Computing the Discrete Fréchet Distance with Imprecise Input

Hee-Kap Ahn[1], Christian Knauer[2], Marc Scherfenberg[2], Lena Schlipf[3], and Antoine Vigneron[4]

[1] Department of Computer Science and Engineering, POSTECH, Pohang, Korea
heekap@postech.ac.kr
[2] Institute of Computer Science, Universität Bayreuth, 95440 Bayreuth, Germany
{Christian.knauer,marc.scherfenberg}@uni-bayreuth.de
[3] Institute of Computer Science, Freie Universität Berlin, Germany
schlipf@mi.fu-berlin.de
[4] INRA, UR 341 Mathématiques et Informatique Appliquées, 78352 Jouy-en-Josas, France
antoine.vigneron@jouy.inra.fr

Abstract. We consider the problem of computing the discrete Fréchet distance between two polygonal curves when their vertices are imprecise. An imprecise point is given by a region and this point could lie anywhere within this region. By modelling imprecise points as balls in dimension d, we present an algorithm for this problem that returns in time $2^{O(d^2)}m^2n^2\log^2(mn)$ the Fréchet distance lower bound between two imprecise polygonal curves with n and m vertices, respectively. We give an improved algorithm for the planar case with running time $O(mn\log^2(mn)+(m^2+n^2)\log(mn))$. In the d-dimensional orthogonal case, where points are modelled as axis-parallel boxes, and we use the L_∞ distance, we give an $O(dmn\log(dmn))$-time algorithm.

We also give efficient $O(dmn)$-time algorithms to approximate the Fréchet distance upper bound, as well as the smallest possible Fréchet distance lower/upper bound that can be achieved between two imprecise point sequences when one is allowed to translate them. These algorithms achieve constant factor approximation ratios in "realistic" settings (such as when the radii of the balls modelling the imprecise points are roughly of the same size).

1 Introduction

Shape matching is an important ingredient in a wide range of computer applications such as computer vision, computer–aided design, robotics, medical imaging, and drug design. In shape matching, we are given two geometric objects and we compute their distance according to some geometric similarity measure. The Fréchet distance is a natural distance function for continuous shapes such as curves and surfaces, and is defined using reparameterizations of the shapes [3,4,5,16].

O. Cheong, K.-Y. Chwa, and K. Park (Eds.): ISAAC 2010, Part II, LNCS 6507, pp. 422–433, 2010.

The discrete Fréchet distance is a variant of the Fréchet distance in which we only consider vertices of polygonal curves. In dimension d, given two polygonal curves with n and m vertices, respectively, there is a dynamic programming algorithm that computes the discrete Fréchet distance between them in $\Theta(dmn)$ time [9]. Later, Aronov et al. [6] presented efficient approximation algorithms for computing the discrete Fréchet distance of two natural classes of curves: κ-bounded curves and backbone curves. They also proposed a pseudo-output-sensitive algorithm for computing the discrete Fréchet distance exactly.

Most of previous works on the Fréchet distance assume that the input curves are given precisely. The input curve, however, could be only an approximation; In many cases, geometric data comes from measurements of continuous real-world phenomenons, and the measuring devices have finite precision. This impreciseness of geometric data has been studied lately, and quite a few algorithms that handle imprecise data have been given for fundamental geometric problems: for example, computing the Hausdorff distance [12], Voronoi diagrams [17], planar convex hulls [13], and Delaunay triangulations [11,14].

Imprecise data can be modelled in different ways. One possible model, for data that consists of points, is to assign each point to a region, typically a disk or a square. In this case, existing algorithms for computing the Fréchet distance could be too sensitive to the precision of the measurements, and they may return a solution without providing any guarantee on its correctness or preciseness. One solution to this problem is to take the impreciseness of the input into account in the design of algorithms, so that they return a solution with some additional information on its quality.

Our results. In this paper, we study the problem of computing the discrete Fréchet distance between two polygonal curves, where the vertices of a polygonal curve are imprecise. Each vertex belongs to a region, which is either a Euclidean ball or an axis-parallel box in $\mathbb{R}^d$. We consider two cases: the orthogonal case and the Euclidean case. In the orthogonal case, the regions are boxes, and we use the L_∞ distance. In the Euclidean case, the regions are balls and we use the Euclidean distance.

Typical applications of this problem include computing similarity of two spatio-temporal data sets such as polygonal trajectories of moving objects (e.g. cars, people, animals) whose vertex locations are obtained by some positioning services (e.g. the Global Positioning System), and therefore imprecise.

Given two imprecise sequences of n and m points, respectively, we give algorithms for computing the Fréchet distance lower bound between these two sequences. In the d-dimensional orthogonal case, our algorithm runs in time $O(dmn \log(dmn))$. In the Euclidean case, we give an $2^{O(d^2)}m^2n^2 \log^2(mn)$-time algorithm for arbitrary dimension d, and we give an improved $O(mn \log^2(mn) + (m^2 + n^2) \log(mn))$-time algorithm in the plane.

We also give efficient $O(dmn)$-time algorithms to approximate the Fréchet distance upper bound, as well as the smallest possible Fréchet distance lower and upper bound that can be achieved between two imprecise point sequences when one is allowed to translate them. These algorithms achieve constant factor

approximation ratios in realistic settings, such as when the radii of the balls modelling the imprecise points are roughly of the same size, or when any two consecutive imprecise points are well-separated (so that their imprecision regions do not overlap).

