decision procedure into a precise one as follows.

Lemma 30.4.3. *Let* π *and* σ *be two polygonal curves in* $\mathbb{R}^d$*, and let* $1 \geq \varepsilon > 0$ *and* $\delta > 0$ *be two parameters. Then, there is an algorithm* decider$(\pi,\sigma,\delta,\varepsilon)$ *that, in* $O(\mathsf{N}(\varepsilon,\pi,\sigma))$ *time, returns one of the following outputs: (i) a* $(1+\varepsilon)$*-approximation to* $\mathsf{d}_{\mathcal{F}}(\pi,\sigma)$*, (ii)* $\mathsf{d}_{\mathcal{F}}(\pi,\sigma) < \delta$*, or (iii)* $\mathsf{d}_{\mathcal{F}}(\pi,\sigma) > \delta$*. The answer returned is correct.*

Proof: Let $\delta' = \delta/(1+\varepsilon')$, for $\varepsilon' = c\varepsilon$, $c = 1/3$. We run the algorithm of Lemma 30.4.2 with parameters δ and ε'. If the call returns "$\mathsf{d}_{\mathcal{F}}(\pi,\sigma) > \delta$", then we return this result.

Otherwise, we call Lemma 30.4.2 with parameters δ' and ε'. If it returns that "$\mathsf{d}_{\mathcal{F}}(\pi,\sigma) \leq (1+\varepsilon')\delta'$" then $\mathsf{d}_{\mathcal{F}}(\pi,\sigma) \leq (1+\varepsilon')\delta' = \delta$, and we return this result.

The only remaining possibility is that the two calls returned "$\mathsf{d}_{\mathcal{F}}(\pi,\sigma) \leq (1+\varepsilon')\delta$" and "$\mathsf{d}_{\mathcal{F}}(\pi,\sigma) > \delta'$". But then we have found the required approximation. Therefore, the resulting approximation factor of the reparameterizations returned by the call with δ is $\leq \dfrac{(1+\varepsilon')\delta}{\delta'} = (1+c\varepsilon)^2 < (1+\varepsilon)$ as can be easily verified, since $0 < \varepsilon \leq 1$. ■

Searching for the Fréchet Distance – fixed interval. It is now straightforward to perform a binary search on an interval $[\alpha,\beta]$ to approximate the value of the Fréchet distance, if it falls inside this interval. Indeed, partition this interval into subintervals of length $\varepsilon\alpha$ and perform a binary search to find the interval that contains the Fréchet distance. There are $O(\beta/\varepsilon\alpha)$ intervals, and this would require $O(\log(\beta/\varepsilon\alpha))$ calls to decider. By using exponential subintervals, one can do slightly better, as testified by the following lemma. We thus get the following.

Lemma 30.4.4. *Given two curves* π *and* σ *in* $\mathbb{R}^d$*, a parameter* $1 \geq \varepsilon > 0$*, and an interval* $[\alpha,\beta]$*, one can perform a binary search in* $[\alpha,\beta]$ *and obtain a* $(1+\varepsilon)$*-approximation to* $\mathsf{d}_{\mathcal{F}}(\pi,\sigma)$ *if* $\mathsf{d}_{\mathcal{F}}(\pi,\sigma) \in [\alpha,\beta]$*, or report that* $\mathsf{d}_{\mathcal{F}}(\pi,\sigma) \notin [\alpha,\beta]$*. The algorithm, denoted by* searchInterval$(\pi,\sigma,[\alpha,\beta],\varepsilon)$*, takes* $O\left(\log\dfrac{\beta}{\varepsilon\alpha}\right)$ *calls to* decider*.*

30.4.3. Searching over critical events efficiently

Clearly, the procedure searchInterval$(\pi, \sigma, [\alpha, \beta], \varepsilon)$ alone does not suffice to solve our main problem, since the interval of distances we are searching over might have arbitrarily large "spread" (i.e., $\log \beta/\alpha$ might be arbitrarily large). However, the Fréchet distance must be sufficiently close to a free space event in one of the "approximate" diagrams, i.e., a free space diagram of the two simplified curves. Thus, we can identify two kinds of critical values to search over, which are candidate values for the approximate Fréchet distance. These are the events where (i) the simplification of an input curve changes, or (ii) the reachability within the approximate free space diagram changes (i.e., a free space event; see Section 30.2.4).

We use a simple "approximate" search over the critical events. It is sufficient to search over a set of values which approximate the event values by a constant factor, since we will use Lemma 30.4.4 to refine the resulting search interval in the main algorithm. Note, for instance, that we can easily use this lemma to turn a constant factor approximation of the Fréchet distance into a $(1+\varepsilon)$-approximation.

Algorithm 30.4.5. Let searchEvents(π, σ, Z) denote the algorithm that performs a binary search over the values of Z, to compute the atomic interval of Z that contains the Fréchet distance between π and σ. This procedure uses *decider* (Lemma 30.4.3) to perform the decisions during the search.

30.4.3.1. Searching over Simplifications

Consider the events when the simplified curves change, see Algorithm 30.3.1. Consider the set of all pairwise distances between vertices of π and σ. Observe that it breaks the real line into $\binom{n}{2}+1$ atomic intervals, such that in each such interval the simplification does not change. Thus simpl(π, μ) (resp. simpl(σ, μ)) might result in $O(n^2)$ different curves depending on the value of μ, where n is the total number of vertices of π and σ. As a first step we would therefore like to use Algorithm 30.4.5 to perform a binary search over those distances to find the atomic interval that contains the required Fréchet distance. Naively, there are $\Theta(n^2)$ distances to consider, but fortunately, one can reduce the number to linear number of distances using well-separated pairs decomposition (WSPD).

