Fast Algorithms for Fréchet-Based Similarity

ZIJIN HUANG

Supervisor: Joachim Gudmundsson
Associate Supervisor: André van Renssen

A thesis submitted in fulfilment of
the requirements for the degree of
Doctor of Philosophy

School of Computer Science
Faculty of Engineering
The University of Sydney
Australia

1 May 2026

Abstract

Polygonal curves arise naturally when recording movement data from GPS traces, animal tracking, or eye-gaze measurements. A fundamental task is to compare such curves using a formal notion of similarity. The Fréchet distance is a widely used metric for this purpose, as it respects both the geometric proximity and the traversal order of two curves. This thesis develops faster algorithms for four Fréchet-based similarity problems, each using the freespace diagram as its central tool.

In Chapter 2, we study the Fréchet distance under geometric transformations, which is essential when comparing shapes regardless of their position or orientation. We present the first improvement to the decision problem for a wide class of transformations, including translations, rotations, and scalings.

In Chapter 3, we consider the Fréchet edit distance, which allows inserting or deleting vertices before measuring the continuous Fréchet distance. This variant is motivated by the need for robustness against outliers and noise in real-world trajectory data. We provide faster algorithms for several edit models, improving on prior bounds by up to a factor of $k \cdot n$.

In Chapter 4, we study subtrajectory clustering on c-packed trajectories, a realistic model of movement data with bounded self-overlap. We present a near-linear time approximation algorithm for the subtrajectory cluster problem, circumventing the conditional cubic lower bound for general curves.

In Chapter 5, we connect the partial weak Fréchet similarity to shortest paths in weighted planar regions. We construct a near-linear size approximate spanner for the $0/1/\infty$ weighted region problem, and apply it to obtain an approximation algorithm for the partial weak Fréchet similarity.

Statement of Attribution

The contents of this thesis are based on three papers and one unpublished manuscript. I was the corresponding author and the main contributor on all four papers. As per convention in theoretical computer science, authors are listed alphabetically.

Kevin Buchin, Maike Buchin, Zijin Huang, André Nusser, Sampson Wong. *Faster Fréchet distance under transformations.* In Proceedings of the 52nd EATCS International Colloquium on Automata, Languages, and Programming, ICALP 2025.

Joachim Gudmundsson, Zijin Huang, André van Renssen, Sampson Wong. *Faster Fréchet Edit Distance.* Under submission.

Joachim Gudmundsson, Zijin Huang, André van Renssen, Sampson Wong. *Computing a Subtrajectory Cluster from c-packed Trajectories.* In the 34th International Symposium on Algorithms and Computation, ISAAC 2023.

Joachim Gudmundsson, Zijin Huang, André van Renssen, Sampson Wong. *Spanner for the $0/1/\infty$ Weighted Region Problem.* In the 19th Algorithms and Data Structures Symposium, WADS 2025.

Zijin Huang

As supervisor for the candidature upon which this thesis is based, I can confirm that the authorship attribution statements above are correct.

Joachim Gudmundsson

Statement of Attribution of Generative AI

During the preparation of the thesis the author used ChatGPT and Claude for the purposes of text enhancement. The use of these generative AI tools includes paraphrasing, improving sentence structure, refining grammar and spelling, and enhancing clarity of academic writing. The author confirms that where text was modified by generative AI, the content was reviewed for possible errors, inaccuracies, and bias. The author takes full responsibility for the submitted thesis and ensures the work is their own and has used generative AI within the parameters of use (refer to the University of Sydney generative AI guide for researchers).

Zijin Huang

Statement of Originality

This is to certify that to the best of my knowledge, the content of this thesis is my own work. This thesis has not been submitted for any degree or other purposes. I certify that the intellectual content of this thesis is the product of my own work and that all the assistance received in preparing this thesis and sources have been acknowledged.

Zijin Huang

Funding

This research was supported by the Faculty of Engineering Research Stipend Scholarship at the University of Sydney.

