THE FRÉCHET DISTANCE REVISITED AND EXTENDED

BY

BENJAMIN ADAM RAICHEL

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2011

Urbana, Illinois

Adviser:

Associate Professor Sariel Har-Peled

Abstract

Given two simplicial complexes, and start and end vertices in each complex, we show how to compute curves (in each complex) between these vertices, such that the Fréchet distance between these curves is minimized. As a polygonal curve is a complex, this generalizes the regular notion of Fréchet distance between curves. We also generalize the algorithm to handle an input of k simplicial complexes.

Using this new algorithm we can solve a slew of new problems, from computing a median curve for a given collection of curves, to various motion planning problems. Additionally, we show that for the median curve problem, when the k input curves are c-packed, one can $(1+\varepsilon)$-approximate the median curve in near linear time, for fixed k and ε.

To my parents, Donna and Ted.

Acknowledgements

This thesis is dedicated to my mother and father, Donna Melkonian and Ted Raichel, who have always encouraged me to ardently pursue my interests, whatever they may be. Additionally, this thesis is dedicated to my girlfriend, Aubrey Merrill, and to all my friends and family who have supported me over the years.

The author would also like to sincerely thank his adviser, Sariel Har-Peled, without who this thesis, and indeed my entire graduate path, might never have occurred.

Table of Contents

Chapter 1 Introduction **1**

Chapter 2 Preliminaries **4**
2.1 Curves and the Fréchet Distance 4
2.2 Complexes 4
2.3 Product Spaces 5

Chapter 3 Computing Optimal Fréchet Paths in Complexes **7**
3.1 Proof of Convexity 7
3.2 Algorithm 8
3.3 Analysis 9
3.3.1 Correctness 9
3.3.2 Running Time Analysis 10
3.3.3 Applications 11
3.4 Generalized Algorithm for k Complexes 12

Chapter 4 Applications **13**
4.1 Median Curve 13
4.2 Walking a Pack of Dogs 14
4.3 More General Settings 15
4.3.1 Minimizing Perimeter of Motion 15

Chapter 5 Computing the Median Curve for c-packed Curves **17**
5.1 Preliminaries 17
5.1.1 Definitions and Lemmas 17
5.1.2 Subroutines 18
5.2 Algorithm 19
5.3 Correctness and Running Time 20
5.3.1 Correctness 20
5.3.2 Running Time 20
5.3.3 Free Space Complexity 21
5.4 The Result 22

Chapter 6 Computing Optimal Fréchet Paths for DAG Complexes 23
6.1 The Decision Procedure 23
6.2 Using the Decision Procedure 25
6.2.1 Computing the Critical Values in an Interval 25
6.2.2 Sampling Critical Values 27
6.3 Analysis 27

Chapter 7 Conclusions 29

References 30

Chapter 1

Introduction

The Fréchet distance provides a way to measure the similarity between curves. Unlike the Hausdorff distance, which treats the curves as sets, the Fréchet distance takes into account the structure of the curves, by requiring continuous reparameterizations of the curves. Informally, the Fréchet distance between two curves, π and σ, is the minimum length leash needed to walk a dog when the person walks along π and the dog walks along σ.

In this thesis, we are interested in extending this concept to facilitate solving more general motion planning problems.

Previous Work. 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 [BBG08a, BBG+08b].

Alt and Godau [AG95] showed how to compute the Fréchet distance between two polygonal curves in $\mathbb{R}^d$, of total complexity n, in $O(n^2 \log n)$ time. It is an open problem to find a subquadratic algorithm for computing the Fréchet distance for two curves. The decision problem (i.e., deciding whether the Fréchet distance is smaller than a given value) has a lower bound of $\Omega(n \log n)$ [BBK+07]. Driemel *et al.* [DHW10] provided a $(1+\varepsilon)$-approximation for polygonal curves, that works in $O(N(\varepsilon, \pi, \sigma) + N(1, \pi, \sigma) \log n)$ time, where $N(\varepsilon, \pi, \sigma)$ is the relative free space complexity of two curves under simplification. In particular, their algorithm runs in $O(cn/\varepsilon + cn \log n)$ time for c-packed curves.

The notion of the Fréchet distance can also be generalized to encompass distances between surfaces. Unfortunately, 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].

Moving away from Fréchet distances between surfaces, Alt *et al.* [AERW03] presented an $O(n^2 \log^2 n)$ time algorithm to compute the Fréchet distance between two graphs. Specifically, they require that one of the two graphs has to be entirely traversed and in the other graph we seek the path that minimizes the Fréchet distance to the path of this traversal.

Complexes. The notion of a ***complex*** (which is an abstract simplicial complex together with its realization), defined formally in Section 2.2, is a generalization of polygonal curves, triangulations, meshes, straight line graphs, etc. In particular, our algorithm uses complexes as inputs and as such would apply for all these different inputs in a verbatim fashion.

Our Contribution. Given two complexes and start and end points in each one of them, we present a general algorithm that computes the two curves in these complexes that are closest to each other, under the Fréchet distance, and connect the corresponding start and end points. The expected running time of this new algorithm is $O(n^2)$. Our algorithm can be interpreted as an extension of the algorithm of Alt and Godau [AG95] for computing the weak Fréchet distance between polygonal curves. Our main contribution is the usage of the product complex instead of the parametric space – this enables us to easily encode the, potentially very complicated, connectivity information of the two input complexes in a simple way.

As concrete applications of our algorithm consider the following variants, all of them immediately solvable by our algorithm:

(A) **Fréchet for paths with thickness.** Imagine the classical setting of the Fréchet distance where a person walks a dog, but both the dog and the person might walk on paths that have non-zero width. That is, the input is two simple polygons (i.e., "thickened" paths) and one needs to compute the two paths of minimum Fréchet distance between them that lie inside their respective polygons.

(B) In a similar vane, consider a wiring problem: You are given a three dimensional model (of say a car or an airplane) specified by its mesh, and you are given a rough suggested path connecting two points in the mesh. Our algorithm can compute the optimal wiring path inside the model that is closest, under the Fréchet distance, to the suggested rough path.

Interestingly, this approach also extends to inputs of more than two complexes, and also to arbitrary convex functions between these different complexes. Specifically, consider a situation where the input includes k complexes $\mathcal{C}_1, \ldots, \mathcal{C}_k$. The reader might think about the complex $\mathcal{C}_i$ as the domain of the ith agent. Given a location in each of these complexes of their respective agent (i.e., a point p_i inside the complex $\mathcal{C}_i$ and the simplex $\Delta_i \subseteq \mathcal{C}_i$ that contains it) consider a scoring function $f(\mathsf{p}_1, \ldots, \mathsf{p}_k)$ that assigns a cost to the configuration $(\mathsf{p}_1, \ldots, \mathsf{p}_k)$. Furthermore, assume that this scoring function is convex on the domain $\Delta_1 \times \Delta_2 \times \cdots \times \Delta_k$, and this holds for any combination of such simplices. Now, given that the agents want to move from some starting vertices $v_1, \ldots, v_k$ to ending vertices $v'_1, \ldots, v'_k$, the new algorithm can compute the synced motion of these k agents from the starting configuration to the ending configuration, such that the maximum cost of any configuration used throughout the motion is minimized.

The reader might consider these settings a bit abstract, so here are a few examples of problems that can be solved using this framework:

(P1) **Median curve.** Given a set of k curves in $\mathbb{R}^d$, find a new curve that minimizes the maximum Fréchet distance between this new curve and each of the input curves. Namely, this computes a median curve for a given collection of curves.

(P2) One can compute the optimal way to walk k agents on k curves/complexes such that the maximum distance between any pair of agents, at any point in time, is minimized.

(P3) Compute the optimal way for the k agents to walk on the k curves/complexes, such that the maximum average distance between any pair of agents is minimized (the average is over all pairs).

(P4) Walk a pack of dogs while minimizing a weighted sum of the leash lengths (i.e. maybe some dogs need to be kept close since they like to chase squirrels).

(P5) Motion minimizing the perimeter of the convex hull. Given k curves/complexes that k agents have to move on (in the plane), compute a motion from the start points to the end points, such that the maximum perimeter of the convex hull is minimized throughout the motion.

The expected running time of all these algorithms for k input complexes of total complexity n is $O(n^k)$. The new algorithm/framework is quite general and should be applicable, in a plug and play fashion, to many other problems.

As a side problem, we also consider the problem when the input is two DAG complexes, which are directed acyclic straight line graphs embedded in $\mathbb{R}^d$. By considering the product space of two such complexes (instead of the parametric space) we show that the decision problem can be solved in $O(n^2)$ time. We then present a simple randomized technique to solve the general problem in $O(n^2 \log n)$ time. In particular, this provides an alternative algorithm that computes the (strong) Fréchet distance between two polygonal curves without using parametric search. Specifically, this algorithm is considerably simpler than the algorithm of Alt and Godau [AG95], while matching its running time. Previous efforts to avoid the parametric search by using randomization resulted in algorithms that are slower by a logarithmic factor [vOV04, CW09]. This new algorithm uses ideas applied for the problem of slope selection [Mat91] to the computation of the Fréchet distance. See Theorem 6.3.2 for details.

Organization. In Chapter 2, we define the Fréchet distance and complexes formally, as well as introduce the key concept of the freely space. Chapter 3 outlines the main algorithm of the paper, where it is shown that by applying the convexity property of the freely space, our problem can be converted into the problem of computing minimum spanning trees. We also generalize the algorithm to handle k input complexes, as well as arbitrary convex functions. In Chapter 4 we outline some applications of the main algorithm. In Chapter 5 we show that when the k input curves are c-packed, one can solve the median curve problem without the exponential dependence on k that the general algorithm has. In Chapter 6, we present an algorithm for computing the monotone Fréchet distance between two curves or between two DAG complexes.

Chapter 2

Preliminaries

2.1 Curves and the Fréchet Distance

Let $\pi \subseteq \mathbb{R}^d$ be a curve; that is, a continuous mapping from $[0,1]$ to $\mathbb{R}^d$. In the following, we will identify π with its range $\pi([0,1]) \subseteq \mathbb{R}^d$ if it is clear from the context.

A ***reparameterization*** is a continuous one-to-one function $f : [0,1] \to [0,1]$, such that $f(0) = 0$ and $f(1) = 1$. Given two reparameterizations f and g for two curves π and σ, respectively, define their ***width*** as

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

This can be interpreted as the maximum length of a leash one needs to walk a dog, where the dog walks along π according to f, while the handler walks along σ according to g. In this analogy, the Fréchet distance is the shortest possible leash admitting such a walk. Formally, given two curves π and σ in $\mathbb{R}^d$, the ***monotone Fréchet distance*** between them is

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

where f and g are orientation-preserving reparameterizations of the curves π and σ, respectively. In some cases, we will be interested in the ***weak Fréchet distance***, where the reparameterizations are required to be continuous but not necessarily bijections (i.e., one is allowed to walk backwards on their respective curve).