Lemma 30.4.6. *Given a set P of n points in $\mathbb{R}^d$. Then, one can compute, in $O(n \log n)$ time, a set Z of $O(n)$ numbers, such that for any $y \in \mathcal{D}(P)$, there exist numbers $x, x' \in Z$ such that $x \leq y \leq x' \leq 2x$, where $\mathcal{D}(P)$ is the set of all pairwise distances in P. Let approxDistances(P) denote this algorithm.*

Proof: Compute an 8-well-separated pairs decomposition of P. By Theorem 30.7.1 this can be done in $O(n \log n)$ time, and results in a set of pairs of subsets $\{(X_1, Y_1), \ldots, (X_m, Y_m)\}$, where $m = O(n)$, such that for any two points $q, s \in P$ there exists a pair (X_i, Y_i) in the above decomposition, such that: (i) $q \in X_i$ and $s \in Y_i$ (or vice versa), and (ii) $\max(\operatorname{diam}(X_i), \operatorname{diam}(Y_i)) \leq \min_{q_i \in X_i, s_i \in Y_i} \|q_i - s_i\| /8$.

This implies that the distance of any pair of points in X_i and Y_i, respectively, are the same up to a small constant. As such, for every pair (X_i, Y_i), for $i = 1, \ldots, m$, we pick representative points $q_i \in X_i$ and $s_i \in Y_i$, and set $\ell_i = (3/4)\|q_i - s_i\|$. Let $Z = \{\ell_1, \ldots, \ell_m, 2\ell_1, \ldots, 2\ell_m\}$ be the computed set of values.

Consider any pair of points $q, s \in P$. For the specific pair (X_i, Y_i) that contains the pair of points q and s that we are interested in, we have that $\ell_i = (3/4)\|q_i - s_i\| \leq \|q_i - s_i\| - \operatorname{diam}(X_i) - \operatorname{diam}(Y_i) \leq \|q - s\| \leq \|q_i - s_i\| + \operatorname{diam}(X_i) + \operatorname{diam}(Y_i) \leq (5/4)\|q_i - s_i\| \leq 2\ell_i$, thus establishing the claim. ■

30.4.3.2. Monotonicity Events

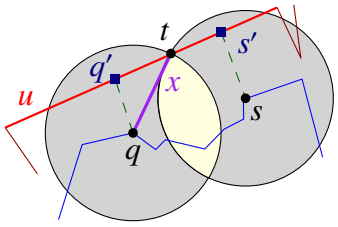
The following lemma testifies that the radius of a monotonicity event must be "close" to either a vertex-edge event or to the distance between two vertices. Since we approximate the vertex-vertex

distances and perform a binary search over them, this implies that we only need to consider vertex-edge events. Furthermore, by Observation 30.2.9, the number of those vertex-edge events which remain in the resulting search range can be bounded by the complexity of the reachable free space.

Lemma 30.4.7. *Let x be the radius of a monotonicity event involving vertices q, s and a segment u. Then there exists a number y such that $y/2 \leq x \leq 3y$, and y is either in $\mathcal{W}$ or y is the radius of a vertex-edge event, where $\mathcal{W} = \mathcal{D}(V(\pi) \cup V(\sigma))$ is the set of all distances between pairs of points of $V(\pi) \cup V(\sigma)$.*

Proof: Let t be the intersection point of $\mathbb{S}(q, x) \cap \mathbb{S}(s, x)$ which lies on u. Let q' (resp. s') be the closest point on u to q (resp. s).

Clearly $\|q' - s'\| \leq \|q - s\|$ (since the projection onto the nearest neighbor of a convex set is a contraction), and since $q' \in \text{ball}(q, x)$ and $s' \in \text{ball}(s, x)$, the point t lies on the segment $q's'$.



This implies that $x = \|q - t\| \leq \|q - q'\| + \|q' - t\| \leq \|q - q'\| + \|q' - s'\| \leq \|q - q'\| + \|q - s\|$, by the triangle inequality.

A similar argument implies that

$$x = \|q - t\| \geq \|q - q'\| - \|q' - t\| \geq \|q - q'\| - \|q' - s'\| \geq \|q - q'\| - \|q - s\| .$$

If $\|q - q'\| \geq 2\|q - s\|$ then the above implies that $x \in [1/2, 3/2]\,\|q - q'\|$. If q' is an endpoint of u then $\|q - q'\|$ is in $\mathcal{W}$. Otherwise, $\|q - q'\|$ is the radius of the vertex-edge event between q and u. In either case, this implies the claim.

If $\|q - q'\| \leq 2\|q - s\|$ then $x = \|q - t\| \leq \|q - q'\| + \|q - s\| \leq 2\|q - s\| + \|q - s\| = 3\|q - s\|$, and of course $\|q - s\| \in \mathcal{W}$. Now, the two balls of radius x centered at q and s, respectively, cover the segment qs, and we have that $\|q - s\|/2 \leq x$, which implies the claim. ■

30.4.4. Searching with a fixed simplification

Assume that we have found simplifications τ and η, such that the Fréchet distance of those curves yields the desired $(1 + \varepsilon)$-approximation. Clearly, an approximation of $\mathsf{d}_{\mathcal{F}}(\tau, \eta)$ suffices for our result. To this end, let *searchIntervalNoSimp*$(\pi, \sigma, [\alpha, \beta], \varepsilon)$ be the variant of searchInterval from Lemma 30.4.4 that uses Lemma 30.2.8 directly instead of calling decider. This version searches for the Fréchet distance in the given interval, but does not perform simplification before calling the decision procedure. It returns a $(1 + \varepsilon)$-approximation of the Fréchet distance, given that it is contained in this interval. Note that correctness and running time of Lemma 30.4.4 are not affected by this modification.