2 Notation and Preliminaries

We work in $\mathbb{R}^d$, and we use a metric $\mathrm{dist}(\cdot,\cdot)$ which is either the Euclidean distance, or the L_∞ distance. Let $A = a_1, \ldots, a_n$ and $B = b_1, \ldots, b_m$ denote two sequences of points in $\mathbb{R}^d$. A *coupling* is a sequence of ordered pairs $(\alpha_1, \beta_1), \ldots, (\alpha_c, \beta_c)$ such that:

- $\alpha_1 = 1$, $\beta_1 = 1$, $\alpha_c = n$ and $\beta_c = m$.
- for each $1 \leqslant k < c$, one of the three statements below is true:
  - $\alpha_{k+1} = \alpha_k + 1$ and $\beta_{k+1} = \beta_k + 1$.
  - $\alpha_{k+1} = \alpha_k + 1$ and $\beta_{k+1} = \beta_k$.
  - $\beta_{k+1} = \beta_k + 1$ and $\alpha_{k+1} = \alpha_k$

The *discrete Fréchet distance* $\mathrm{F}(A, B)$ is the minimum, over all couplings, of $\max_{1 \leqslant k \leqslant c} \mathrm{dist}(a_{\alpha_k}, b_{\beta_k})$. (See Figure 1.)

In what follows, we consider the case where the two point-sequences A and B are *imprecise*. So, instead of knowing the position of each a_i, b_j, we are given two sequences of regions of $\mathbb{R}^d$ denoted by $H = h_1, \ldots, h_n$ and $V = v_1, \ldots, v_m$. These regions will be either Euclidean balls, or axis-aligned boxes. They specify where the points a_i, b_j may lie, and thus for each i, j, we have $a_i \in h_i$ and $b_j \in v_j$. For all $i \leqslant n$, we denote by H_i the subsequence $h_1, \ldots, h_i$, and for all $j \leqslant m$, we denote $V_j = v_1, \ldots, v_j$.

We will consider two different cases. In the *Euclidean case*, the regions are Euclidean balls in $\mathbb{R}^d$ and we use the Euclidean distance. In the *orthogonal case*, the regions are axis-aligned boxes and the distance we use is the L_∞ metric.

A *realization* of the region sequence H is a point sequence $A = a_1, \ldots, a_n$ such that $a_i \in h_i$ for all $1 \leqslant i \leqslant n$. Similarly, a realization of the region sequence V is a point sequence $B = b_1, \ldots, b_m$ such that $b_j \in v_j$ for all $1 \leqslant j \leqslant m$. We denote by $A \in_R H$ and $B \in_R V$ the fact that A is a realization of H, and B is a realization of V, respectively. When $A \in_R H$ and $B \in_R V$, we will say that (A, B) is a realization of (H, V). This will be denoted as $(A, B) \in_R (H, V)$.

Definition 1. *For two region sequences H and V, the* Fréchet distance lower bound $\mathrm{F}^{\min}(H, V)$ *is the minimum, over all realizations (A, B) of (H, V), of the discrete Fréchet distance* $\mathrm{F}(A, B)$*:*

$$\mathrm{F}^{\min}(H, V) = \min_{(A,B) \in_R (H,V)} \mathrm{F}(A, B).$$

The Fréchet distance upper bound $\mathrm{F}^{\max}(H, V)$ *is the maximum, over all realizations (A, B) of (H, V), of the discrete Fréchet distance* $\mathrm{F}(A, B)$*:*

$$\mathrm{F}^{\max}(H, V) = \max_{(A,B) \in_R (H,V)} \mathrm{F}(A, B).$$

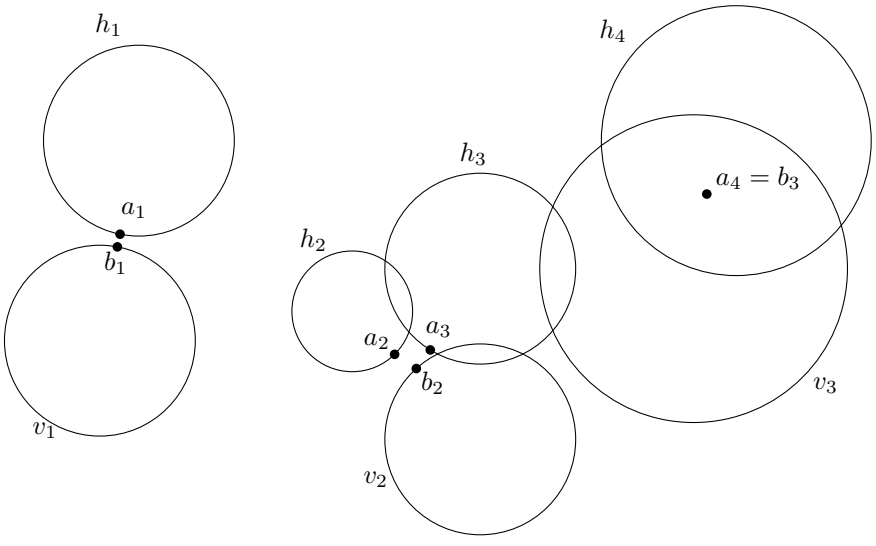

Fig. 1. The discrete Fréchet distance between the sequences $A = a_1, a_2, a_3, a_4$ and $B = b_1, b_2, b_3$ is achieved by the coupling $(1,1),(2,2),(3,2),(4,3)$, and we have $\mathrm{F}(A,B) = \mathrm{dist}(a_2,b_2) = \mathrm{dist}(a_3,b_2)$. The sequences A and B are realizations of the sequences of regions $H = h_1, h_2, h_3, h_4$ and $V = v_1, v_2, v_3$, which is denoted by $(A,B) \in_R (H,V)$. The Fréchet distance lower bound $\mathrm{F}^{\min}(H,V)$ is achieved by the realization (A,B), so we have $\mathrm{F}^{\min}(H,V) = \mathrm{F}(A,B)$.