2.2 Complexes

An n-dimensional ***simplex*** is the convex hull of n+1 affinely independent vertices. We call the convex hull of any m+1 vertex subset of the vertices of a simplex, an m-dimensional ***subcell*** (or face) of that simplex (note that a subcell is in fact an m-dimensional simplex). A proper subcell is one such that $m < n$.

An ***abstract simplicial complex*** $\mathcal{C}_1 = (P, \mathcal{F})$, is a set system. The elements of P are ***points*** and the elements of $\mathcal{F}$ are subsets of P called ***simplices***. An abstract simplicial complex is downward closed; that is for any $\Psi \in \mathcal{F}$, and $\Upsilon \subseteq \Psi$, it holds that $\Upsilon \in \mathcal{F}$. For our purposes, the

ground set P will always be a subset of $\mathbb{R}^d$. We also use the natural ***realization*** of the abstract simplicial complex $(P, \mathcal{F})$, by mapping any simplex $\Psi \in \mathcal{F}$ to $\text{rel}(\Psi) = \mathcal{CH}(\Psi)$, where $\mathcal{CH}(\Psi)$ denotes the convex hull of Ψ. Throughout our discussion we assume that for any $\Psi \in \mathcal{F}$, we have $|\Psi| = \dim(\mathcal{CH}(\Psi)) + 1$ (i.e. Ψ is affinely independent). We also require that our realization is locally consistent; that is $\forall \Psi, \Upsilon \in \mathcal{F}$, if $\Psi \cap \Upsilon \neq \emptyset$ then $\text{rel}(\Psi) \cap \text{rel}(\Upsilon) = \text{rel}(\Psi \cap \Upsilon)$.

Note, that the geometric realization of such an abstract simplicial complex does not induce a simplicial complex. For example, such an abstract simplicial complex might define a self intersecting polygonal curve, where two disjoint simplices Ψ and Υ have that $\text{rel}(\Psi)$ and $\text{rel}(\Upsilon)$ intersect in their interior. In the following, we will refer to an abstract simplicial complex together with its realization as a ***complex***.

For a complex, $\mathcal{C}_1$, we will refer to any simplex in $\mathcal{C}_1$ as a ***cell*** of $\mathcal{C}_1$. The dimension of a complex is the maximum dimension of any of its cells. We say $\Psi \in \mathcal{C}_1$ is a maximal cell of $\mathcal{C}_1$ if there is no $\Upsilon \in \mathcal{C}_1$ such that $\Psi \subset \Upsilon$ (note that a maximum cell is one such that $dim(\Psi) = dim(\mathcal{C}_1)$).

A pair of simplices Ψ, Υ are ***adjacent*** if $\Psi \subseteq \Upsilon$ or $\Upsilon \subseteq \Psi$. A ***simplicial path*** in a complex is a function $f : [0, 1] \to \mathcal{F}$, such that: (A) For any $\Delta \in \mathcal{F}$, we have that $f^{-1}(\Delta)$ is a finite union of open intervals and points. (B) If $f(\cdot)$ has only two distinct values (say Δ and Ψ) on an interval $[x, y] \subseteq [0, 1]$, then the simplices Δ and Ψ are adjacent.

A curve $\pi \subseteq \mathbb{R}^d$ parameterized over $[0, 1]$ is a realization of a simplicial path f, if for any $t \in [0, 1]$ we have that $\pi(t) \in \text{rel}(f(t))$ and $f(t)$ is the simplex of lowest dimension of $\mathcal{F}$ that contains $\pi(t)$ (hence not all simplicial paths have realizations). In our applications, a maximal interval (x, y) such that f is constant corresponds to a straight segment of π. In particular, when dealing with a curve $\pi \subseteq \mathbb{R}^d$, we will assume that its associated simplicial path is also known.

In the following we will abuse notation and refer to Ψ as a shorthand for $\text{rel}(\Psi)$. In particular, for a point $p \in \mathbb{R}^d$, we will say that p is in the simplex Ψ if $p \in \text{rel}(\Psi)$.

2.3 Product Spaces

Let $\mathcal{C}_1 = (\mathsf{P}_1, \mathcal{F}_1)$ and $\mathcal{C}_2 = (\mathsf{P}_2, \mathcal{F}_2)$ be two simplicial complexes in $\mathbb{R}^d$. Consider the product space $\mathcal{C}_1 \times \mathcal{C}_2$. Intuitively, we view the product space as a subset of the space $\mathbb{R}^{2d}$, where the first d coordinates are from $\mathcal{C}_1$ and the remaining d coordinates are from $\mathcal{C}_2$. With this view, $\mathcal{C}_1 \times \mathcal{C}_2$ is similar to a simplicial complex although the cells will be convex polyhedra instead of just simplices (in the literature this is known as a **CW-*complex***). We define a cell (Ψ, Υ) of $\mathcal{C}_1 \times \mathcal{C}_2$ to be the product of any cell Ψ from $\mathcal{C}_1$ with any cell Υ from $\mathcal{C}_2$. Its realization is the set $\text{rel}(\Psi, \Upsilon) = \text{rel}(\Psi) \times \text{rel}(\Upsilon)$. In the CW-complex $\mathcal{C}_1 \times \mathcal{C}_2$, two cells (Ψ, Υ) and (Ψ', Υ') are ***adjacent*** if Ψ is adjacent to Ψ' in $\mathcal{C}_1$ and $\Upsilon = \Upsilon'$, or $\Psi = \Psi'$ and Υ is adjacent to Υ' in $\mathcal{C}_2$. Also, note that $\mathcal{C}_1 \times \mathcal{C}_2$ is connected since, by assumption, the complexes $\mathcal{C}_1$ and $\mathcal{C}_2$ are connected.

Let π and σ be curves with reparameterizations f and g, respectively. Let $\text{cell}_\pi(\cdot)$ and $\text{cell}_\sigma(\cdot)$ be the simplicial paths associated with f and g, respectively. Since the Cartesian product of two continuous functions is continuous, we have that $h(t) = (\pi(f(t)), \sigma(g(t)))$ defines a curve $\tau = \cup_t h(t)$

in $\mathcal{C}_1 \times \mathcal{C}_2$, which we call the ***product curve*** of $\pi(f)$ and $\sigma(g)$. The curve τ has a corresponding ***product cell path*** which is the function $\mathrm{cell}_{\pi,\sigma}(t) = (\mathrm{cell}_\pi(t), \mathrm{cell}_\sigma(t))$. (For the sake of simplicity of exposition, we are assuming here that $\mathrm{cell}_\pi(t)$ and $\mathrm{cell}_\sigma(t)$ do not change their value simultaneously at the same time t.)

For two complexes $\mathcal{C}_1$ and $\mathcal{C}_2$ in $\mathbb{R}^d$, and a parameter $\delta \geq 0$, consider a cell (Δ_1, Δ_2) in $\mathcal{C}_1 \times \mathcal{C}_2$. For a point $\mathsf{p} = (\mathsf{p}_1, \mathsf{p}_2) \in (\Delta_1, \Delta_2)$, its ***elevation*** is the quantity $\mathrm{elev}(\mathsf{p}) = \mathrm{elev}(\mathsf{p}_1, \mathsf{p}_2) = \|\mathsf{p}_1 - \mathsf{p}_2\|$.

The feasible region in the cell $\Delta_1 \times \Delta_2$ is the set

$$F_{\leq\delta}(\Delta_1, \Delta_2) = \left\{ (x, y) \in \mathbb{R}^{2d} \;\middle|\; \begin{array}{l} x \in \mathrm{rel}(\Delta_1) \subseteq \mathbb{R}^d \\ y \in \mathrm{rel}(\Delta_2) \subseteq \mathbb{R}^d \\ \mathrm{elev}(\mathsf{p}_1, \mathsf{p}_2) \leq \delta \end{array} \right\};$$

The feasible region for $\mathcal{C}_1 \times \mathcal{C}_2$ (which we will refer to as the ***freely space***[1]) is the set $F_{\leq\delta}(\mathcal{C}_1, \mathcal{C}_2) = \cup_{\Delta_1 \in \mathcal{C}_1, \Delta_2 \in \mathcal{C}_2} F_{\leq\delta}(\Delta_1, \Delta_2)$.

Observation 2.3.1 *Let π and σ be paths in $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively, and let f and g be reparameterizations of π and σ respectively, that realize the value δ of the Fréchet distance. The product curve, τ, is contained in $F_{\leq\delta}(\mathcal{C}_1, \mathcal{C}_2)$. Indeed, for any $t \in [0, 1]$, we have $\mathrm{elev}(\pi(f(t)), \sigma(g(t))) \leq \delta$, since f and g realize the Fréchet distance between π and σ.*

Observation 2.3.2 *Consider a curve σ in $\mathcal{C}_1 \times \mathcal{C}_2$, such that for any point $\mathsf{p} \in \sigma$ we have that $\mathrm{elev}(\mathsf{p}) \leq \delta$. Then, the projection of this curve into the corresponding curves in $\mathcal{C}_1$ and $\mathcal{C}_2$ results in two curves σ_1 and σ_2 such that $\mathsf{d}_{\mathcal{F}}(\sigma_1, \sigma_2) \leq \delta$.*

Formally, for $t \in [0, 1]$, let $\sigma(t) = (\sigma_1(t), \sigma_2(t)) \in \mathcal{C}_1 \times \mathcal{C}_2$ be a parameterization of σ, and let $\mathrm{cell}_\sigma(t) = (\mathrm{cell}_{\sigma_1}(t), \mathrm{cell}_{\sigma_2}(t))$ be its associated product cell path, such that for any t we have $\sigma(t) \in \mathrm{rel}(\mathrm{cell}_\sigma(t))$. Clearly, $\sigma_1(t)$ and $\sigma_2(t)$ are parameterized curves in the complexes $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively. Furthermore, for any $t \in [0, 1]$, we have that $\|\sigma_1(t) - \sigma_2(t)\| = \mathrm{elev}(\sigma(t)) \leq \delta$. As such, $\mathsf{d}_{\mathcal{F}}(\sigma_1, \sigma_2) \leq \delta$.

[1] An homage to free space.

Chapter 3

Computing Optimal Fréchet Paths in Complexes

We are given as input two complexes, $\mathcal{C}_1$ and $\mathcal{C}_2$, along with corresponding start and end vertices s_1, t_1 and s_2, t_2. We wish to compute the paths π and σ in $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively, that minimize the Fréchet distance over all paths that start and end at the respective start and end vertices.

3.1 Proof of Convexity

To this end, we need the following result which is standard by now, which we prove for the sake of completeness.