Lemma 30.4.8. *Let τ and η be two given curves in $\mathbb{R}^d$, with total complexity n, and let $[h^-, h^+]$ be an interval, such that (i) $\mathsf{d}_{\mathcal{F}}(\tau, \eta) \in [h^-, h^+]$, and (ii) there is no value of $\mathcal{W} = \mathcal{D}(V(\tau) \cup V(\eta))$ in the interval $[h^-, h^+]$. Then, for $\varepsilon > 0$, one can $(1+\varepsilon)$-approximate $\mathsf{d}_{\mathcal{F}}(\tau, \eta)$ and compute reparametrizations in $O((n + N)\log(N/\varepsilon))$ time, where $N = N_{\leq h^+}(\tau, \eta)$.*

Let aprxFréchetNoSimp$(\tau, \eta, [h^-, h^+], \varepsilon)$ denote this algorithm.

Proof: For two real numbers $x, y > 0$, we define $[x/y] = \max(x, y)/\min(x, y)$.

Compute $F_{\leq h^+}(\tau, \eta)$, using Lemma 30.2.8. Next, using Observation 30.2.9, compute from $F_{\leq h^+}(\tau, \eta)$ the set Z of all the radii of the vertex-edge events of τ and η with radius at most h^+. Next, we sort Z, and perform a binary search over Z, using Lemma 30.2.8, for the atomic interval $\mathfrak{I} = [\alpha, \beta]$ of Z

aprxFréchetI(π, σ, ε)

(A) $P = V(\pi) \cup V(\sigma)$

(B) $Z \leftarrow$ *approxDistances*(P) (Lemma 30.4.6).

(C) $[\alpha, \beta] \leftarrow$ *searchEvents*$(\pi, \sigma, Z, \varepsilon)$ (Algorithm 30.4.5).

(D) Call searchInterval$(\pi, \sigma, [\alpha, 4\alpha'], \varepsilon)$, where $\alpha' = (30/\varepsilon)\alpha$ (Lemma 30.4.4).

(E) Call searchInterval$(\pi, \sigma, [\beta'/4, \beta], \varepsilon)$, where $\beta' = \beta/3$.

(F) Let $\pi' = \text{simpl}(\pi, \mu)$ and $\sigma' = \text{simpl}(\sigma, \mu)$, for $\mu = 3\alpha$ (Algorithm 30.3.1)

(G) $\delta \leftarrow$ *aprxFréchetNoSimp*$(\pi', \sigma', [\alpha', \beta'], \varepsilon/4)$ (Lemma 30.4.8).

(H) Compute and return the resulting reparameterizations of π and σ and their width as the approximation.

Figure 30.13: The basic approximation algorithm.

that contains the Fréchet distance $\mathsf{d}_{\mathcal{F}}(\tau, \eta)$. Next, call searchIntervalNoSimp$(\tau, \eta, [\alpha, 4\alpha], \varepsilon)$ and searchIntervalNoSimp$(\tau, \eta, [\beta/4, \beta], \varepsilon)$. We claim that one of these two searches performed on the respective intervals will discover two consecutive values x and $(1+\varepsilon)x$, such that the two corresponding calls to the algorithm of Lemma 30.4.4 imply that $\mathsf{d}_{\mathcal{F}}(\tau, \eta) \in [x, (1+\varepsilon)x]$.

Indeed, the interior of $[\alpha, \beta]$ does not contain any value in $\mathcal{W}$ or a radius of a vertex-edge event of τ and η. Therefore, the interval $[\alpha, \beta]$ might contain only monotonicity events of τ and η. By Lemma 30.4.7, for a monotonicity event with radius r there exists a $y \in Z \cup \mathcal{W}$, such that $[r/y] \leq 3$. But since there is no value of $Z \cup \mathcal{W}$ in the interior of $[\alpha, \beta]$, and therefore, for any $r'' \in [4\alpha, \beta/4]$ and $y'' \in Z \cup \mathcal{W}$, we have that $[r''/y''] \geq 4$.

We conclude that no monotonicity event, vertex-edge event, or value of $\mathcal{W}$ lies in the interval $[4\alpha, \beta/4]$. Since the Fréchet distance must be equal to one such value, it follows that $\mathsf{d}_{\mathcal{F}}(\tau, \eta) \notin (4\alpha, \beta/4)$, but this implies that either $\mathsf{d}_{\mathcal{F}}(\tau, \eta) \in [\alpha, 4\alpha]$ or $\mathsf{d}_{\mathcal{F}}(\tau, \eta) \in [\beta/4, \beta]$. In either case, the above algorithm would have found the approximate distance.