3 Computing the Fréchet Distance Lower Bound $\mathrm{F}^{\min}$

In this section, we give algorithms for computing $\mathrm{F}^{\min}(H,V)$. We first give a decision algorithm that, given a real number $\delta \geqslant 0$, decides whether $\mathrm{F}^{\min}(H,V) \leqslant \delta$. Then we give an improved decision algorithm for the Euclidean case. Based on these decision algorithms, we finally give optimization algorithms, which compute $\mathrm{F}^{\min}(H,V)$ in the orthogonal case and in the Euclidean case.

We denote by h_i^δ (resp. v_j^δ) the set of points that are at distance at most δ from h_i (resp. v_j). In the Euclidean case, where h_i is a ball with radius r, the set h_i^δ is the concentric ball with radius $r + \delta$. In the orthogonal case, if $h_i = [x_1, y_1] \times \cdots \times [x_d, y_d]$, we have $h_i^\delta = [x_1 - \delta, y_1 + \delta] \times \cdots \times [x_d - \delta, y_d + \delta]$.

3.1 Decision Algorithm for the Orthogonal Case

Our decision algorithm is based on dynamic programming. In this sense, it is related to Eiter and Mannila's algorithm [9] for computing the discrete Fréchet distance, but we use additional invariants to address the impreciseness. These new invariants are carefully chosen *feasibility regions*, which indicate where the current points (a_i, b_j) may lie. Note that a straightforward discretization of the space of realizations of H, V would yield an exponential time bound, because one

would have to consider the arrangement of nm surfaces in dimension $(m+n)d$ defined by the equation $\text{dist}(a_i, b_j) \leqslant \delta$ for each pair i, j.

So in each cell of an array with n rows and m columns, we will store two feasibility regions $\text{FH}_\delta(i,j) \subset \mathbb{R}^d$ and $\text{FV}_\delta(i,j) \subset \mathbb{R}^d$. The ith row represents the region H_i, and the jth column represents V_j. We will compute these fields row by row, from $i = 1$ to $i = n$.

Remember that A_i (resp. B_j) denotes the sequence $a_1, \ldots, a_i$ (resp. $b_1, \ldots, b_j$). As we shall see in Lemma 1, the feasibility region $\text{FH}_\delta(i,j)$ represents the possible locations of a_i, where (A_i, B_j) is a realization of (H_i, V_j), and there exists a coupling that achieves $\text{F}(A_i, B_j) \leqslant \delta$ whose last two pairs are not $(i-1, j), (i, j)$. The other feasibility region $\text{FV}_\delta(i,j)$ represents the possible locations of b_j, when there is such a coupling whose last two pairs are not $(i, j-1), (i, j)$. Thus, the Fréchet distance lower bound $\text{F}^{\min}(H_i, V_j)$ is more than δ if and only if both of these feasibility regions $\text{FH}_\delta(i,j)$ and $\text{FV}_\delta(i,j)$ are empty.

The pseudocode of our decision algorithm *DecideFréchetMin* is given below. Lines 1 to 8 initialize some of the fields of our array for the first row and column, as well as an extra zeroth column and row. It allows boundary cases when $i = 1$ and $j = 1$ to be handled correctly in the main loop. The main loop is from line 9 to 15. As we are in the orthogonal case, lines 12–15 consist in intersecting two axis-aligned boxes in dimension d. It can be done trivially in $O(d)$ time, so our algorithm runs in $O(dmn)$ time.

Algorithm *DecideFréchetMin*
Input: Two sequences of regions $H = h_1, \ldots, h_n$ and $V = v_1, \ldots, v_m$, and a value $\delta \geqslant 0$.
Output: TRUE when $\text{F}^{\min}(H, V) \leqslant \delta$, and FALSE otherwise.

1. **for** $i \leftarrow 1$ **to** n
2. $\quad \text{FH}_\delta(i, 0) \leftarrow \emptyset$
3. $\quad \text{FV}_\delta(i, 0) \leftarrow \emptyset$
4. **for** $j \leftarrow 1$ **to** m
5. $\quad \text{FH}_\delta(0, j) \leftarrow \emptyset$
6. $\quad \text{FV}_\delta(0, j) \leftarrow \emptyset$
7. $\text{FH}_\delta(0, 0) \leftarrow \mathbb{R}^d$
8. $\text{FV}_\delta(0, 0) \leftarrow \mathbb{R}^d$
9. **for** $i \leftarrow 1$ **to** n
10. $\quad$ **for** $j \leftarrow 1$ **to** m
11. $\quad\quad$ **if** $\text{FH}_\delta(i-1, j-1) = \emptyset$ **and** $\text{FV}_\delta(i-1, j-1) = \emptyset$
12. $\quad\quad\quad$ **then** $\text{FH}_\delta(i, j) \leftarrow \text{FH}_\delta(i, j-1) \cap v_j^\delta$
13. $\quad\quad\quad\quad\quad \text{FV}_\delta(i, j) \leftarrow \text{FV}_\delta(i-1, j) \cap h_i^\delta$
14. $\quad\quad\quad$ **else** $\text{FH}_\delta(i, j) \leftarrow h_i \cap v_j^\delta$
15. $\quad\quad\quad\quad\quad \text{FV}_\delta(i, j) \leftarrow h_i^\delta \cap v_j$
16. **if** $\text{FH}_\delta(n, m) = \emptyset$ **and** $\text{FV}_\delta(n, m) = \emptyset$
17. $\quad$ **then return** FALSE
18. $\quad$ **else return** TRUE

In order to prove that our decision algorithm *DecideFréchetMin* is correct, we need the following lemma.