Lemma 3.1.1 *Let $F_{\leq\delta} = F_{\leq\delta}(\mathcal{C}_1, \mathcal{C}_2)$ be the freely space of the complexes $\mathcal{C}_1$ and $\mathcal{C}_2$, both contained in $\mathbb{R}^d$. Then $F_{\leq\delta}(\Psi, \Upsilon) = F_{\leq\delta} \cap (\mathrm{rel}(\Psi) \times \mathrm{rel}(\Upsilon))$ is a convex set, for any cell (Ψ, Υ) of $\mathcal{C}_1 \times \mathcal{C}_2$.*

Putting it differently, the elevation function $\mathrm{elev}(\cdot)$ *is convex over* $\mathrm{rel}(\Psi) \times \mathrm{rel}(\Upsilon)$*, for any cell (Ψ, Υ) of $\mathcal{C}_1 \times \mathcal{C}_2$.*

Proof: Let Ψ and Υ be simplices in $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively, and let $F = F_{\leq\delta}(\Psi, \Upsilon)$. By the definition of freely space, we know that F is just the sublevel set (i.e. the level set and everything less than that level) of the function $h : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, where $h(u, v) = \|u - v\|$, when applied to $\Psi \times \Upsilon$. It is known that the sublevel set of a convex function with a convex domain, is convex. Hence all we need to show is that h is convex (note that the domain is convex since Ψ and Υ are convex).

So let $u, u' \in \Psi$ and $v, v' \in \Upsilon$. We show that $(t)h(u, v)+(1-t)h(u'v') \geq h\big(t(u, v) + (1 - t)(u', v')\big)$, for $t \in [0, 1]$. Equivalently, we show that the function $\widehat{h}(t) = h\big(t(u, v) + (1 - t)(u', v')\big)$ is convex on the interval $[0, 1]$, i.e. $\widehat{h}(t) \leq (1 - t)\widehat{h}(0) + (t)\widehat{h}(1)$ (actually we need to prove such an inequality for all choices of $u, u' \in \Psi$ and $v, v' \in \Upsilon$, which we will indeed prove since they were chosen arbitrarily).

Expanding out this function we get,

$$\begin{aligned}\widehat{h}(t) &= h\big(t(u, v) + (1 - t)(u', v')\big) = h\big(u' + t(u - u'), v' + t(v - v')\big) \\ &= \big\|u' + t(u - u') - v' - t(v - v')\big\| = \big\|(u' - v') + t(u + v' - u' - v)\big\|.\end{aligned}$$

Hence $\widehat{h}(t)$ is just the equation for the distance between a point on a linearly parameterized line and the origin. We have by Lemma 3.1.2 that this function is convex and so we are done. ■

Lemma 3.1.2 *The function representing the distance between a point on a linearly parameterized line $l(t)$ and the origin, is a convex function. Specifically, let a and b be vectors in $\mathbb{R}^d$, then the function $f(t) = \|a + tb\|$, is convex.*

Proof: We know that $f(t)$ is of the form,

$$f(t) = \sqrt{\sum_i (a_i + tb_i)^2} = \sqrt{\alpha t^2 + \beta t + \gamma},$$

where α, β, and γ are some constants such that $\alpha t^2 + \beta t + \gamma$ is non-negative. By the helper lemma below, however, we know such a function is convex. ■

Lemma 3.1.3 *Consider the quadratic function $\alpha t^2 + \beta t + \gamma$, where α, β and γ are some constants such that the function is non-negative. Then, the function $f(t) = \sqrt{\alpha t^2 + \beta t + \gamma}$ is convex.*

Proof: Since $\alpha t^2 + \beta t + \gamma \geq 0$ for all t, it must be that $\alpha > 0$, and the corresponding quadratic formula either has no roots, or a single root, which implies that $\beta^2 - 4\alpha\gamma \leq 0$. Now,

$$f'(t) = \frac{2\alpha t + \beta}{2\sqrt{\alpha t^2 + \beta t + \gamma}} = \frac{h(t)}{f(t)},$$

for $h(t) = \alpha t + \beta/2$. Similarly,

$$f''(t) = \frac{f(t)h'(t) - f'(t)h(t)}{(f(t))^2} = \frac{\alpha f(t) - (h(t))^2/f(t)}{(f(t))^2} = \frac{(f(t))^2 - (h(t))^2/\alpha}{(f(t))^3/\alpha}.$$

Now, since $f(t)$ is always non-negative, we have that

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

since $\alpha > 0$ and $\beta^2 - 4\alpha\gamma \leq 0$. ■

3.2 Algorithm

We construct a graph $\mathsf{G} = (V, E)$, called the ***cell graph*** of $\mathcal{C}_1 \times \mathcal{C}_2$. Specifically, each cell (Δ_1, Δ_2) of $\mathcal{C}_1 \times \mathcal{C}_2$ corresponds to a vertex $v_{(\Delta_1,\Delta_2)} \in V$, and for every pair $v_{(\Delta_1,\Delta_2)}, v_{(\Delta_1',\Delta_2')} \in V$ we create an edge iff (Δ_1, Δ_2) and (Δ_1', Δ_2') are adjacent in $\mathcal{C}_1 \times \mathcal{C}_2$. For $\Delta_1 \in \mathcal{C}_1$ and $\Delta_2 \in \mathcal{C}_2$, the ***elevation*** of their corresponding vertex $v = v_{(\Delta_1,\Delta_2)} \in V$ is $\operatorname{elev}(v) = \mathsf{d}(\Delta_1, \Delta_2)$, where $\mathsf{d}(\Delta_1, \Delta_2) = \min_{\mathsf{p}\in\operatorname{rel}(\Delta_1),\mathsf{q}\in\operatorname{rel}(\Delta_2)} \operatorname{elev}(\mathsf{p}, \mathsf{q}) = \min_{\mathsf{p}\in\operatorname{rel}(\Delta_1),\mathsf{q}\in\operatorname{rel}(\Delta_2)} \|\mathsf{p} - \mathsf{q}\|$ is the distance between these simplices. The point realizing this minimum is the ***realization*** of the vertex v, and is denoted by $\operatorname{rel}(v)$.

A path connecting two vertices u and v of G is a *uv **path***. The cell graph is clearly connected since $\mathcal{C}_1 \times \mathcal{C}_2$ is connected. As such, for any pair of vertices $u, v \in V(\mathsf{G})$ there exists a uv path in G. The ***elevation*** of a path ρ, denoted by $\operatorname{elev}(\rho)$, is the maximum elevation of any vertex in ρ. The ***lowest*** uv path in G is the uv path with minimum elevation.

We compute the lowest st path in G (where $s = (s_1, s_2)$ and $t = (t_1, t_2)$), in order to determine the desired curves with minimum Fréchet distance. To this end, we set the elevation of any edge $uv \in E(\mathsf{G})$ to be $\operatorname{elev}(uv) = \max(\operatorname{elev}(u), \operatorname{elev}(v))$, and we compute the MST (minimum spanning tree) T of G under this weight function. Then, we compute the unique path between s and t in T, and let $\rho = v^1 \dots v^m$ be the resulting path. We return the polygonal path $\operatorname{rel}(v^1) \operatorname{rel}(v^2) \cdots \operatorname{rel}(v^m) \subseteq \mathcal{C}_1 \times \mathcal{C}_2$ as the desired curve (which by Observation 2.3.2 encodes the two desired curves and their reparameterizations).

3.3 Analysis

3.3.1 Correctness

The following easy lemma shows that the MST indeed contains our desired path.

Lemma 3.3.1 *Let G be a graph with non-negative weights on its edges. For any two vertices $u, v \in V(\mathsf{G})$, for the unique path τ between u and v in the MST, we have that $\operatorname{elev}(\tau) \leq \operatorname{elev}(\sigma)$, where σ is any uv path in G, and $\operatorname{elev}(\tau)$ is the maximum weight edge along the path τ.*

Proof: Consider a uv path σ in G. If σ is contained in the MST then we are done. Otherwise, let e be any edge of σ that is not contained in the MST. Introducing the edge e into the MST creates a cycle, where all the other edges on the cycle are lighter than e (otherwise, e must be in the MST). As such, we can replace e in σ by the portion of this cycle connecting its endpoints. This new path σ' has one less edge outside the MST, and it holds that $\operatorname{elev}(\sigma') \leq \operatorname{elev}(\sigma)$. Continuing in this fashion, we end up with a path τ' in the MST between u and v such that $\operatorname{elev}(\tau') \leq \operatorname{elev}(\sigma)$. Since the path in the MST between u and v is unique, the claim now follows. ■

As the following lemmas show, the cell graph captures the relevant information for our problem.

Lemma 3.3.2 *Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two complexes, and let s_1 and t_1 be vertices of $\mathcal{C}_1$ and let s_2 and t_2 be vertices of $\mathcal{C}_2$. Then, if there exists an s_1t_1 path π, in $\mathcal{C}_1$, and an s_2t_2 path σ, in $\mathcal{C}_2$, such that $\mathsf{d}_{\mathcal{F}}(\pi, \sigma) = \delta$ then there exists a $v_{(s_1,s_2)}v_{(t_1,t_2)}$ path, ρ, in $\mathsf{G}(\mathcal{C}_1, \mathcal{C}_2)$ such that $\operatorname{elev}(\rho) \leq \delta$.*

Proof: Let f and g be the reparameterizations of π and σ, respectively, that achieve the value δ for the Fréchet distance. By Observation 2.3.1 the product curve $\tau = \bigcup_t \big(\pi(f(t)), \sigma(g(t))\big)$, defines a path in $\mathcal{C}_1 \times \mathcal{C}_2$ from (s_1, s_2) to (t_1, t_2) that is contained in the freely space $F_{\leq\delta}(\mathcal{C}_1, \mathcal{C}_2)$. Let $\operatorname{cell}_{\pi,\sigma}(t)$ be the product cell path in $\mathcal{C}_1 \times \mathcal{C}_2$ that corresponds to $\tau(t)$. Naturally, the value of $\operatorname{cell}_{\pi,\sigma}(t)$ corresponds to a vertex in G, and let $v(t)$ denote this vertex. It is easy to verify that the

sequence of different vertices visited by $v(t)$, as t increases from 0 to 1, is a valid path in G. Indeed, a product cell path defines a sequence of adjacent cells of $\mathcal{C}_1 \times \mathcal{C}_2$ as t increases from 0 to 1, which corresponds to a path $\rho = v^1, \ldots, v^m$ in G.