Computing and sorting the set of vertex-edge events takes $O(N \log N)$ time by Observation 30.2.9. The binary search requires $O(\log |Z|)$ calls to the algorithm of Lemma 30.2.8. The two calls to searchIntervalNoSimp require $O(\log(1/\varepsilon))$ calls to Lemma 30.2.8. Now, observe that all these calls to the algorithm of Lemma 30.2.8 are done with values of $\delta \leq h^+$. Thus the complexity of the reachable free space is bounded (up to a constant factor) by the number of vertex-edge events of values $\leq h^+$, and this number is bounded by $|Z|$. Therefore, a call to Lemma 30.2.8 takes $O(|Z|)$ time. Thus, the overall running time is $O((n + |Z|) \log(|Z|/\varepsilon))$, and by definition $|Z| = O(N_{\leq h^+}(\tau, \eta))$. ■

30.4.5. The Approximation Algorithm

The resulting approximation algorithm is depicted in Figure 30.13. It will be used by the final approximation algorithm as a subroutine. We first analyze this basic algorithm. The algorithm depicted in Figure 30.13 performs numerous calls to decider, with approximation parameter $\varepsilon > 0$. If any of these calls discover the approximate distance, then the algorithm immediately stops and returns the approximation. Therefore, at any point in the execution of the algorithm, the assumption is that all previous calls to decider returned a direction where the optimal distance must lie. In particular, a call to searchInterval$(\pi, \sigma, \mathcal{I}, \varepsilon)$, would either find the approximate distance in the interval $\mathcal{I}$ and return immediately, or the desired value is outside this interval.

30.4.5.1. Correctness

Lemma 30.4.9. *Given two polygonal curves π and σ, and a parameter $1 > \varepsilon > 0$, the algorithm aprxFréchetI$(\pi, \sigma, \varepsilon)$ computes a $(1+\varepsilon)$-approximation to $\mathsf{d}_{\mathcal{F}}(\pi, \sigma)$.*

Proof: If the algorithm found the approximation before step ((F)), then clearly it is the desired approximation, and we are done. (In particular, this must be the case if $4\alpha' > \beta'/4$.)

Otherwise, because of ((C)), we know that $\mathsf{d}_{\mathcal{F}}(\pi, \sigma) \in [\alpha, \beta]$. By steps ((D)) and ((E)) it must be that $\mathsf{d}_{\mathcal{F}}(\pi, \sigma) \in [4\alpha', \beta'/4]$. Since $\mu = 3\alpha = (\varepsilon/10)\alpha' \leq \beta'/4$, it follows, by the triangle inequality, that

$$\mathsf{d}_{\mathcal{F}}(\pi', \sigma') \leq \mathsf{d}_{\mathcal{F}}(\pi', \pi) + \mathsf{d}_{\mathcal{F}}(\pi, \sigma) + \mathsf{d}_{\mathcal{F}}(\sigma, \sigma') \leq 2\mu + \beta'/4 < \beta'.$$

A similar argument shows that $\mathsf{d}_{\mathcal{F}}(\pi', \sigma') > \alpha'$. Hence, the algorithm of Lemma 30.4.8 can be applied to π' and σ' for the range $[\alpha', \beta']$, as $\mathsf{d}_{\mathcal{F}}(\pi', \sigma') \in [\alpha', \beta']$.

Now, by Lemma 30.4.8, we have that the value δ resulting from step ((G)), is contained in the interval $[\mathsf{d}_{\mathcal{F}}(\pi', \sigma'), (1+\varepsilon/4)\mathsf{d}_{\mathcal{F}}(\pi', \sigma')]$. By the triangle inequality we conclude that the returned Fréchet distance is

$$\begin{aligned}\delta &\leq \mathsf{d}_{\mathcal{F}}(\pi, \pi') + \delta + \mathsf{d}_{\mathcal{F}}(\sigma, \sigma') \leq \mathsf{d}_{\mathcal{F}}(\pi, \pi') + (1+\varepsilon/4)\mathsf{d}_{\mathcal{F}}(\pi', \sigma') + \mathsf{d}_{\mathcal{F}}(\sigma', \sigma) \\ &\leq (1+\varepsilon/4)(2\mu + \mathsf{d}_{\mathcal{F}}(\pi, \sigma) + 2\mu) \leq 5\mu + (1+\varepsilon/4)\mathsf{d}_{\mathcal{F}}(\pi, \sigma) \leq (1+\varepsilon)\mathsf{d}_{\mathcal{F}}(\pi, \sigma),\end{aligned}$$

since $5\mu = 15\alpha = (\varepsilon/2)(30/\varepsilon)\alpha = (\varepsilon/2)\alpha' \leq (\varepsilon/2)\mathsf{d}_{\mathcal{F}}(\pi, \sigma)$.

Note that $\delta \geq \mathsf{d}_{\mathcal{F}}(\pi, \sigma)$ since it is the width of a specific reparameterization between the two curves. ■

30.4.5.2. Running Time

Lemma 30.4.10. *For any $x, y \in (2\alpha, \beta/2)$, we have $\mathrm{simpl}(\pi, x) = \mathrm{simpl}(\pi, y)$ and $\mathrm{simpl}(\sigma, x) = \mathrm{simpl}(\sigma, y)$.*

Proof: Indeed, the interval (α, β) does not contain any value of Z. As such, by Lemma 30.4.6, $(2\alpha, \beta/2)$ does not contain any value of the pairwise distances between vertices of the vertex set of π and σ which implies that the simplification is the same for any value inside this interval. ■