Lemma 1. *For any $2 \leqslant i \leqslant n$, $2 \leqslant j \leqslant m$, we have $\mathrm{F}^{\min}(H_i, V_j) \leqslant \delta$ if and only if $\mathrm{FH}_\delta(i,j) \neq \emptyset$ or $\mathrm{FV}_\delta(i,j) \neq \emptyset$. More precisely, for any $x, y \in \mathbb{R}^d$, we have:*

(a) $x \in \mathrm{FH}_\delta(i,j)$ if and only if there exists $(A_i, B_j) \in_R (H_i, V_j)$ such that $a_i = x$, and such that there exists a coupling achieving $\mathrm{F}(A_i, B_j) \leqslant \delta$ whose last two pairs are not $(i-1,j),(i,j)$.
(b) $y \in \mathrm{FV}_\delta(i,j)$ if and only if there exists $(A_i, B_j) \in_R (H_i, V_j)$ such that $b_j = y$, and such that there exists a coupling achieving $\mathrm{F}(A_i, B_j) \leqslant \delta$ whose last two pairs are not $(i,j-1),(i,j)$.

We now prove Lemma 1 when $i, j \geqslant 3$. The boundary cases where $i = 2$ or $j = 2$ can be easily checked. We only prove Lemma 1(a); the proof of (b) is similar. Our proof is done by induction on (i,j), so we assume that Lemma 1 is true for all the cells that have been handled before cell (i,j) by our algorithm; in particular, it is true for all cells $(i',j') \neq (i,j)$ such that $i' \leqslant i$ and $j' \leqslant j$.

We first assume that $x \in \mathrm{FH}_\delta(i,j)$, and we want to prove that there exists $(A_i, B_j) \in_R (H_i, V_j)$ such that $a_i = x$, and such that there exists a coupling achieving $\mathrm{F}(A_i, B_j) \leqslant \delta$ whose last two pairs are not $(i-1,j),(i,j)$. We distinguish between two cases:

- First case: $\mathrm{FH}_\delta(i-1,j-1) \neq \emptyset$ or $\mathrm{FV}_\delta(i-1,j-1) \neq \emptyset$. Then, by induction, there exists $(A_{i-1}, B_{j-1}) \in_R (H_{i-1}, V_{j-1})$ such that $\mathrm{F}(A_{i-1}, B_{j-1}) \leqslant \delta$. We also know that $\mathrm{FH}_\delta(i,j)$ was set to $h_i \cap v_j^\delta$ at line 14. In other words, $x \in h_i$, and there exists $y' \in v_j$ such that $\mathrm{dist}(x,y') \leqslant \delta$. So we extend A_{i-1} and B_{j-1} by choosing $a_i = x$ and $b_j = y'$. We extend a coupling achieving $\mathrm{F}(A_{i-1}, B_{j-1}) \leqslant \delta$ with the pair (i,j), and obtain a coupling achieving $\mathrm{F}(A_i, B_j) \leqslant \delta$ whose last two pairs are $(i-1,j-1),(i,j)$.
- Second case: $\mathrm{FH}_\delta(i-1,j-1) = \emptyset$ and $\mathrm{FV}_\delta(i-1,j-1) = \emptyset$. Then $\mathrm{FH}_\delta(i,j)$ was set to $\mathrm{FH}_\delta(i,j-1) \cap v_j^\delta$ at line 12. Thus $x \in \mathrm{FH}_\delta(i,j-1)$, so by induction, there exists $(A_i, B_{j-1}) \in_R (H_i, V_{j-1})$ such that $a_i = x$ and $\mathrm{F}(A_i, B_{j-1}) \leqslant \delta$. Since $x \in v_j^\delta$, there exists $y' \in v_j$ such that $\mathrm{dist}(x,y') \leqslant \delta$. So we extend B_{j-1} by choosing $b_j = y'$. We extend a coupling achieving $\mathrm{F}(A_i, B_{j-1}) = \delta$ with the pair (i,j), and we obtain a coupling achieving $\mathrm{F}(A_i, B_j) \leqslant \delta$ whose last two pairs are $(i,j-1),(i,j)$.