Observe, that for any $t \in [0,1]$, we have that

$$\operatorname{elev}(v(t)) = \operatorname{elev}\left(v_{\operatorname{cell}_{\pi,\sigma}(t)}\right) = \min_{\substack{\mathsf{p}\in\operatorname{cell}_\pi(t),\\ \mathsf{q}\in\operatorname{cell}_\sigma(t)}} \|\mathsf{p}-\mathsf{q}\| \leq \|\pi(f(t)) - \sigma(g(t))\| \leq \delta.$$

As such, $\operatorname{elev}(\rho) = \max_i \operatorname{elev}\left(v^i\right) = \max_t \operatorname{elev}(v(t)) \leq \delta$. ■

Lemma 3.3.3 *Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two complexes, and let s_1 and t_1 be vertices of $\mathcal{C}_1$ and let s_2 and t_2 be vertices of $\mathcal{C}_2$. Then, if there exists a $v_{(s_1,s_2)}v_{(t_1,t_2)}$ path ρ in $\mathsf{G}(\mathcal{C}_1,\mathcal{C}_2)$ such that $\operatorname{elev}(\rho) = \delta$ then there exists an s_1t_1 path, π, in $\mathcal{C}_1$ and an s_2t_2 path, σ, in $\mathcal{C}_2$, such that $\mathsf{d}_{\mathcal{F}}(\pi,\sigma) = \delta$.*

Proof: Let $\rho = v^1 \ldots v^m$, where $v^1 = v_{(s_1,s_2)}$ and $v^m = v_{(t_1,t_2)}$. Each vertex v^i in ρ corresponds to a pair of cells $\Delta^i = (\Delta_1^i, \Delta_2^i)$, where $\Delta_1^i \in \mathcal{C}_1$ and $\Delta_2^i \in \mathcal{C}_2$. Furthermore, for every i, there exists two points $\mathsf{p}_1^i \in \Delta_1^i$ and $\mathsf{p}_2^i \in \Delta_2^i$, such that $\operatorname{elev}\left(\mathsf{p}^i\right) = \left\|\mathsf{p}_1^i - \mathsf{p}_2^i\right\| = \mathsf{d}\left(\Delta_1^i, \Delta_2^i\right)$, where $\mathsf{p}^i = \left(\mathsf{p}_1^i, \mathsf{p}_2^i\right)$.

Observe, that for all the vertices of the polygonal path $Z = \mathsf{p}^1\mathsf{p}^2 \ldots \mathsf{p}^m$, we have that $\operatorname{elev}\left(\mathsf{p}^i\right) = \mathsf{d}\left(\Delta_1^i, \Delta_2^i\right) = \operatorname{elev}\left(v^i\right) \leq \operatorname{elev}(\rho) = \delta$. As such, all the vertices of Z are in the freely space $F_{\leq\delta}$.

For any i, the ith segment of Z is $\mathsf{p}^i\mathsf{p}^{i+1}$. It corresponds to the edge v^iv^{i+1} in the graph G, which connects adjacent cells in $\mathcal{C}_1 \times \mathcal{C}_2$. In particular, it must be that either $\Delta^i \subseteq \Delta^{i+1}$ or $\Delta^{i+1} \subseteq \Delta^i$. Assume the latter happens (the other case is handled in a symmetric fashion). We have that $\mathsf{p}^i\mathsf{p}^{i+1} \subseteq \Delta^i$. Furthermore, by the convexity of the freely space inside a single cell (i.e., Lemma 3.1.1), we have that $\mathsf{p}^i\mathsf{p}^{i+1} \subseteq \Delta^i \cap F_{\leq\delta}$. We conclude that $Z \subseteq F_{\leq\delta}$. Since the two endpoints of Z are $(s_1,s_2) = \mathsf{p}^1$ and $(t_1,t_2) = \mathsf{p}^m$, Z corresponds to the desired paths π and σ such that $\mathsf{d}_{\mathcal{F}}(\pi,\sigma) = \delta$. ■

Corollary 3.3.4 *Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two complexes, and let s_1 and t_1 be vertices of $\mathcal{C}_1$ and let s_2 and t_2 be vertices of $\mathcal{C}_2$. Moreover, let π and σ be the paths in $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively, that minimize the Fréchet distance over all pairs of s_1t_1 and s_2t_2 paths. Then we have that $\mathsf{d}_{\mathcal{F}}(\pi,\sigma) = \delta$ if and only if the lowest $v_{(s_1,s_2)}v_{(t_1,t_2)}$ path, ρ, in $\mathsf{G}(\mathcal{C}_1,\mathcal{C}_2)$ has $\operatorname{elev}(\rho) = \delta$.*

3.3.2 Running Time Analysis

Computing the MST takes linear time in expectation [MR95]. Since a vertex in the cell graph represents a pair of simplices from $\mathcal{C}_1$ and $\mathcal{C}_2$, we know that $|V(\mathsf{G})| = O(|\mathcal{C}_1||\mathcal{C}_2|)$. We also know that $|E(\mathsf{G})| = O(|V(\mathsf{G})|)$ since each cell in $\mathcal{C}_1 \times \mathcal{C}_2$ has at most $O(1)$ proper subcells (specifically $O\left(2^{2d}\right) = O(1)$). Hence the running time of the algorithm is $O\left(n^2\right)$, where $n = \max(|\mathcal{C}_1|, |\mathcal{C}_2|)$.

Putting everything together, we get the following result.

Theorem 3.3.5 *Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two simplicial complexes, and $n = \max(|\mathcal{C}_1|, |\mathcal{C}_2|)$. Given any pair of start and end vertices from $\mathcal{C}_1$ and any pair of start and end vertices from $\mathcal{C}_2$, we can*

compute, in expected $O(n^2)$ time, the paths π and σ in $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively, that minimize the Fréchet distance over all paths that start and end at the respective start and end vertices.

Remark 3.3.6 It is easy to verify that Theorem 3.3.5 yields a path that is locally as low as possible. Formally, if the solution in the CW complex is a curve π, then for any subcurve $\sigma \subseteq \pi$, we have the property that for any other curve τ, that has the same endpoints of σ, it holds that $\mathrm{elev}(\tau) \geq \mathrm{elev}(\sigma)$.

When computing the Fréchet distance for two curves for example, this property implies that the parameterization we get is never lazy – it always tries to be as tight as possible at any given point in time.

3.3.3 Applications

Fréchet for paths with thickness. Given two polygons (maybe with holes) in the plane and start and end vertices in the two polygons, one can triangulate the two polygons and then feed them into Theorem 3.3.5. This results in two paths in the two triangulations that minimize the Fréchet distance between the paths. As a concrete application, this can be used for solving the classical Fréchet distance problem where the input curves have thickness associated with them and one can move in this enlarged region. Indeed, each "thickened" curve can be represented as a polygon, and hence we can apply the above algorithm.

Wiring. The wiring problem, mentioned in the introduction, can be solved by immediate plug and play into the above result.

Motion planning in planar environments. Consider the case where you need to plan the motion of two entities in a two dimensional environment, where they have to stay close together (i.e., Fréchet distance) while complying with different constraints on which part of the environment they can travel on. As a concrete example, one entity might be a pedestrian and the other might be a vehicle. The pedestrian can not use the road, and the vehicle can not use the sidewalk or the parks available. Finding the best motion for the two entities is no more than solving the Fréchet problem in this setting. Indeed, we compute a triangulation of the environment for the first entity, and then remove all triangles and edges that can not be used by the first entity. Similarly, we compute a triangulation for the second entity, removing the regions that are unusable for it.

Now, applying the algorithm of Theorem 3.3.5 to these two triangulations (with the desired starting and ending points) results in the desired motion.

Naturally, the algorithm of Theorem 3.3.5 can be applied in more general settings where the input is three dimensional, etc.

3.4 Generalized Algorithm for k Complexes

Let us recap the algorithm from the previous section. We considered finding the path in the product space (of two complexes) such that the maximum value of $f(x, y) = \|x - y\|$ among all the points (x, y) in the path is minimized. If we add an extra dimension for the value of f, then one can think of f as defining a terrain. Then the problem becomes computing a path that does not traverse high in this terrain. The Freely space was the sublevel set of f for some parameter δ. Next, we defined the elevation of a vertex in the cell graph to be the minimum value of f for the cell that the vertex corresponds to. By observing that f was a convex function within each cell in the product space, we were able to argue that the value of the best path (i.e. lowest maximum value of f) was equivalent to the elevation of a path between the corresponding vertices in an MST of the cell graph, and thus the problem was efficiently solvable.

With this abstract description, the only property of f that we used was that it was convex within each cell in the product space. Hence, we can conclude that the same procedure will work for any choice of f, so long as it is convex within each cell in the product space.

We can generalize the problem even further. Earlier we considered only two complexes. However, there is no reason why we can not consider an input of k complexes, for some arbitrary integer k. In order to handle this case we generalize all our earlier definitions for two complexes in the following natural way.

Let $\mathcal{C}_1 = (\mathsf{P}_1, \mathcal{F}_1), \ldots, \mathcal{C}_k = (\mathsf{P}_k, \mathcal{F}_k)$ be a set of k simplicial complexes in $\mathbb{R}^d$. Consider the product space $\mathcal{C}_1 \times \cdots \times \mathcal{C}_k$. Intuitively, we view the product space as a subset of the space $\mathbb{R}^{kd}$. We define a cell $(\Delta_1, \ldots, \Delta_k)$ of $\mathcal{C} = \mathcal{C}_1 \times \cdots \times \mathcal{C}_k$ to be the product of k cells, where $\Delta_i \in \mathcal{C}_i$, for $i = 1, \ldots, k$. Its realization is the set $\mathrm{rel}(\Delta_1, \ldots, \Delta_k) = \mathrm{rel}(\Delta_1) \times \ldots \times \mathrm{rel}(\Delta_k)$. In $\mathcal{C}_1 \times \ldots \times \mathcal{C}_k$, two cells $(\Delta_1, \ldots, \Delta_k)$ and $(\Psi_1, \ldots, \Psi_k)$ are ***adjacent*** if there is a j such that for all $i \neq j$, $\Delta_i = \Psi_i$ and Δ_j is adjacent to Ψ_j in $\mathcal{C}_j$.

We now are given a function f defined over $\mathbb{R}^{kd}$ that is convex for any cell $\mathrm{rel}(\Delta_1, \ldots, \Delta_k)$. As before, we build the cell graph G of the CW complex $\mathcal{C}$. Every vertex v of G corresponds to a cell Δ of $\mathcal{C}$, and its elevation is the minimum value of f in this cell.

As before, we are given start vertices $s_1, \ldots, s_k$ and end vertices $t_1, \ldots, t_k$ in these k complexes. We compute the lowest elevation path between the vertex in G corresponding to $(s_1, \ldots, s_k)$ and the vertex in G corresponding to $(t_1, \ldots, t_k)$. Arguing as before, it is easy to show that the resulting path in the graph can be realized by a path in $\mathcal{C}$ that yields the k desired paths and their reparameterizations. As such, we get the following result.