Lemma 30.4.11. *Given two polygonal curves π and σ with a total of n vertices in $\mathbb{R}^d$, and a parameter $1 > \varepsilon > 0$, the running time of aprxFréchetI$(\pi, \sigma, \varepsilon)$ is $O(\mathsf{N}(\varepsilon, \pi, \sigma) \log n)$.*

Proof: Computing Z (and sorting it) takes $O(n \log n)$ time by Lemma 30.4.6. Steps ((C)), ((D)) and ((E)) perform $O(\log n + \log(1/\varepsilon)) = O(\log n)$ calls to decider, by Lemma 30.4.4. (Here, we assume that $\varepsilon = \Omega(1/n)$. If $\varepsilon < 1/n$ then we can just use the algorithm of Alt and Godau [AG95] since its running time is faster than our approximation algorithm in this case.) Each call to decider takes $O(\mathsf{N}(\varepsilon, \pi, \sigma))$ time, so overall this takes $O(\mathsf{N}(\varepsilon, \pi, \sigma) \log n)$ time. Computing the simplifications in step ((F)) with Algorithm 30.3.1 takes $O(n)$ time.

By Lemma 30.4.8, a call to aprxFréchetNoSimp$(\pi', \sigma', [\alpha', \beta'], \varepsilon/4)$ takes $T = O((n+N)\log(N/\varepsilon))$ time, with $N = N_{\leq \beta'}(\pi', \sigma')$. Now, 3α and β' are both inside the interval $(2\alpha, \beta/2)$, and as such, by Lemma 30.4.10, we have that $\pi' = \mathrm{simpl}(\pi, 3\alpha) = \mathrm{simpl}(\pi, \beta')$ and $\sigma' = \mathrm{simpl}(\sigma, 3\alpha) = \mathrm{simpl}(\sigma, \beta')$. Therefore, we have that

$$N = N_{\leq \beta'}(\pi', \sigma') = N_{\leq \beta'}(\mathrm{simpl}(\pi, \beta'), \mathrm{simpl}(\sigma, \beta')) \leq \mathsf{N}(1, \pi, \sigma).$$

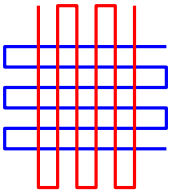
Thus, step ((G)) takes $T = O(\mathsf{N}(1, \pi, \sigma) \log(\mathsf{N}(1, \pi, \sigma) n/\varepsilon)) = O(\mathsf{N}(1, \pi, \sigma) \log n)$, time since $\mathsf{N}(1, \pi, \sigma) \leq n^2$ and $\varepsilon = \Omega(1/n)$. Observe that $\mathsf{N}(1, \pi, \sigma) \leq \mathsf{N}(\varepsilon, \pi, \sigma)$ for $\varepsilon \leq 1$.

Finally, in order to compute the resulting reparameterizations in step ((H)), we compute the reparametrizations of π and π' (resp. σ and σ') as described in the proof of Lemma 30.3.2 and chain them with the reparameterizations of the simplified curves, which we obtained from step ((G)). Clearly, this and computing the resulting width takes $O(n)$ time. The term $\mathsf{N}(\varepsilon, \pi, \sigma)$ dominates over $O(n)$. ■

30.4.6. The Result

Putting the above together, we get the following result.

Theorem 30.4.12. *Given two polygonal curves π and σ with a total of n vertices in $\mathbb{R}^d$, and a parameter $1 > \varepsilon > 0$, one can $(1+\varepsilon)$-approximate the Fréchet distance between π and σ in $O(\mathsf{N}(\varepsilon, \pi, \sigma) \log n)$ time (see Definition 30.3.3).*



As mentioned earlier, the use of simplification by itself is not sufficient to guarantee that the presented algorithm is efficient. Indeed, in might not be possible to simplify the input curves at all without losing too much information. See the figure to the left for one such example. Fortunately, we can do much better for c-packed curves. Plugging the bound on the relative complexity of c-packed curves (Lemma 30.3.8) into Theorem 30.4.12, implies the following.

Theorem 30.4.13. *Given two polygonal c-packed curves π and σ with a total of n vertices in $\mathbb{R}^d$, and a parameter $1 > \varepsilon > 0$, one can $(1+\varepsilon)$-approximate the Fréchet distance between π and σ in $O((cn/\varepsilon) \log n)$ time.*

30.5. Bibliographical notes

The first algorithm for computing the Fréchet distance between curves in $O(n^2 \log n)$ time is due to Alt and Godau [AG95], which essentially started the study of this problem in Computational Geometry. Their algorithm uses parametric search, which is quite complicated, and our simplified algorithm is due to Har-Peled and Raichel [HR14]. The decider procedure description (Section 30.2.3) is taken from Driemel et al. [DHW12], who also introduced the class of c-packed curves, and provided the approximation algorithm described in Section 30.4 – their algorithm is slightly faster because they work harder.

The class of c-packed curve is quite wide, and is related to other classes such as low density curves. See Driemel et al. [DHW12] for details.

Using input models as a way to bridge the gap between worst case analysis of algorithms and their much better practical behavior. These so-called *realistic input models* are commonly used for the analysis of problems where the worst case complexity is dominated by degenerate or contrived configurations which are highly unlikely to occur in practice, see [BKSV02] for an overview. The class of c-packed

Curve simplification. There is extensive literature on curve simplification, as this is an important problem in practice. See Agarwal et al. [AHMW05] and references therein. In particular, Agarwal et al. [AHMW05] suggested a more aggressive (but slightly slower and more complicated) simplification algorithm than the one we used.