Acknowledgements

It used to be that highly polished writing is a means to show one's seriousness, but today, when an LLM writes with absolute perfection, it seems only a flawed, clumsy and cringe writing such as this acknowledgement can show my sincerity — I was helped by so many, and I am so grateful.

They say only few times in a man's life does he meet a mentor that alters his trajectory for the better. I find this fitting because that mentor of mine is Joachim Gudmundsson and we worked on...trajectories! I thank Joachim wholeheartedly, for being the awesome supervisor he is. I was never a smart student, but Joachim managed to kindle whatever little there was into someone who can do a bit of research, with his ingenuity, his persistent presence, and his patience — my my did he need patience with me. He paid attention to and nurtured my passion for research and my curiosity by encouraging me to work on problems that interest me. Meanwhile, he also taught me that passion itself is insufficient — careful examination is also required. He pushed me to read more papers and be more careful with proofs, not just by telling me, but by acting as an example. Be passionate but verify rigorously — Joachim never said those words, but he is their embodiment, and I aspire to spend the rest of my life trying to be like him — minus the golf addiction, of course.

Besides Joachim, I was fortunate enough to have two more remarkable individuals I regard as mentors: André van Renssen and Sampson Wong. André was the first person I did research with, and he was the first to show me that doing research in a supportive environment can be fun. During my PhD study, André was always there to offer guidance and answer my questions. He read countless drafts of mine, battling through my proofs by pictures, "it is easy to see ...", and "refer to the picture two sections later". But he never faltered and always nudged me to be better little by little. Sampson was an exemplar PhD student, and I have learned so much from him. Even after he decided to move from the warm and sunny Sydney to the cold and sunless Copenhagen, for whatever reasons, he would still find time

to work with me, and his 8am and my 6pm somehow gave us enough time to do something meaningful.

Besides my mentors, I was surrounded by wonderful people along this journey. I thank John Stavrakakis, who gave me my first job as an academic tutor. I was clearly not the best candidate, as English is my second language and the ability to explain difficult concepts in English does not come naturally to me. I consider this job as a key building block leading to my PhD studies, and I thank John for this wonderful opportunity.

I thank the people in the algorithm group for a nurturing environment: Clément, Lindsey, Joy, Julián, Abigail, Sasha, Tony, and Aditya. The reading groups, the seminars, the day-to-day chats, and the advice I received in this environment are far beyond any expectation I may have about a good research group. I thank the people I visited in Dortmund who opened their welcoming arms to me: Kevin, Antonia, Mart, Maike, and Lucas.

There are those who taught me and there are those who nurtured my heart and spirit. I thank my two cats: 小虎 and 小斗儿. I cherish my memories with you, and you will always occupy a piece of my mind. The times we shared were the most valuable memories I have, and it seems only fitting to immortalise you together with this thesis.

I thank my lovely partner Jiayi. You brightened my days and nurtured my soul when I needed it the most. It seems odd to observe both tenderness and tenacity in someone. But you are that person — tenderness for me and tenacious within. I find both comfort and inspiration from you, and I am grateful for your companionship along this journey.

我最感谢，最无言以谢，也是最无需言谢的人是我的母亲。是妈妈鼓励我留学，读博，以及放弃短期利益，永远做最难最有收获的事情。是她支持我不实际的梦想。我人生中最大的目标就是把你给我的一切也给我的孩子。永远爱你老妈！

Finally, looking back to my journey through my PhD, I realise that, for me, the PhD studies were not about doing research or publishing papers. It is a journey of becoming, to shake off pride, self-abasement, feelings of adequacy and inadequacy, to simply be, pursue, evolve in spiral, to learn and to unlearn. Therefore, I bow to my past selves. You read the papers, did the research, and wrote the papers. You endured your own doubts and shook off the consuming thoughts. You savoured your victories and drew lessons from them. You looked inward and grew from what you found — all to become the next version of yourselves,

and eventually, who I am today. I promise to put in as much effort as you did — to live, to achieve, to make mistakes, to shed traits, and to gain new ones — so that our future selves can say what I have in mind today: *what a ride*.