Theorem 3.4.1 *We are given k simplicial complexes $\mathcal{C}_1, \ldots, \mathcal{C}_k$, $n = \max_i |\mathcal{C}_i|$, start vertices $s_1 \in \mathcal{C}_1, \ldots s_k \in \mathcal{C}_k$, end vertices $t_1 \in \mathcal{C}_1, \ldots, t_k \in \mathcal{C}_k$, and a function $f : \mathrm{rel}(\mathcal{C}) \to \mathbb{R}$ that is convex for any cell in the realization of $\mathcal{C} = \mathcal{C}_1 \times \cdots \times \mathcal{C}_k$.*

Then, one can compute, in expected $O(n^k)$ time, k curves $\pi_1, \ldots, \pi_k$ (and their reparameterizations $\psi_1, \ldots, \psi_k$) connecting $s_1, \ldots, s_k$ to $t_1, \ldots, t_k$, respectively, such that $\max_t f(\pi_1(\psi_1(t)), \ldots, \pi_k(\psi_k(t)))$ is minimized, among all such curves and reparameterizations.

Chapter 4

Applications

4.1 Median Curve

We are given k polygonal curves $\pi_1, \ldots, \pi_k$ in $\mathbb{R}^d$, and we would like to compute a curve σ that minimizes the maximum Fréchet distance between σ and each one of the curves $\pi_1, \ldots, \pi_k$.

For a set of points $\mathsf{P} \subseteq \mathbb{R}^d$, let $r_{\min}(\mathsf{P})$ denote the radius of the minimum enclosing ball of P.

Lemma 4.1.1 *Let $\mathsf{P}(t)$ be a set of points in $\mathbb{R}^d$ moving linearly with t. Then, the function $r_{\min}(t) = r_{\min}(\mathsf{P}(t))$ is convex.*

Proof: Fix any three times, $x < y < z$, where $y = \alpha x + (1-\alpha)z$ for some $\alpha \in (0,1)$. Let $\mathsf{p}_i(t)$ denote the ith moving point of $\mathsf{P}(t)$.

Let v_x (resp. v_z) be the center of the minimum enclosing ball of $\mathsf{P}(x)$ (resp. $\mathsf{P}(z)$), and let $\mathsf{v}(y) = \alpha \mathsf{v}_x + (1-\alpha)\mathsf{v}_z$. Observe that

$$\begin{aligned} r_{\min}(y) &= r_{\min}\big(\mathsf{P}(\alpha x + (1-\alpha)z)\big) \leq \max_i \|\mathsf{v}(\alpha x + (1-\alpha)z) - \mathsf{p}_i(\alpha x + (1-\alpha)z)\| \\ &\leq \max_i (\alpha \|\mathsf{v}_x - \mathsf{p}_i(x)\| + (1-\alpha)\|\mathsf{v}_z - \mathsf{p}_i(z)\|) \\ &= \alpha r_{\min}(x) + (1-\alpha) r_{\min}(z)\,, \end{aligned}$$

since the distance between a pair of linearly moving points is convex (for example by Lemma 3.1.2). ∎

Using the lemma above, we get the following desired result.

Lemma 4.1.2 *Given k curves $\pi_1, \ldots, \pi_k$ in $\mathbb{R}^d$ with total complexity n, one can compute, in $O\big(n^k\big)$ expected time, a curve σ that minimizes $\max_i \mathsf{d}^{\mathrm{w}}_{\mathcal{F}}(\pi_i, \sigma)$, where $\mathsf{d}^{\mathrm{w}}_{\mathcal{F}}(\pi_i, \sigma)$ is the weak Fréchet distance between π_i and σ.*

Proof: A cell in the CW complex of $\pi_1 \times \cdots \times \pi_k$ is the product of k segments (or points) in $\mathbb{R}^d$. For a point $\mathsf{p} = (\mathsf{p}_1, \ldots, \mathsf{p}_k) \in \mathbb{R}^{dk}$ inside such a cell, consider the elevation of p to be $f(\mathsf{p}) = r_{\min}(\{\mathsf{p}_1, \ldots, \mathsf{p}_k\})$. Lemma 4.1.1 implies that $f(\cdot)$ is convex inside each such cell. As such, applying Theorem 3.4.1 to the given curves, using the function $f(\cdot)$, results in a parameterization

that minimizes the maximum radius of the minimum enclosing ball throughout the motion. Since the center of the minimum enclosing ball (for continuously moving points) changes continuously over time, the curve formed by this center throughout the motion is a natural median curve. Let σ denote this curve. It is easy to prove that the maximum Fréchet distance of σ to any of the curves $\pi_1, \ldots, \pi_k$ is the minimum such value among all possible curves. ■

4.2 Walking a Pack of Dogs

So suppose you have a pitbull, a chiwawa, a corgi, and a terrier. You want to walk all the dogs at the same time instead of walking each one individually.[1] However, as before, long leashes are expensive, so you want to minimize the maximum length leash (among all the leashes) that you need to use.

Formally, you are given k complexes, $\mathcal{C}_1, \ldots, \mathcal{C}_k$, and start and end vertices $s_i, t_i \in \mathcal{C}_i$, for $i = 1, \ldots, k$. The first complex corresponds to the person leading the dogs, and the complexes $\mathcal{C}_2, \ldots, \mathcal{C}_k$ corresponds to the $k-1$ given dogs. You wish to find the set of paths, $\pi_1, \ldots, \pi_k$, and corresponding reparameterizations, $\psi_1, \psi_2, \ldots, \psi_k$, such that,

$$\max_{t\in[0,1]} \max_{i>1} \|\pi_1(\psi_1(t)) - \pi_i(\psi_i(t))\| = \max_{t\in[0,1]} f(\pi_1(\psi_1(t)), \ldots, \pi_k(\psi_k(t))),$$

is minimized, where $f(\mathsf{p}_1, \ldots, \mathsf{p}_k) = \max_i \|\mathsf{p}_1 - \mathsf{p}_i\|$.

Lemma 4.2.1 *Given k polygonal curves $\pi_1, \ldots, \pi_k$ of total complexity n, one can compute non-monotone reparameterizations of these curves such that $\max_t \max_i \|\pi_1(\psi_1(t)) - \pi_i(\psi_i(t))\|$ is minimized. The expected running time of the algorithm is $O\big(n^k\big)$.*

This works verbatim for complexes, and in this case the algorithm also computes the paths inside the complexes realizing the Fréchet distance.

Proof: We need to prove that the function $f(\mathsf{p}_1, \ldots, \mathsf{p}_k) = \max_i \|\mathsf{p}_1 - \mathsf{p}_i\|$ is convex within each cell in order to apply Theorem 3.4.1.

So, consider a cell $\Delta = (\Delta_1, \ldots, \Delta_k) \in \mathcal{C} = \mathcal{C}_1 \times \cdots \times \mathcal{C}_k$. Its realized cell $\mathrm{rel}(\Delta) = \mathrm{rel}(\Delta_1) \times \cdots \times \mathrm{rel}(\Delta_k)$ is a convex set. In particular, consider the functions of the form $f_i(\mathsf{p}_1, \mathsf{p}_i) = \|\mathsf{p}_1 - \mathsf{p}_i\|$, defined over $\mathrm{rel}(\Delta_1) \times \mathrm{rel}(\Delta_i)$, for $2 \le i \le k$. Each of these functions are convex by Lemma 3.1.1 on the domain $\mathrm{rel}(\Delta_1) \times \mathrm{rel}(\Delta_i)$. In particular, setting $g_i(\mathsf{p}_1, \ldots, \mathsf{p}_k) = f_i(\mathsf{p}_1, \mathsf{p}_i)$, for $i = 1, \ldots, k$, results in k convex functions over $\mathrm{rel}(\Delta)$.

Clearly, $f(\mathsf{p}_1, \ldots, \mathsf{p}_k) = \max_i g_i(\mathsf{p}_1, \ldots, \mathsf{p}_k)$, which is convex as the maximum of a set of convex functions is a convex function. As such, plugging this into Theorem 3.4.1 implies the result. ■

[1] Since clearly you are a person that is very concerned with efficiency.

4.3 More General Settings

From the previous example, consider the person and the dogs at any given time as vertices in space. The leashes are thus edges connecting the vertices. Hence in the above example the topology of the graph is that of star graphs (i.e. the person is at the center and the dogs are the ends of the star). The "weight" of each edge in the graph is the value of a convex function between the respective pair of vertices at a given instance of time (i.e. the distance of the person to a specific dog at a specific time). The general function we were trying to minimize was the maximum value over the functions between each pair of vertices. We were able to conclude that the overall function was convex because the maximum value of a set of convex functions, is a convex function.

Let the above described graph be called a ***dependency graph***. In general we can consider any topology for the dependency graph. More formally, between every pair of complexes we define a convex function (note that the zero function is convex, and so we can ignore certain pairs if we like). For our global function we can then take any function of these functions, which preserves convexity. For example, taking the maximum, the sum, or (positively) weighted sum of convex functions is again a convex function. Therefore, all of the applications (P1)–(P4) mentioned in the introduction are solvable immediately within this framework.

4.3.1 Minimizing Perimeter of Motion

We are given k complexes $\mathcal{C}_1, \ldots, \mathcal{C}_k$ all with realizations in the plane. As before, we are given k starting vertices $s_1, \ldots, s_k$ and k ending vertices $t_1, \ldots, t_k$, in these k complexes, respectively. We are interested in computing the k polygonal paths (and their reparameterizations) connecting these endpoints, such that the maximum perimeter is minimized. As before, to use the framework, we need to show that the perimeter function is convex inside a cell of the resulting CW complex. So, consider two points $\mathsf{p} = (\mathsf{p}_1, \ldots, \mathsf{p}_k)$ and $\mathsf{q} = (\mathsf{q}_1, \ldots, \mathsf{q}_k)$. We need to show that the perimeter function

$$\text{perim}(t) = \text{perim}(t\mathsf{p} + (1-t)\mathsf{q}) = \text{perimeter}\Big(\mathcal{CH}\big(\{t\mathsf{p}_1 + (1-t)\mathsf{q}_1, \ldots, t\mathsf{p}_k + (1-t)\mathsf{q}_k\}\big)\Big)$$

is convex. This fact, which we state below as a lemma, is proved in [AC10] using the Cauchy-Crofton inequality.

Lemma 4.3.1 ([AC10]) *The perimeter of a set of linearly moving points in the plane is a convex function.*

This implies that the perimeter function is convex inside each cell of $\mathcal{C} = \mathcal{C}_1 \times \cdots \times \mathcal{C}_k$, and hence the framework applies. We thus get the following result.

Lemma 4.3.2 *Given k complexes $\mathcal{C}_1, \ldots, \mathcal{C}_k$ all with realizations in the plane, k starting vertices $s_1, \ldots, s_k$ and k ending vertices $t_1, \ldots, t_k$, in these k complexes, respectively, then one can compute*

paths in these complexes, and their corresponding reparameterizations, such that the maximum perimeter of the moving points during this motion is minimized over all such motions. The expected running time of the algorithm is $O(n^k)$.

The running time stated above is under the assumption that computing the minimum perimeter for k points whose locations are restricted by a cell of the CW complex, can be done in constant time. This constant would depend on k, naturally.