Har-Peled and Raichel [HR14] presented an incremental scheme for curve simplification – the algorithm computes an ordering the vertices of the polygonal curve, such that for any k, the first k vertices according to this ordering, gives a close to optimal approximation to the original curve with k vertices.

Lower bound. A lower bound of $\Omega(n \log n)$ was given by Buchin et al. [BBK+07] for the decision problem. Alt [Alt09] conjectured that the decision problem may be 3SUM-hard – this is still open. A major breakthrough was made by Bringmann [Bri14a, Bri14b] who showed that unless SETH (a conjecture that states that SAT with n variables can not be solved in better time than $\Omega(c^n)$, for a constant $c > 1$) is false, one can not compute the Fréchet distance in better than $O(n^{2-\delta})$, for any constant $\delta > 0$. This lower bound also holds for constant approximation and the discrete case. Surprisingly, Bringmann also shows that one can not do much better for c-packed curves – that is, the running time $\Omega(cn/\sqrt{\varepsilon})$ is required.

30.5.1. A short and biased literature survey on the Fréchet distance

30.5.1.1. Applications

The Fréchet distance and its variants have been used to measure similarity between curves in applications such as dynamic time-warping [KP99], speech recognition [KHM+98], signature and handwriting recognition [MP99, SKB07], matching of time series in databases [KKS05], as well as geographic applications, such as map-matching of vehicle tracking data [BPSW05, WSP06], and moving objects analysis [BBG08, BBG+08].

30.5.1.2. Exact algorithms.

As mentioned above, for two polygonal curves of total complexity n in the plane, their Fréchet distance can be computed in $O(n^2 \log n)$ time [AG95], and their Hausdorff distance can be computed in $O(n \log n)$ time [Alt09]. Alt et al. [AKW04] shows that for closed convex curves the Hausdorff distance and the Fréchet distance are the same, an thus can be computed in linear time.

Saving some logs (or fractions of logs). Agarwal et al. [AAKS14] showed how to get (slightly) faster algorithm for the discrete Fréchet distance. They get running time $O\left(\frac{n^2 \log\log n}{\log n}\right)$ Oversimplifying, the improved running time is achieved by preprocessing blocks of the free space diagram (which are usually of logarithmic size), for fast queries, one can get such speedups. Buchin et al. [BBMM17] presented an algorithm for the continuous case, with running time $O(n^2\sqrt{\log n}/(\log\log n)^{3/2})$. They also present algorithms in the decision-tree and word RAM model. In particular, they get a roughly $O(n^2/\log n)$ algorithm for the continuous weak Fréchet case.

30.5.1.3. Data-structures

Distance between subcurve and a segment. Driemel and Har-Peled [DH13] presented a data-structures such that given a curve, one can preprocess it, such that given a segment, one can approximate the Fréchet distance between any subcurve and the segment in polylogarithmic time. They also extended this to queries made out of k segments, where the query time is roughly $O(k^2\text{polylog})$, and with constant approximation.

Approximate nearest neighbor. Driemel and Silvestri [DS17] and the earlier work by Indyk [Ind02].

30.5.1.4. Extensions

Shortcuts. One way to try and address the non-robustness of the Fréchet distance is by allowing shortcuts – namely, one can shortcut one of the curves (by traveling along a straight segment) a certain

number of times. Driemel and Har-Peled [DH13] presented a near linear time $(3+\varepsilon)$-approximation algorithm for c-packed curves (for a constant number of shortcuts). Surprisingly, the general problem of computing exactly the shortcut Fréchet distance is NP-Hard [BDS13, BDS14].

Surfaces. It is natural to try and use the Fréchet distance for surfaces. Unfortunately, the problem becomes surprisingly nasty. For general surfaces the decision problem is NP-hard [God99]. In fact, whether the Fréchet distance for general surfaces is computable is still an open problem. Recently Alt and Buchin [AB10] showed that the problem is semi-computable between surfaces, and polynomial time computable for the weak Fréchet distance. The problem is hard even if the surfaces are well-behaved terrains, see Buchin et al. [BBS10].

Taking a walk in a map. Alt et al. [AERW03] presented an $O(n^2 \log^2 n)$ time algorithm to compute the Fréchet distance between either a curve and a graph or between two graphs (where this distance can be viewed as a generalization of partial curve matching, as introduced in [AG95]). Maheshwari et al. [MSSZ11] show that for DAGs one can improve the running time by a $\log(n)$ factor for the curve to graph distance problem (they also drop $\log(n)$ factors from other problems in [AG95]).

Multiple curves. One can generalize the problem to consider an input of multiple curves. Dumitrescu and Rote [DR04] consider the problem of simultaneously minimizing the Fréchet distance between all pairs of a set of k curves. They show one can get a 2-approximation in $O(n^2 \log n)$ time, whereas the naive extension to k curves of the exact algorithm takes time at least $\Omega(n^k)$. Buchin et al. [BBK+10] consider the problem of finding the median trajectory (polygonal curve) of a set of k trajectories in the plane that all share the same starting and ending points, where the median trajectory is a trajectory contained in the union of the k input trajectories that must cross at least half of the input trajectories in order to reach the unbounded face.