Now we assume that there exists $(A_i, B_j) \in_R (H_i, V_j)$ such that there exists a coupling $\mathcal{C}$ achieving $\mathrm{F}(A_i, B_j) \leqslant \delta$ whose last two pairs are not $(i-1,j),(i,j)$. We want to prove that $a_i \in \mathrm{FH}_\delta(i,j)$. We distinguish between two cases:

- First case: $\mathrm{FH}_\delta(i-1,j-1) \neq \emptyset$ or $\mathrm{FV}_\delta(i-1,j-1) \neq \emptyset$. It implies that $\mathrm{FH}_\delta(i,j)$ was set to $h_i \cap v_j^\delta$ at line 14. Since $A_i \in_R H_i$, we have $a_i \in h_i$. Since $B_j \in_R V_j$ and $\mathrm{F}(A_i, B_j) \leqslant \delta$, it follows that $\mathrm{dist}(a_i, b_j) \leqslant \delta$, and thus $a_i \in v_j^\delta$. Thus, $a_i \in \mathrm{FH}_\delta(i,j)$.
- Second case: $\mathrm{FH}_\delta(i-1,j-1) = \emptyset$ and $\mathrm{FV}_\delta(i-1,j-1) = \emptyset$. Then, by induction, we have $\mathrm{F}^{\min}(H_{i-1}, V_{j-1}) > \delta$, which implies that $\mathrm{F}(A_{i-1}, B_{j-1}) > \delta$, so the pair $(i-1,j-1)$ cannot appear in $\mathcal{C}$. It follows that the last three pairs of $\mathcal{C}$ can only be $(i,j-2),(i,j-1),(i,j)$ or $(i-1,j-2),(i,j-1),(i,j)$.

So, by induction, we have $a_i \in \mathrm{FH}_\delta(i, j-1)$. Since $\mathrm{F}(A_i, B_j) \leqslant \delta$, we have $a_i \in v_j^\delta$. As $\mathrm{FH}_\delta(i-1, j-1) = \emptyset$ and $\mathrm{FV}_\delta(i-1, j-1) = \emptyset$, the value of $\mathrm{FH}_\delta(i, j)$ was set to $\mathrm{FH}_\delta(i, j-1) \cap v_j^\delta$ at line 14, so we have $a_i \in \mathrm{FH}_\delta(i, j)$.

This completes the proof of Lemma 1. It follows immediately from Lemma 1 that Algorithm *DecideFréchetMin* decides correctly whether $\mathrm{F}^{\min}(H, V) \leqslant \delta$. As we observed above, our algorithm runs in $O(dmn)$ time. Thus, we obtain the following result:

Theorem 1. *In the d-dimensional orthogonal case, given $\delta \geqslant 0$, and given two imprecise sequences H and V of n and m points, respectively, we can decide in $O(dmn)$ time whether $\mathrm{F}^{\min}(H, V) \leq \delta$.*

3.2 Decision Algorithm for the Euclidean Case

In this section, we give an efficient algorithm for the Euclidean case. A naive implementation of Algorithm *DecideFréchetMin* would require to construct the regions $\mathrm{FH}_\delta(i, j)$ and $\mathrm{FV}_\delta(i, j)$, which may be intersections of $\Omega(n)$ balls in $\mathbb{R}^d$. Even in $\mathbb{R}^2$, it would increase the running time of our algorithm by an order of magnitude. To improve the running time, we will show how to compute these intersections in amortized $2^{O(d^2)} \log(mn)$ time per step. We will need the following result:

Lemma 2. *We can decide in $2^{O(d^2)}k$ time whether k balls in d-dimensional Euclidean space have an empty intersection.*

Proof. We consider a collection of k balls in $\mathbb{R}^d$. We use the standard lifting-map [8, Section 1.2], which maps any point $x = (x_1, \ldots, x_d) \in \mathbb{R}^d$ to the point $\hat{x} = \left(x_1, \ldots, x_d, \sum_{i=1}^d x_i^2\right) \in \mathbb{R}^{d+1}$. Then a ball $\mathcal{B} \subset \mathbb{R}^d$ can be mapped to an affine hyperplane $\mathcal{H} \subset \mathbb{R}^{d+1}$ such that $x \in \mathcal{B}$ if and only if $\hat{x}$ is below $\mathcal{H}$. Thus, deciding whether k balls have a non-empty intersection reduces to deciding whether there is a point x such that $\hat{x}$ is below all the corresponding hyperplanes. To do this, it suffices to decide whether there is a point $\hat{y} = (y_1, \ldots, y_{d+1})$ below all these hyperplanes and such that $\sum_{i=1}^d y_i^2 \leqslant y_{d+1}$. It can be done in $2^{O(d^2)}k$ time using an algorithm of Dyer [7] for some generalized linear programs in fixed dimension; in our case, the linear constraints for Dyer's algorithm are given by our set of hyperplanes, and the convex function we use is $(y_1, \ldots, y_{d+1}) \mapsto -y_{d+1} + \sum_{i=1}^d y_i^2$.

We now explain how we implement line 13 in amortized $2^{O(d^2)} \log n$ time. We fix the value of j, and we show how to build an incremental data structure that decides in amortized $2^{O(d^2)} \log n$ time whether $\mathrm{FV}_\delta(i, j) = \emptyset$. To achieve this, we do not maintain the region $\mathrm{FV}_\delta(i, j)$ explicitly: we only maintain an auxiliary data structure that allows us to decide quickly whether it is empty or not. During the course of Algorithm *DecideFréchetMin*, the region $\mathrm{FV}_\delta(i, j)$ can be reset to $h_i^\delta \cap v_j$ at line 15, and otherwise, it is the intersection of $\mathrm{FV}_\delta(i-1, j)$ with h_i^δ. So at any time, we have $\mathrm{FV}_\delta(i, j) = h_{i_0}^\delta \cap h_{i_0+1}^\delta \cdots \cap h_i^\delta \cap v_j$ for some $1 \leqslant i_0 \leqslant i$.