Chapter 5

Computing the Median Curve for c-packed Curves

Driemel et al. [DHW10] introduced a realistic class of curves, called c-packed curves. We now show that when the k input curves are c-packed, one can compute a $(1+\varepsilon)$-approximation to the median curve in $\widetilde{O}(n \log n)$ time, where $\tilde{O}()$ is used to emphasize that the constant depends on ε and c, and exponentially on k and d (see Lemma 5.3.1 and Lemma 5.3.2 for details). This is a significant improvement over the algorithm for the general case, presented in Section 4.1, where the running time is $O(n^k)$.

In this section, when we refer to the free space, it is meant with respect to the median curve distance function. In particular, for k curves $\pi_1, \ldots, \pi_k$ let $\mathsf{d}_{med}(\pi_1, \ldots, \pi_k)$ denote the maximum distance of the median curve to the π_i's, for the optimum reparameterizations.

5.1 Preliminaries

5.1.1 Definitions and Lemmas

We first cover the definitions and lemmas from [DHW10] that are relevant to our problem.

Definition 5.1.1 For a parameter $c > 0$, 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 the portions of π inside the ball $\mathbf{b}(\mathsf{q}, r)$ is at most cr.

Algorithm 5.1.2 Given a polygonal curve $\pi = \mathsf{q}_1\mathsf{q}_2\mathsf{q}_3 \ldots \mathsf{q}_k$ 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 $\pi' = \mathrm{simpl}(\pi, \mu)$ as being the ***μ-simplification*** of π. Note, that this simplification can be computed in linear time.

We need the following useful facts about μ-simplifications from [DHW10].

Lemma 5.1.3 *(i) For any curve π in $\mathbb{R}^d$, and $\mu > 0$, we have that $\mathsf{d}_{\mathcal{F}}\big(\pi, \mathrm{simpl}(\pi, \mu)\big) \leq \mu$.*

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

Observation 5.1.4 *Let π and σ be two given curves, and let π' and σ' be their μ simplified curves, for some value μ. By Lemma 5.1.3, $\mathsf{d}_{\mathcal{F}}(\pi, \pi') \leq \mu$ and $\mathsf{d}_{\mathcal{F}}(\sigma, \sigma') \leq \mu$. Hence we have reparameterizations f and g such that $\|\pi(f(t)) - \pi'(t)\| \leq \mu$ and $\|\sigma(g(t)) - \sigma'(t)\| \leq \mu$ for all $t \in [0,1]$ (without loss of generality we can assume these reparameterizations are bijective). Let $\mathsf{d}^{\mathrm{w}}_{\mathcal{F}}(\pi', \sigma') = \delta$. Then we have that $\mathsf{d}^{\mathrm{w}}_{\mathcal{F}}(\pi, \sigma) \leq \delta + 2\mu$, since we can just map each pair $(x, y) \in (\pi', \sigma')$ that is seen in the optimal (not necessarily injective) reparameterizations of π' and σ', to the corresponding pair in (π, σ) determined by f and g. In particular, this implies that for curves $\pi_1, \ldots, \pi_k$ with corresponding μ simplifications $\pi'_1, \ldots, \pi'_k$, we have that $\mathsf{d}_{med}(\pi_1, \ldots, \pi_k) \leq \mathsf{d}_{med}(\pi'_1, \ldots, \pi'_k) + 2\mu$.*

Let $\pi_1, \ldots, \pi_k$ be k given curves. The ***complexity*** of the reachable free space for these curves, for a distance δ, denoted by $\mathcal{N}_{\leq\delta}(\pi_1, \ldots, \pi_k)$, is the total number of cells in the CW-complex with non-empty intersection with $F_{\leq\delta}(\pi_1, \ldots, \pi_k)$ such that there exists a path with elevation $\leq \delta$ from the start vertex to that cell.

Definition 5.1.5 For k curves $\pi_1, \ldots, \pi_k$, let

$$\mathsf{N}(\varepsilon, \pi_1, \ldots, \pi_k) = \max_{\delta \geq 0} \mathcal{N}_{\leq\delta}\big(\mathrm{simpl}(\pi_1, \varepsilon\delta), \ldots, \mathrm{simpl}(\pi_k, \varepsilon\delta)\big)$$

be the maximum complexity of the reachable free space for the simplified curves. We refer to $\mathsf{N}(\varepsilon, \pi_1, \ldots, \pi_k)$ as the ***ε-relative free space complexity*** of $\pi_1, \ldots, \pi_k$.

5.1.2 Subroutines

We now list the relevant subroutines from [DHW10], which carry over directly for our problem.

Using the same procedure as in [DHW10], one can build a decider, **decider**$(\delta, \varepsilon, \pi_1, \ldots, \pi_k)$ that runs in $O(\mathsf{N}(\varepsilon, \pi_1, \ldots, \pi_k))$ time (the only difference being that in our case the BFS ignores monotonicity). Specifically, we have the following.

Lemma 5.1.6 *Let $\pi_1, \ldots, \pi_k$ be k polygonal curves in $\mathbb{R}^d$ with total complexity n, and let $1 \geq \varepsilon > 0$ and $\delta > 0$ be two parameters. Then, there is an algorithm* **decider**$(\delta, \varepsilon, \pi_1, \ldots, \pi_k)$ *that, in $O(\mathsf{N}(\varepsilon, \pi_1, \ldots, \pi_k))$ time, returns one of the following outputs: (i) a $(1+\varepsilon)$-approximation to $\mathsf{d}_{med}(\pi_1, \ldots, \pi_k)$, (ii) $\mathsf{d}_{med}(\pi_1, \ldots, \pi_k) < \delta$, or (iii) $\mathsf{d}_{med}(\pi_1, \ldots, \pi_k) > \delta$.*

Definition 5.1.7 Given a finite set $Z \subseteq \mathbb{R}$, we say an interval $[\alpha, \beta]$ is ***atomic*** if it is a maximal interval on the real line that does not contain any point of Z in its interior.

Algorithm 5.1.8 For a set of numbers Z, let **searchEvents**$(Z, \varepsilon, \pi_1, \ldots, \pi_k)$ denote the algorithm that performs a binary search over the values of Z, to compute the atomic interval of Z that contains $\mathsf{d}_{med}(\pi_1, \ldots, \pi_k)$. This procedure would use **decider** (Lemma 5.1.6) to perform the decisions during the search.

Lemma 5.1.9 *Given a set P of n points in $\mathbb{R}^d$, let $\binom{\mathsf{P}}{2}$ be the set of all pairwise distances of points in P. Then, one can compute in $O(n \log n)$ time a set Z of $O(n)$ numbers, such that for any $y \in \binom{\mathsf{P}}{2}$, there exist numbers $x, x' \in Z$ such that $x \leq y \leq x' \leq 2x$. Let* **approxDistances**(P) *denote this algorithm.*

The following subroutine, from [DHW10], will allow us to efficiently check intervals with bounded spread for $\mathsf{d}_{med}(\pi_1, \ldots, \pi_k)$.

Lemma 5.1.10 *Given k curves $\pi_1, \ldots, \pi_k$ in $\mathbb{R}^d$ of total complexity n, a parameter $1 \geq \varepsilon > 0$, and an interval $[\alpha, \beta]$, one can compute a $(1+\varepsilon)$-approximation to $\mathsf{d}_{med}(\pi_1, \ldots, \pi_k)$ if $\mathsf{d}_{med}(\pi_1, \ldots, \pi_k) \in [\alpha, \beta]$, or report that $\mathsf{d}_{med}(\pi_1, \ldots, \pi_k) \notin [\alpha, \beta]$. The algorithm, denoted by* **searchInterval** *$([\alpha, \beta], \varepsilon, \pi_1, \ldots, \pi_k)$, takes $O\left(\mathsf{N}(\varepsilon, \pi_1, \ldots, \pi_k) \log \frac{\log(\beta/\alpha)}{\varepsilon}\right)$ time.*

We also need the following new ingredient.

Lemma 5.1.11 *Let $\pi_1, \ldots, \pi_k$ be k polygonal curves in $\mathbb{R}^d$ with total complexity n, $1 \geq \varepsilon > 0$ be a given parameter, $\delta^* = \mathsf{d}_{med}(\pi_1, \ldots, \pi_k)$, and $\mathsf{N} = \mathsf{N}(\varepsilon, \pi_1, \ldots, \pi_k)$. Let $[\alpha, \beta]$ be an atomic interval that contains δ^*, and such that for any $\mu, \mu' \in [\alpha, \beta]$, $\mathrm{simpl}(\pi_i, \mu) = \mathrm{simpl}(\pi_i, \mu')$ for $i = 1, \ldots, k$. Then one can compute in $O(\mathsf{N} \log \mathsf{N})$ time, a value δ such that $\delta^* \in [\delta - 2\alpha, \delta + 2\alpha]$. Let this algorithm be denoted by* **solver**$([\alpha, \beta], \pi_1, \ldots, \pi_k)$

Proof: Let $\mu = \beta$. We run the algorithm of Lemma 4.1.2 on $\pi'_1 = \mathrm{simpl}(\pi_1, \mu)$, $\ldots$, $\pi'_k = \mathrm{simpl}(\pi_k, \mu)$, except with the following modifications. First, instead of using the randomized algorithm for MSTs on the cell graph, we will use Prim's algorithm, starting from the vertex that corresponds to the starting points of the curves, where we stop when we reach the vertex that corresponds to the ending points of the curves. Also, instead of explicitly computing the cell graph, we only compute the relevant parts of the cell graph on the fly as they are needed for Prim's algorithm. Note that if δ is the elevation of the shortest path in the MST from s to t, then Prim's is guaranteed to stay within $\mathcal{N}_{\leq \delta}(\pi'_1, \ldots, \pi'_k)$ until reaching t.

This modified version of the algorithm computes a curve σ that minimizes $\max_i \mathsf{d}^{\mathrm{w}}_{\mathcal{F}}(\pi'_i, \sigma)$, in $O(\mathsf{N} \log \mathsf{N})$ time, since we are running Prim's on an effective graph of size N (and where $E(G) = O(V(G))$). Observe that since the μ simplification is constant on the interval $[\alpha, \beta]$, δ is the same value that would be returned had we set $\mu = \delta^*$. Also, again since the μ simplification is constant on this interval, by Observation 5.1.4 and considering $\mu = \alpha$, we know that $\delta^* \in [\delta - 2\alpha, \delta + 2\alpha]$. ■

5.2 Algorithm

Given k curves, $\pi_1, \ldots, \pi_k$, Figure 5.1 shows the algorithm to efficiently compute a $1+\varepsilon$ approximation to $\mathsf{d}_{med}(\pi_1, \ldots, \pi_k)$. Note that the algorithm depicted in Figure 5.1 performs numerous calls to **decider**, with an approximation parameter $\varepsilon > 0$. If any of these calls discovers the approximate distance, then the algorithm immediately stops and returns the approximation. As such, at

aprxMedian(ε, π_1, ...,π_k)
(A) $\mathsf{P} = V(\pi_1) \cup \cdots \cup V(\pi_k)$
(B) $Z \leftarrow$ **approxDistances**(P) (Lemma 5.1.9).
(C) $[\alpha, \beta] \leftarrow$ **searchEvents**$(Z, \varepsilon, \pi_1, \ldots, \pi_k)$ (Algorithm 5.1.8).
(D) Call **searchInterval**$([\alpha, 8\alpha], \varepsilon, \pi_1, \ldots, \pi_k)$ (Lemma 5.1.10).
(E) Call **searchInterval**$([\beta/2, \beta], \varepsilon, \pi_1, \ldots, \pi_k)$.
(F) $\delta \leftarrow$ **solver**$([2\alpha, \beta/2], \pi_1, \ldots, \pi_k)$ (Lemma 5.1.11).
(G) Return the value returned by **searchInterval**$([\delta/2, 3\delta/2], \varepsilon, \pi_1, \ldots, \pi_k)$.