Extending the notion of the free space diagram. Har-Peled and Raichel [HR14] observe that the free space diagram can be interpreted as the Cartesian product of the two curves. They extend the idea to general complexes (this idea is implicit in earlier work), thus being to solve more general problems, such as compute the two closest curves, in the Fréchet distance, that are restricted to lie in some domains. Other extensions includes problems like how to walk a pack of dogs, or find an mean curve of a given set of curves. Unfortunately, their work is restricted to using the weak Fréchet distance, and it would be interesting to extend it to the regular Fréchet distance.

30.5.1.5. Approximation algorithms

κ-bounded curves. Alt et al. showed a κ-approximation algorithms with $O(n \log n)$ running time [AKW04]. For a curve to be κ-bounded means, roughly, that for any two points on the curve the portion of the curve in between them cannot be further away from either point than $\kappa/2$ times the distance between the two points. For closed convex curves the Fréchet distance equals the Hausdorff distance and for κ-bounded curves the Fréchet distance is at most $(1+\kappa)$ times the Hausdorff distance, and hence the $O(n \log n)$ algorithm for the Hausdorff distance applies.

Backbone curves. Aronov et al. [AHK+06, AHK+15] provided a near linear time $(1+\varepsilon)$-approximation algorithm for the *discrete* Fréchet distance, which only considers distances between vertices of the curves.

Their algorithm works for *backbone curves*, which are used to model protein backbones in molecular biology. Backbone curves are required to have, roughly, unit edge length and a minimal distance between any pair of vertices. They use curve simplification to speed up their algorithm.

c-packed curves. As mentioned above Driemel et al. [DHW12] presented a $Oc(n \log n + cn/\varepsilon)$ time. The dependency on ε was improved to $cn/\sqrt{\varepsilon}$ by Bringmann and Künnemann [BK15]. Removing the $O(n \log n)$ term remains an interesting open problem.

30.5.1.6. Other papers

Other work of interest on the Fréchet distance includes [BIG13], and [DKS15, DKS16].

30.6. Exercises

Exercise 30.1 (Linear time bottleneck path). Let G be a graph with n edges and m vertices, and with unique weights on its edges. Let $\mathcal{T}$ be the MST of G.

(A) Present a deterministic linear time algorithm that computes the heaviest edge in $\mathcal{T}$. [**Hint:** Compute the edge of median weight of G. Now decide if it is heavier or lighter than the bottleneck edge.]

(B) Given two vertices s and t in G, present a $O(n+m)$ time deterministic algorithm that compute a bottleneck path between s and t, see Definition 30.1.4.

30.7. From previous lectures

Theorem 30.7.1. *For $1 \geq \varepsilon > 0$, and a set P of n points in $\mathbb{R}^d$, one can construct, in $O\left(n \log n + n\varepsilon^{-\mathsf{d}}\right)$. time, an ε^{-1}-WSPD of P of size $n\varepsilon^{-\mathsf{d}}$.*

References

[AAKS14] P. K. Agarwal, R. B. Avraham, H. Kaplan, and M. Sharir. Computing the discrete fréchet distance in subquadratic time. *SIAM J. Comput.*, 43(2): 429–449, 2014.

[AB10] H. Alt and M. Buchin. Can we compute the similarity between surfaces? *Discrete Comput. Geom.*, 43(1): 78–99, 2010.

[AERW03] H. Alt, A. Efrat, G. Rote, and C. Wenk. Matching planar maps. *J. Algorithms*, 49: 262–283, 2003.

[AG95] H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *Int. J. Comput. Geom. Appl.*, 5: 75–91, 1995.

[AHK+06] B. Aronov, S. Har-Peled, C. Knauer, Y. Wang, and C. Wenk. Fréchet distance for curves, Revisited. *Proc. 14th Annu. Euro. Sympos. Alg.* (ESA), 52–63, 2006.

[AHK+15] B. Aronov, S. Har-Peled, C. Knauer, Y. Wang, and C. Wenk. Fréchet distance for curves, revisited. *CoRR*, abs/1504.07685, 2015.

[AHMW05] P. K. Agarwal, S. Har-Peled, N. Mustafa, and Y. Wang. Near-linear time approximation algorithms for curve simplification in two and three dimensions. *Algorithmica*, 42(3-4): 203–219, 2005.

[AKW04] H. Alt, C. Knauer, and C. Wenk. Comparison of distance measures for planar curves. *Algorithmica*, 38(1): 45–58, 2004.

[Alt09] H. Alt. The computational geometry of comparing shapes. *Efficient Algorithms: Essays Dedicated to Kurt Mehlhorn on the Occasion of His 60th Birthday*. Springer-Verlag, 2009, pp. 235–248.

[BBG+08] K. Buchin, M. Buchin, J. Gudmundsson, M. L., and J. Luo. Detecting commuting patterns by clustering subtrajectories. *Proc. 19th Annu. Internat. Sympos. Algorithms Comput.* (ISAAC), 644–655, 2008.

[BBG08] K. Buchin, M. Buchin, and J. Gudmundsson. Detecting single file movement. *Proc. 16th ACM SIGSPATIAL Int. Conf. Adv. GIS,* 288–297, 2008.

[BBK+07] K. Buchin, M. Buchin, C. Knauer, G. Rote, and C. Wenk. How difficult is it to walk the dog? *Proc. 23rd Euro. Workshop on Comput. Geom.,* 170–173, 2007.

[BBK+10] K. Buchin, M. Buchin, M. Kreveld, M. Löffler, R. I. Silveira, C. Wenk, and L. Wiratma. Median trajectories. *Proc. 18th Annual European Symposium on Algorithms (ESA),* vol. 6346. 463–474, 2010.