So our auxiliary data structure needs to perform three types of operations:

1. Set $\mathcal{S} = \emptyset$.
2. Insert the next ball into $\mathcal{S}$.
3. Decide whether the intersection of the balls in $\mathcal{S}$ is empty.

When we run Algorithm *DecideFréchetMin* on column j, the sequence of n balls $h_1^\delta, \ldots, h_n^\delta$ is known in advance, but not the sequence of operations. So this is the assumption we make for our auxiliary data structure: we know in advance the sequence of balls, but the sequence of operations is given online. A trivial implementation using Lemma 2 requires $2^{O(d^2)}n$ time per operation. Using exponential and binary search [15], we will show how to do it in amortized $2^{O(d^2)} \log n$ time per operation.

Operation 1 is trivial to implement. To implement operation 2, suppose that, before we perform this operation, the cardinality $|\mathcal{S}|$ of $\mathcal{S}$ is $s = 2^\ell$, for some integer ℓ. Then, using Lemma 2, we check whether the intersection of the balls in $\mathcal{S}$ and the next s balls is empty. If so, we find by binary search the first subsequence of balls, starting at the balls of $\mathcal{S}$, whose intersection is empty. By Lemma 2, it can be done in $2^{O(d^2)}s \log s$ time. Then we can perform in constant time each operation of type 2 or 3 until the next time operation 1 is performed. On the other hand, if the intersection of the balls in $\mathcal{S}$ and the next s balls is not empty, we record this fact. Then, until the cardinality of S reaches $2s = 2^{\ell+1}$, or we perform operation 1, we can perform each operation of type 2 or 3 in constant time.

This data structure needs only amortized $2^{O(d^2)} \log n$ time per operation. Keeping one such data structure for each value of j, we can perform line 13 of Algorithm *DecideFréchetMin* in amortized $2^{O(d^2)} \log n$ time. Similarly, we can implement line 12 in amortized $2^{O(d^2)} \log m$ time. Overall, we obtain the following result:

Theorem 2. *In the d-dimensional Euclidean case, given $\delta \geqslant 0$, and given two imprecise sequences H and V of n and m points, respectively, we can decide in $2^{O(d^2)}mn \log(mn)$ time whether $\mathrm{F}^{\min}(H, V) \leq \delta$.*

3.3 Optimization Algorithms

In this section, we give optimization algorithms for computing the Fréchet distance lower bound in the orthogonal case, and in the Euclidean case. They are based on the decision algorithms of sections 3.1 and 3.2.

We first consider the orthogonal case. The result of the decision algorithm may only change at some value of δ such that a box $\mathrm{FH}_\delta(i, j)$ or $\mathrm{FV}_\delta(i, j)$ degenerates to a box of dimension less than d. It may happen when the sides of two boxes of type h_i^δ, h_i, v_j^δ, or v_j have a common supporting hyperplane. Therefore, if we denote by $(x_1, \ldots, x_d, y_1, \ldots, y_d)$ the coordinates of the box $[x_1, y_1] \times \cdots \times [x_d, y_d]$, and if we denote by $(c_1, \ldots, c_k)$ the sequence of all these coordinates in increasing order, the optimal value $\mathrm{F}^{\min}(H, V)$ has to be of the form $c_j - c_i$ or $(c_j - c_i)/2$ for some $i \leqslant j$. The matrix with coefficients $c_{ij} = \max\{0, c_j - c_{k+1-i}\}$ is a k-by-k monotone matrix with $k \leqslant dmn$, so using the technique by Frederickson

and Johnson [1,10] for searching in such a matrix, we can find $\mathrm{F}^{\min}(H, V)$ using $O(\log(dmn))$ calls to our decision algorithm. Thus, we obtained the following result:

Theorem 3. *In the d-dimensional orthogonal case, given two imprecise sequences H and V of n and m points, respectively, we can compute $\mathrm{F}^{\min}(H, V)$ in time $O(dmn \log(dmn))$.*

This approach does not work in the Euclidean case, so instead of using Frederickson and Johnson's technique, we use parametric search [1,2]. Using the algorithm from Theorem 2 both as the decision algorithm and the generic algorithm (without making it parallel), we obtain the following result:

Theorem 4. *In the d-dimensional Euclidean case, given two imprecise sequences H and V of n and m points, respectively, we can compute $\mathrm{F}^{\min}(H, V)$ in time $2^{O(d^2)} m^2 n^2 \log^2(mn)$.*

We can improve this result when $d = 2$. To achieve this, we apply parametric search in a different way. Observe that the result of Algorithm *DecideFréchetMin* only changes when there is a change in the combinatorial structure of the arrangement of the circles bounding the disks $h_i, h_i^\delta, v_j, v_j^\delta$ for all i, j. So, as a generic algorithm, we use an algorithm that computes the arrangement of these $2m + 2n$ circles. There exists such an algorithm with running time $O(\log(mn))$ using $O(m^2 + n^2)$ processors [2]. The decision algorithm is just our algorithm *DecideFréchetMin*, which runs in $O(mn \log(mn))$ time. So we need a total of $O((m^2 + n^2) \log(mn))$ time to run the generic algorithm, and a total of $O(mn \log^2(mn))$ time for the decision algorithm. Thus, we obtain the following result:

Theorem 5. *In the two-dimensional Euclidean case, given two imprecise sequences H and V of n and m points, respectively, we can compute $\mathrm{F}^{\min}(H, V)$ in $O(mn \log^2(mn) + (m^2 + n^2) \log(mn))$ time.*

4 Approximation Algorithms

The running time of our algorithm for computing $\mathrm{F}^{\min}$ exactly in the Euclidean case, when the dimension is larger than 2, may be too large for some applications. The situation is worse for the problem of computing $\mathrm{F}^{\max}$ since we currently do not even have a polynomial time algorithm. The problem of matching imprecise shapes with respect to the discrete Fréchet distance under translations seems even more complicated; in particular, we currently do not know how to solve it in polynomial time.