Figure 5.1: The basic approximation algorithm.

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.

5.3 Correctness and Running Time

5.3.1 Correctness

In order to apply Lemma 5.1.11 we first need to find an atomic interval (or subinterval), $[\alpha, \beta]$, that contains $\delta^* = \mathsf{d}_{med}(\pi_1, \ldots, \pi_k)$, such that none of the μ simplifications of any of the k curves change for any choice of $\mu \in [\alpha, \beta]$. Note that by the way in which μ simplified curves are constructed, Algorithm 5.1.2, if we consider increasing the value of μ from 0 to ∞, the only events at which any of the μ simplifications of any of the curves change, are when μ is equal to one of the distances between a pair of vertices on one of the curves. Hence if Y denotes the set of all pairwise distances between vertices in P (step (A) in the algorithm) then in order to apply Lemma 5.1.11 we want the atomic interval with respect to Y that contains δ^*. Since it is costly to compute Y explicitly, we instead compute an $O(n)$ sized set Z (step (B)), such that each value in Y is 2-approximated by some value in Z. Step (C) performs a binary search over Z, using **decider**, in order to find an atomic interval $[\alpha, \beta]$ containing δ^*. Since each value in Y is 2-approximated by some value in Z, we know that the interval $[2\alpha, \beta/2]$ is a subinterval of an atomic interval of Y. Hence by Lemma 5.1.10 we know that steps (D) and (E) ensure that $[2\alpha, \beta/2]$ is a subinterval of an atomic interval of Y that contains δ^* (and if not, these steps returned a $(1+\varepsilon)$-approximation for δ^*). By Lemma 5.1.11 we know that in step (F), when we call **solver** on the interval $[2\alpha, \beta/2]$ we get a value δ such that $\delta^* \in [\delta - 4\alpha, \delta + 4\alpha]$. However, (D) guaranteed that $\delta \geq 8\alpha$, since we checked the interval $[\alpha, 8\alpha]$. This implies $\alpha \leq \delta/8$ and so $\delta^* \in [\delta - 4\alpha, \delta + 4\alpha]$ implies that $\delta^* \in [\delta - \delta/2, \delta + \delta/2]$. Hence we have an interval with bounded spread which contains δ^* and so by Lemma 5.1.10, (G) efficiently computes a $(1 + \varepsilon)$-approximation for δ^*.

5.3.2 Running Time

Let $n = |P|$ and $\mathsf{N} = \mathsf{N}(\varepsilon, \pi_1, \ldots, \pi_k)$. By Lemma 5.1.9 that the call to **approxDistances** in (B) takes $O(n \log n)$ time. Since **searchEvents** just preforms a binary search over the $O(n)$ values

returned by **approxDistances** by using **decider**, which runs in $O(\mathsf{N})$ time by Lemma 5.1.6, we know that (C) takes $O(\mathsf{N}\log n)$ time. Since $[\alpha, 8\alpha]$, $[\beta/2, \beta]$, and $[\delta/2, 3\delta/2]$ are all intervals with bounded spread, we have by Lemma 5.1.10, that steps (D), (E), and (G) run in $O(\mathsf{N}\log(1/\varepsilon))$ time. Finally, by Lemma 5.1.11, the call to **solver** in line (F) takes $O(\mathsf{N}\log\mathsf{N})$ time. We thus have the following.

Lemma 5.3.1 *Let $\pi_1, \ldots, \pi_k$ be k given polygonal c-packed curves in $\mathbb{R}^d$ of total complexity n, let $\varepsilon > 0$ be a parameter, and let $\mathsf{N} = \mathsf{N}(\varepsilon, \pi_1, \ldots, \pi_k)$. Then one can compute, in $O(\mathsf{N}\log(n/\varepsilon) + n\log n)$ time, reparameterizations of the curves that $1+\varepsilon$ approximate the value of $\mathsf{d}_{med}(\pi_1, \ldots, \pi_k)$. In particular, one can $1+\varepsilon$ approximate the median curve of $\pi_1, \ldots, \pi_k$.*

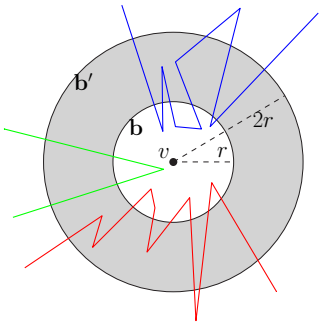
5.3.3 Free Space Complexity

Lemma 5.3.2 *For k c-packed curves $\pi_1, \ldots, \pi_k$ in $\mathbb{R}^d$ of total complexity n, and $0 < \varepsilon < 1$, we have that $\mathsf{N} = \mathsf{N}(\varepsilon, \pi_1, \ldots, \pi_k) = O((c/\varepsilon)^{k-1}n)$.*

Proof: Let $\delta \geq 0$ be a fixed parameter, $\mu = \varepsilon\delta$, and $\pi'_1 = \mathrm{simpl}(\pi_1, \mu)$, $\ldots$, $\pi'_k = \mathrm{simpl}(\pi_k, \mu)$.

The free space in the CW-complex is partitioned into connected components. We must bound the size of the component which contains the start vertex, that is the reachable free space, $\mathcal{R}$. Observe that one can charge a maximal dimensional cell in the CW-complex to an adjacent lower dimensional cell, since maximal cells contribute to N only if one of their adjacent proper subcells contributes to N. A non-maximal cell corresponds to some vertex v on one of the curves, and either a vertex or an edge from each one of the $k-1$ other curves. Consider a ball, $\mathbf{b}$, of radius $r = 2\delta$ centered at v. We now wish to count the number of features from the other curves (i.e. edges or vertices) that intersect this ball.

To this end, consider one of the other curves, π'_i. Let X_i be the set of all features of π'_i that intersect $\mathbf{b}$. Consider a ball, $\mathbf{b}'$, of radius $2r$ around v. Since $r \geq \mu$ and the edges of a μ simplified curve are of length $\geq \mu$ (with the exception of the last edge), every edge feature in X_i must contribute at least length μ to the intersection of $\mathbf{b}'$ and π'_i (note that if the feature is a vertex, then it is adjacent to an edge which contributes at least length μ). By Lemma 5.1.3, the total length of π'_i inside this $\mathbf{b}'$ is at most $12cr$. Therefore,



$$|X_i| = O\left(\frac{\|\pi'_i \cap \mathbf{b}'\|}{\mu}\right) = O\left(\frac{cr}{\mu}\right) = O\left(\frac{c\delta}{\varepsilon\delta}\right) = O\left(\frac{c}{\varepsilon}\right).$$

Similarly, for each of the other $k-1$ simplified curves, there are also $O(c/\varepsilon)$ features close enough to v, that can be involved in a cell that contributes to N. Such a cell in the CW-complex involves choosing the vertex v, and one of these $O(c/\varepsilon)$ features from each of the other $k-1$ curves, and

hence there are $|X_1| \cdot |X_2| \dots \cdot |X_{k-1}| = O((c/\varepsilon)^{k-1})$ such cells. Since there are n vertices in total we thus have that $\mathsf{N} = O((c/\varepsilon)^{k-1} n)$. ■

5.4 The Result

Theorem 5.4.1 *Let $\pi_1, \dots, \pi_k$ be k given polygonal c-packed curves in $\mathbb{R}^d$ with total complexity n, let $\varepsilon > 0$ be a parameter, and let $\mathsf{N} = \mathsf{N}(\varepsilon, \pi_1, \dots, \pi_k) = O((c/\varepsilon)^{k-1} n)$. Then one can compute, in $O(\mathsf{N} \log \mathsf{N})$ time, reparameterizations of the curves that $(1+\varepsilon)$-approximate the value of $\mathsf{d}_{med}(\pi_1, \dots, \pi_k)$. In particular, one can $1+\varepsilon$ approximate the median curve of $\pi_1, \dots, \pi_k$ in $O(\mathsf{N} \log \mathsf{N})$ time.*

Chapter 6

Computing Optimal Fréchet Paths for DAG Complexes

In this section, we present a simple algorithm for computing exactly the monotone Fréchet distance between two polygonal curves. This algorithm has running time $O\big(n^2 \log n\big)$ time, and uses randomization instead of parametric search. In fact, the algorithm is considerably more general and applies to a wider class of inputs.

DAG complexes. Consider a directed acyclic graph (DAG) with vertices in $\mathbb{R}^d$, where a directed edge $\mathsf{p} \to \mathsf{q}$ is realized by the segment pq. We refer to such a graph as being a **DAG *complex***. Given two DAG complexes $\mathcal{C}_1$ and $\mathcal{C}_2$, start vertices $s_1 \in V(\mathcal{C}_1), s_2 \in V(\mathcal{C}_2)$, and end vertices $t_1 \in V(\mathcal{C}_1), t_2 \in V(\mathcal{C}_2)$, the problem is finding two directed polygonal paths π_1, π_2 in $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively, such that:

(A) The path π_i uses only edges that appear in $\mathcal{C}_i$, and it traverses them in the direction compliant with the orientation of the edges in $\mathcal{C}_i$, for $i = 1, 2$.
(B) The curve π_i connects s_i to t_i in $\mathcal{C}_i$, for $i = 1, 2$.
(C) The monotone Fréchet distance between π_1 and π_2 is minimized among all such curves.

Note that this problem includes the problem of computing the monotone Fréchet distance between two polygonal curves (i.e., orient the edges of the curves in the natural way and consider them to be DAG complexes).

6.1 The Decision Procedure

The algorithm is a direct extension of the work of [AG95]. Their algorithm relied on the fact that there was a clear topological ordering on the cells of the free space, and hence reachability information could be propagated. In this case, there is also a topological ordering (since it is a DAG). Hence, in the product space of two DAG complexes there is an ordering of the cells according to the underlying ordering of the two DAGs, and this ordering is acyclic.

So, let $\mathcal{C}_1$ and $\mathcal{C}_2$ be the two given DAG complexes, δ a specified radius, and s_1, s_2, t_1, t_2 the given vertices. The problem is to decide if there are paths between the start and end vertices in the corresponding complexes of Fréchet distance at most δ.