[BBMM17] K. Buchin, M. Buchin, W. Meulemans, and W. Mulzer. Four soviets walk the dog: improved bounds for computing the Fréchet distance. *Discrete Comput. Geom.*, 58(1): 180–216, 2017.

[BBS10] K. Buchin, M. Buchin, and A. Schulz. Fréchet distance of surfaces: some simple hard cases. *Proc. 18th Annu. Euro. Sympos. Alg.* (ESA), 63–74, 2010.

[BDS13] M. Buchin, A. Driemel, and B. Speckmann. Computing the Fréchet distance with shortcuts is np-hard. *CoRR*, abs/1307.2097, 2013.

[BDS14] M. Buchin, A. Driemel, and B. Speckmann. Computing the fréchet distance with shortcuts is NP-hard. *Proc. 30th Annu. Sympos. Comput. Geom.* (SoCG), 367, 2014.

[BIG13] M. de Berg, A. F. C. IV, and J. Gudmundsson. Fast fréchet queries. *Comput. Geom. Theory Appl.*, 46(6): 747–755, 2013.

[BK15] K. Bringmann and M. Künnemann. Improved approximation for fréchet distance on c-packed curves matching conditional lower bounds. *Proc. 26th Annu. Internat. Sympos. Algorithms Comput.* (ISAAC), vol. 9472. 517–528, 2015.

[BKSV02] M. de Berg, M. J. Katz, A. v. Stappen, and J. Vleugels. Realistic input models for geometric algorithms. *Algorithmica*, 34(1): 81–97, 2002.

[BPSW05] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On map-matching vehicle tracking data. *Proc. 31st VLDB Conference,* 853–864, 2005.

[Bri14a] K. Bringmann. Why walking the dog takes time: frechet distance has no strongly subquadratic algorithms unless SETH fails. *Proc. 55th Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS), 661–670, 2014.

[Bri14b] K. Bringmann. Why walking the dog takes time: frechet distance has no strongly subquadratic algorithms unless SETH fails. *CoRR*, abs/1404.1448, 2014.

[DH13] A. Driemel and S. Har-Peled. Jaywalking your dog – computing the Fréchet distance with shortcuts. *SIAM J. Comput.*, 42(5): 1830–1866, 2013.

[DHW12] A. Driemel, S. Har-Peled, and C. Wenk. Approximating the Fréchet distance for realistic curves in near linear time. *Discrete Comput. Geom.*, 48(1): 94–127, 2012.

[DKS15] A. Driemel, A. Krivosija, and C. Sohler. Clustering time series under the fréchet distance. *CoRR*, abs/1512.04349, 2015.

[DKS16] A. Driemel, A. Krivosija, and C. Sohler. Clustering time series under the fréchet distance. *Proc. 27th ACM-SIAM Sympos. Discrete Algs.* (SODA), 766–785, 2016.

[DR04] A. Dumitrescu and G. Rote. On the Fréchet distance of a set of curves. *Proc. 16th Canad. Conf. Comput. Geom.* (CCCG), 162–165, 2004.

[DS17] A. Driemel and F. Silvestri. Locality-sensitive hashing of curves. *CoRR*, abs/1703.04040, 2017.

[God99] M. Godau. *On the complexity of measuring the similarity between geometric objects in higher dimensions*. PhD thesis. Free University of Berlin, 1999.

[HR14] S. Har-Peled and B. Raichel. The Fréchet distance revisited and extended. *ACM Trans. Algo.*, 10(1): 3:1–3:22, 2014.

[Ind02] P. Indyk. Approximate nearest neighbor algorithms for Fréchet distance via product metrics. *Proc. 18th Annu. Sympos. Comput. Geom.* (SoCG), 102–106, 2002.

[KHM+98] S. Kwong, Q. H. He, K. F. Man, K. S. Tang, and C. W. Chau. Parallel genetic-based hybrid pattern matching algorithm for isolated word recognition. *Int. J. Pattern Recog. Art. Intel.*, 12(5): 573–594, 1998.

[KKS05] M. Kim, S. Kim, and M. Shin. Optimization of subsequence matching under time warping in time-series databases. *Proc. ACM symp. Applied comput.,* 581–586, 2005.

[KP99] E. J. Keogh and M. J. Pazzani. Scaling up dynamic time warping to massive dataset. *Proc. of the Third Euro. Conf. Princip. Data Mining and Know. Disc.,* 1–11, 1999.

[MP99] M. E. Munich and P. Perona. Continuous dynamic time warping for translation-invariant curve alignment with applications to signature verification. *Proc. 7th Int. Conf. Comp. Vision,* 108–115, 1999.

[MSSZ11] A. Maheshwari, J.-R. Sack, K. Shabaz, and H. Zarrabi-Zadeh. Improved algorithms for partial curve matching. *Proc. 9th Annu. Euro. Sympos. Alg.* (ESA), 518–529, 2011.

[SKB07] E. Sriraghavendra, K. Karthik, and C. Bhattacharyya. Fréchet distance based approach for searching online handwritten documents. *Proc. 9th Int. Conf. Doc. Analy. Recog.,* 461–465, 2007.

[WSP06] C. Wenk, R. Salas, and D. Pfoser. Addressing the need for map-matching speed: localizing global curve-matching algorithms. *Proc. 18th Int. Conf. Sci. Statis. Database Manag.,* 879–888, 2006.