Definition 2. *For two region sequences H and V, the* smallest Fréchet distance lower bound under translation *is the minimum over all translations t of the Fréchet distance lower bound* [1] $\mathrm{F}^{\min}(H + t, V)$*:*

$$\mathrm{F}^{\min}_{\mathrm{tr}}(H, V) = \min_t \ \mathrm{F}^{\min}(H + t, V).$$

[1] For a translation t and a region sequence $H = h_1, \ldots, h_n$ we denote by $H + t$ the translate of H by t. Formally $H + t = h_1 \oplus t, \ldots, h_n \oplus t$ where $h_i \oplus t$ denotes the Minkowski sum of h_i and t, i.e., $h_i \oplus t = \{x + t \mid x \in h_i\}$.

The smallest Fréchet distance upper bound under translation *is the minimum over all translations* t *of the Fréchet distance upper bound* $\mathrm{F}^{\min}(H+t,V)$*:*

$$\mathrm{F}^{\max}_{\mathrm{tr}}(H,V) = \min_t \ \mathrm{F}^{\max}(H+t,V).$$

We obtained efficient algorithms to approximate $\mathrm{F}^{\min}$, $\mathrm{F}^{\max}$, $\mathrm{F}^{\min}_{\mathrm{tr}}$, and $\mathrm{F}^{\max}_{\mathrm{tr}}$ in arbitrary dimension d. Due to space limitation, we only state our results in this section, the proofs and the descriptions of the algorithms will be given in the full version of this paper.

As in the previous sections, we are given two input sequences H and V of n and m imprecise points, respectively, in d-dimensional space. In the Euclidean case, we use the Euclidean distance, and we assume that the imprecision regions h_i, v_j are Euclidean balls with centers a_i^0, b_j^0 and radius $0 < r_{\min} \le r(h_i), r(v_j) \le r_{\max}$. In the orthogonal case, we use the L_∞ distance, and the imprecision region h_i (resp. v_j) is an axis-parallel box that contains an L_∞ ball with radius $r_{\min}$ and center a_i^0 (resp. b_j^0), and is contained in a L_∞ ball with radius $r_{\max}$ and with the same center a_i^0 (resp. b_j^0). In both cases, we denote $A^0 = (a_1^0, \ldots, a_n^0)$ and $B^0 = (b_1^0, \ldots, b_m^0)$.

The approximation quality for $\mathrm{F}^{\max}_{\mathrm{tr}}$ and $\mathrm{F}^{\max}$ depends on the error parameters $r_{\min}$, $r_{\max}$. In particular we get constant factor approximations for the case $r_{\max} = \Theta(r_{\min})$, which seems to be a reasonable assumption in practice. We obtain the following result for approximating the Fréchet distance upper bound.

Theorem 6. *In dimension* d*, given two imprecise sequences* H *and* V *of* n *and* m *points, respectively, we can compute in* $O(dmn)$ *time a value* $\mathrm{APP}^{\max}(H,V)$ *such that*

$$\mathrm{F}^{\max}(H,V) \le \mathrm{APP}^{\max}(H,V) \le (1 + r_{\max}/r_{\min})\mathrm{F}^{\max}(H,V).$$

The proof is omitted due to space limitation. The idea is to place each point at the center of its region, and take $\mathrm{APP}^{\max}(H,V) = \mathrm{F}(A^0,B^0) + 2r_{\max}$.

The approximation quality for $\mathrm{F}^{\min}_{\mathrm{tr}}$ and $\mathrm{F}^{\min}$ depends on the error parameter $r_{\max}$ and an additional parameter measuring how well-separated any two *consecutive* points in an input sequence are:

Definition 3. *For a parameter* $\Delta_{\mathrm{sep}} > 0$*, we say that a region sequence* $H = h_1, \ldots, h_n$ *is* Δ_{sep}*-separated if* $\min_{x \in h_i, y \in h_{i+1}} \mathrm{dist}(x,y) \ge \Delta_{\mathrm{sep}}$ *for all* $1 \le i \le n-1$.

We get constant factor approximations for the case $\Delta_{\mathrm{sep}} = \Omega(r_{\max})$, which again seems to be a realistic assumption. In particular, we obtain the following result for approximating the Fréchet distance lower bound. The proof is omitted due to space limitation.

Theorem 7. *In dimension* d*, given two* Δ_{sep}*-separated region sequences* H *and* V *of* n *and* m *points, respectively, we can compute in* $O(dmn)$ *time a value* $\mathrm{APP}^{\min}(H,V)$ *such that*

$$\mathrm{F}^{\min}(H,V) \le \mathrm{APP}^{\min}(H,V) \le (1 + 8r_{\max}/\Delta_{\mathrm{sep}})\mathrm{F}^{\min}(H,V).$$

Finally, we obtain the results below for approximating the Fréchet distance lower and upper bounds under translation. Our algorithms run in $O(dmn)$ time, and we currently do not know if these values can be computed exactly in polynomial time. The proof is omitted due to space limitation.