Algorithm. Compute the topological orderings of the cells (i.e., vertices and edges) of $\mathcal{C}_1$ and $\mathcal{C}_2$. In the resulting ordering $\prec_i$, it holds that $\Delta \prec_i \Delta'$ if Δ appears before Δ' in this ordering, for $i = 1, 2$, where $\Delta, \Delta' \in \mathcal{C}_i$.

We compute the product complex $\mathcal{C} = \mathcal{C}_1 \times \mathcal{C}_2$, and compute the topological ordering of the cells of $\mathcal{C}$. Formally for $\Delta = (\Delta_1, \Delta_2), \Delta' = (\Delta'_1, \Delta'_2) \in \mathcal{C}$ we have that $\Delta \preceq \Delta'$ if and only if $\Delta_1 \preceq_1 \Delta'_1$ and $\Delta_2 \preceq_2 \Delta'_2$. Clearly, the ordering $\prec$ over the cells of $\mathcal{C}$ is acyclic, and can be computed in linear time in the size of the complex.

Now, just as in [AG95], we start at the start vertex in the product space (s_1, s_2), visit cells according to their topological order, and compute the Freely space and propagate reachability information on the fly when we reach a new cell.

Since we are working in the product space instead of in the parametric space, the two dimensional cells are parallelograms instead of squares.

The reachability information is being propagated in a manner similar to [AG95], except we propagate between adjacent cells, instead of neighboring two dimensional cells. Note, that no pair of two dimensional cells are directly adjacent, as there must be a one dimensional cell separating them. As such, for each edge (i.e., one dimensional cell) of $\mathcal{C}$ we maintain the set of reachable points. Unlike in [AG95], the reachability information along a bounding edge in the product space might not be a single interval, since potentially multiple cells propagate to that bounding edge. However, by Lemma 3.1.1, we only need to compute the first point (according to the ordering along this edge) that is reachable on this edge (notice, that an edge is always a product of a vertex of one curve and a directed edge of the other curve, and as such it has a natural ordering).

In particular, when the algorithm visits a cell Δ in this ordering, it fetches all the cells that are adjacent to it and appear before it in the ordering. For each adjacent cell, the reachability information computed is of constant size, and hence we can compute the reachability information for the new cell in constant time. Indeed, the handling depends on the dimension of Δ:

(A) $\dim(\Delta) = 0$: (Δ is a vertex), the algorithm computes if it is reachable from any of its direct ancestors, and if so we mark it as reached.

(B) $\dim(\Delta) = 1$: (Δ is an edge), the algorithm computes the first point on the edge reachable from its direct ancestors.

(C) $\dim(\Delta) = 2$: (Δ is a parallelogram), the algorithm uses the reachability information on the two incoming edges and the incoming vertex to compute the reachability inside the parallelogram. (Clipping the region to $F_{\leq\delta}$ inside this cell.)

As the algorithm visits the cells in a topological order, the work in maintaining the reachability information, can be charged to a cell's predecessors. As such, overall, the running time of the algorithm is linear in the complex size.

The size of a DAG complex is the number of edges in it (since we assumed implicitly that the input DAG complexes are connected). Let n be the number of edges in the larger of the two DAG complexes under consideration. There are potentially $O(n^2)$ cells in the product space $\mathcal{C}$. As such, the running time of the decision procedure is $O(n^2)$.

Lemma 6.1.1 *Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two DAG complexes, n be the number of edges in the larger of the two, $s_1, t_1 \in \mathcal{C}_1, s_2, t_2 \in \mathcal{C}_2$ be start and end vertices, and $\delta \geq 0$ be a parameter. Then, one can decide, in $O(n^2)$ time, if there exists two paths π_1 and π_2 in $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively, such that (i) π_i connects s_i with t_i, for $i = 1, 2$, and (ii) $\mathsf{d}_{\mathcal{F}}^{\mathrm{m}}(\pi_1, \pi_2) \leq \delta$. Furthermore, if such paths exist, the algorithm returns them together with their respective reparameterizations realizing this distance.*

6.2 Using the Decision Procedure

In the following, let $\mathcal{C}_1$ and $\mathcal{C}_2$ be the two DAG complexes under consideration. We outline a randomized algorithm to compute the value of the Fréchet distance between the two curves in $\mathcal{C}_1$ and $\mathcal{C}_2$, that start and end at their respective start and end vertices, that minimize the Fréchet distance.

The algorithm needs to search over the critical values when the decision procedure changes its behavior. These critical values are the same as in Alt and Godau [AG95] (vertex-vertex, vertex-edge and monotonicity events). Indeed, for any pair of paths in the DAG complexes, the critical values for these two paths are the same as in [AG95]. As such, since DAG complexes are the union of paths, the critical values are the same.

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 above. Let **extract**(a, b) be a procedure that returns all critical values determined by $\mathcal{C}_1$ and $\mathcal{C}_2$ whose radius is in the interval $[a, b]$. Suppose, for the time being, that the following subroutines have the following running times:

(A) **decider**(δ) runs in $O(n^2)$ time (Lemma 6.1.1).
(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 6.1.

6.2.1 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.

compFr$(\mathcal{C}_1, \mathcal{C}_2, s_1, s_2, t_1, t_2)$:
    R: random sample of $\mu = 4n^2$ critical values
    Sort R
    Perform a binary search over R using **decider**
    $\mathcal{I} = [a, b] \leftarrow$ Atomic interval of R containing δ^*
    $S \leftarrow$ **extract**(a, b)
        `// S: all critical values in` $[a, b]$
    Sort S
    $x \leftarrow$ Smallest value in S for which **decider** accepts
        `// Computed using a binary search`
    Return x

Figure 6.1: The algorithm for computing the Fréchet distance between two DAG complexes.

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 two DAG complexes $\mathcal{C}_1$ and $\mathcal{C}_2$, consider finding all monotonicity events between an edge e of $\mathcal{C}_1$, and pairs of vertices from $V = V(\mathcal{C}_2)$. To this end, place a sphere of radius δ at each point of V with radius $\delta = a$. We now increase the radius δ till 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 $\mathcal{C}_1$ and hence it takes $O((n^2 + k') \log n)$ time to compute all the monotonicity events between edges of $\mathcal{C}_1$ and vertices of $\mathcal{C}_2$, 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 $\mathcal{C}_1$. Therefore it takes $O((n^2 + k'') \log n)$ time to compute all the relevant monotonicity events between $\mathcal{C}_1$ and $\mathcal{C}_2$, where k'' is the number of such events (i.e. both those involving edges of $\mathcal{C}_1$ and those involving edges of $\mathcal{C}_2$).

6.2.2 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.

6.3 Analysis

Let R be the random sample of critical values, of size $O(n^2)$. The interval $[a, b]$ computed by **compFr** 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 6.3.1 *Let $\mathcal{I} = [a, b]$ be the interval computed by* **compFr**, *and let c be some positive constant. Then,*

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

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 all our results, we thus have the following theorem.

Theorem 6.3.2 *For two DAG complexes, $\mathcal{C}_1$ and $\mathcal{C}_2$, of total complexity n, with start and end vertices $s_1, t_1 \in \mathcal{C}_1, s_2, t_2 \in \mathcal{C}_2$, the algorithm* **compFr**$(\mathcal{C}_1, \mathcal{C}_2, s_1, t_1, s_2, t_2)$ *returns two curves π_1 and*

π_2, such that π_1 (resp. π_2) connects s_1 (reps. s_2) to t_1 (resp. t_2) in $\mathcal{C}_1$ (resp. $\mathcal{C}_2$). Furthermore, the monotone Fréchet distance between π_1 and π_2, is the minimum among all such curves. The running time of the algorithm is $O\big(n^2 \log n\big)$ time, with probability $\geq 1 - 1/n^c$.

Remark 6.3.3 The above result implies that given two polygonal curves in $\mathbb{R}^d$ one can compute the Fréchet distance between them, in $O(n^2 \log n)$ time (this running time bound holds with high probability), by a simple algorithm that does not use parametric search.

Chapter 7

Conclusions

In this thesis, we showed that the algorithm for computing the (weak) Fréchet distance between two curves can be extended to more general settings. This results in a slew of problems that can be solved using the new framework.

Monotonicity. Our main algorithm from Chapter 3 is an extension of the algorithm of Alt and Godau [AG95] for the weak Fréchet distance. It is natural to ask if the new framework can handle monotonicity. In Chapter 6, we offered a very restricted extension of our framework to this case, in the process presenting a new simpler algorithm for computing the monotone Fréchet distance between polygonal curves.

For more general settings, if the underlying complex is not one dimensional then it is not clear what monotonicity means. Even if we restrict ourselves to the case of k input curves, for $k > 2$, it is not immediately clear how to handle monotonicity efficiently, and we leave this as an open problem for further research. Interestingly, there are cases where monotonicity actually makes the problem easier.

Running Time. The expected running time of the general algorithm is $O\big(n^k\big)$ when handling k input complexes and is probably practical only for very small values of k. In Chapter 5 we showed that one can get a $(1+\varepsilon)$-approximation for the median curve problem for k c-packed curves in $\widetilde{O}(n \log n)$ time. It should be possible to extend this same procedure to approximate, in a similar running time, some of the other problems that are solved by the general framework, under similar assumptions on the input.

References

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

[AC10] H. Ahn and O. Cheong. Aligning two convex figures to minimize area or perimeter. *Algorithmica*, 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. *Internat. J. Comput. Geom. Appl.*, 5:75–91, 1995.

[BBG08a] K. Buchin, M. Buchin, and J. Gudmundsson. Detecting single file movement. In *Proc. 16th ACM SIGSPATIAL Int. Conf. Adv. GIS*, pages 288–297, 2008.

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

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

[BBS10] K. Buchin, M. Buchin, and A. Schulz. Fréchet distance of surfaces: Some simple hard cases. In *Proc. 18th Annu. European Sympos. Algorithms*, pages 63–74, 2010.

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

[CW09] A. F. Cook and C. Wenk. Geodesic fréchet distance inside a simple polygon. *ACM Trans. Algo.*, page to appear, 2009.

[DHW10] A. Driemel, S. Har-Peled, and C. Wenk. Approximating the Fréchet distance for realistic curves in near linear time. In *Proc. 26th Annu. ACM Sympos. Comput. Geom.*, pages 365–374, 2010. `http://arxiv.org/abs/1003.0460`.

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

[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, August 1998.

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

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

[Mat91] J. Matoušek. Randomized optimal algorithm for slope selection. *Inform. Process. Lett.*, 39:183–187, 1991.

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

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, New York, NY, 1995.

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

[vOV04] R. van Oostrum and R. C. Veltkamp. Parametric search made practical. *Comput. Geom. Theory Appl.*, 28(2-3):75–88, 2004.

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