Theorem 8. *In dimension d, given two imprecise sequences H and V of n and m points, respectively, we can compute in $O(dmn)$ time two values $\mathrm{APP}_{\mathrm{tr}}^{\max}(H,V)$ and $\mathrm{APP}_{\mathrm{tr}}^{\min}(H,V)$ such that*

(i) $\mathrm{F}_{\mathrm{tr}}^{\max}(H,V) \le \mathrm{APP}_{\mathrm{tr}}^{\max}(H,V) \le (2+3r_{\max}/r_{\min})\mathrm{F}_{\mathrm{tr}}^{\max}(H,V)$, *and*
(ii) $\mathrm{F}_{\mathrm{tr}}^{\min}(H,V) \le \mathrm{APP}_{\mathrm{tr}}^{\min}(H,V) \le (2+20r_{\max}/\Delta_{\mathrm{sep}})\mathrm{F}_{\mathrm{tr}}^{\min}(H,V)$.

5 Conclusion

In this paper, we gave an efficient algorithm for computing the Fréchet distance lower bound between two imprecise point sequences. We also gave efficient approximation algorithms for the Fréchet distance upper bound, and for the Fréchet distance upper bound and lower bound under translations.

Unfortunately, our dynamic programming approach for the Fréchet distance lower bound does not seem to apply to the Fréchet distance upper bound. So we currently do not have a polynomial-time algorithm for computing the exact Fréchet distance upper bound. This problem may be hard, as it sometimes happens that a maximization problem for imprecise points is much harder than the corresponding minimization problem. For instance, Löffler and Van Kreveld [13] showed that computing the maximum area or perimeter of the convex hull of n imprecise points is NP-hard, even though the corresponding minimization problems can be solved in $O(n^2)$ and $O(n \log n)$ time respectively. Thus, it would be interesting to show that the exact Fréchet distance upper bound problem is NP-hard, or to find a polynomial-time algorithm.

Acknowledgements

Work by Ahn was supported by the Korea Research Foundation Grant funded by the Korean Government(KRF-2008-614-D00008). Work by Knauer and Scherfenberg was supported by the German Science Foundation (DFG) under grant Al 253/5-3. Work by Schlipf was supported by the Deutsche Forschungsgemeinschaft within the research training group 'Methods for Discrete Structures'(GRK 1408).

References

1. Agarwal, P.K., Sharir, M.: Efficient algorithms for geometric optimization. Computing Surveys 30(4), 412–458 (1998)
2. Agarwal, P.K., Sharir, M., Toledo, S.: Applications of parametric searching in geometric optimization. J. Algorithms 17(3), 292–318 (1994)
3. Alt, H., Godau, M.: Computing the Fréchet distance between two polygonal curves. International Journal of Computational Geometry and Applications 5, 75–91 (1995)

4. Alt, H., Knauer, C., Wenk, C.: Matching polygonal curves with respect to the Fréchet distance. In: Ferreira, A., Reichel, H. (eds.) STACS 2001. LNCS, vol. 2010, pp. 63–74. Springer, Heidelberg (2001)
5. Alt, H., Knauer, C., Wenk, C.: Comparison of distance measures for planar curves. Algorithmica 38(1), 45–58 (2003)
6. Aronov, B., Har-Peled, S., Knauer, C., Wang, Y., Wenk, C.: Fréchet distance for curves, revisited. In: Azar, Y., Erlebach, T. (eds.) ESA 2006. LNCS, vol. 4168, pp. 52–63. Springer, Heidelberg (2006)
7. Dyer, M.E.: A class of convex programs with applications to computational geometry. In: Proc. 8th Symposium on Computational Geometry, pp. 9–15. ACM, New York (1992)
8. Edelsbrunner, H.: Geometry and Topology for Mesh Generation. Cambridge University Press, Cambridge (2001)
9. Eiter, T., Mannila, H.: Computing discrete Fréchet distance. Tech. Rep. CD-TR 94/64, Christian Doppler Laboratory for Expert Systems, TU Vienna, Austria (1994)
10. Frederickson, G.N., Johnson, D.B.: Generalized selection and ranking: Sorted matrices. SIAM Journal on Computing 13(1), 14–30 (1984)
11. Khanban, A.A., Edalat, A.: Computing Delaunay triangulation with imprecise input data. In: Proc. 15th Canadian Conference on Computational Geometry, pp. 94–97 (2003)
12. Knauer, C., Löffler, M., Scherfenberg, M., Wolle, T.: The directed Hausdorff distance between imprecise point sets. In: ISAAC. LNCS, vol. 5878, pp. 720–729. Springer, Heidelberg (2009)
13. Löffler, M., van Kreveld, M.J.: Largest and smallest tours and convex hulls for imprecise points. In: Arge, L., Freivalds, R. (eds.) SWAT 2006. LNCS, vol. 4059, pp. 375–387. Springer, Heidelberg (2006)
14. Löffler, M., Snoeyink, J.: Delaunay triangulation of imprecise points in linear time after preprocessing. Computational Geometry: Theory and Applications 43(3), 234–242 (2010)
15. Moffat, A., Turpin, A.: Compression and Coding Algorithms. Kluwer, Dordrecht (2002)
16. Rote, G.: Computing the Fréchet distance between piecewise smooth curves. Computational Geometry: Theory and Applications 37(3), 162–174 (2007)
17. Sember, J., Evans, W.: Guaranteed Voronoi diagrams of uncertain sites. In: Proc. 20th Annual Canadian Conference on Computational Geometry (2008)