Contents

List of Figures

CHAPTER 1

Introduction

Computational Geometry

Computational geometry is the algorithmic study of geometric problems: given a description of geometric objects such as points, lines, curves, or polytopes, the goal is to design efficient algorithms and data structures to answer combinatorial and metric questions about them. The field emerged in the late 1970s and early 1980s, catalysed by the doctoral thesis of Shamos [131] and the foundational textbook of Preparata and Shamos [129]. Since then, computational geometry has matured into a rich discipline with deep theoretical foundations and broad practical impact, as witnessed by the comprehensive textbook of de Berg, Cheong, van Kreveld, and Overmars [15], the monograph of Edelsbrunner [68], and the Handbook of Discrete and Computational Geometry [85].

The field is organised around several core problem families. These include computing convex hulls, constructing Voronoi diagrams [78], analysing arrangements of curves and surfaces [100], computing triangulations, finding shortest paths in geometric domains [104, 125], range searching and orthogonal queries [1, 50], and detecting intersections among geometric objects [15]. Each of these areas has developed a substantial algorithmic toolkit that later chapters of this thesis draw upon.

These foundational problems find applications across computer science and engineering. Computer graphics relies on triangulations and visibility computations; robotics uses motion-planning algorithms built on configuration-space constructions [111]; geographic information

systems (GIS) depend on efficient spatial indexing and map overlay operations [15]; and network design benefits from geometric spanners that guarantee near-shortest-path routing [126]. Beyond these classical domains, autonomous vehicles rely on LiDAR sensors that generate millions of points per second, demanding real-time perception and planning algorithms that build on geometric data structures such as k-d trees and range trees [122]; VLSI chip design places and routes wires across layouts with billions of transistors using partitioning, placement, and routing algorithms that operate on geometric representations [110]; and medical imaging reconstructs three-dimensional organ models from CT and MRI scans via mesh generation and surface triangulation [123]. In each of these settings, the sheer scale of modern data sets—point clouds with billions of points, circuits with billions of components—means that even modest improvements in algorithmic complexity translate directly into the feasibility of real-time or overnight computation.

When exact solutions are too costly, approximation algorithms play a central role. Techniques such as coresets, ε-nets, and random sampling [60, 101] yield provably good approximate solutions in significantly less time. On the lower-bound side, conditional hardness results—most notably reductions from the 3SUM problem [81]—provide evidence that many natural geometric problems cannot be solved substantially faster than by known algorithms.

One prominent family of geometric problems concerns the comparison of shapes. Alt and Guibas [8] survey the algorithmic landscape of shape matching, interpolation, and approximation, and Alt [6] discusses the computational geometry of comparing shapes more broadly. Among shape-comparison problems, measuring the similarity of curves is particularly challenging because it requires a notion of distance that respects the sequential structure of the curves [138]. This requirement leads naturally to the Fréchet distance, the central object of this thesis.

Movement Analysis

The proliferation of positioning technologies—GPS receivers, animal-borne tracking devices, inertial measurement units, sports sensors, and eye-tracking hardware—has made movement

data one of the most abundant and scientifically valuable data types of the 21st century. A moving entity traces a trajectory in space over time, and when sampled at discrete time steps, this trajectory is naturally represented as a polygonal curve. Computational movement analysis [86, 142] studies algorithms and data structures for extracting meaningful patterns and answering queries on such data.

Trajectories may be recorded as sequences of time-stamped positions (point trajectories) or modelled as continuous curves interpolating these positions. Fundamental analysis tasks include segmentation [46], where a trajectory is partitioned into behaviourally homogeneous pieces; clustering, where groups of similar (sub)trajectories are identified [2, 36]; map matching, where a noisy GPS trace is aligned to a road network [52, 88]; and anomaly detection, where unusual movement patterns are flagged [136].

Movement analysis finds applications across diverse domains. In transportation, trajectory data reveals commuting patterns, identifies congestion, and supports urban planning [36, 107]. In ecology, GPS-tagged animals provide insights into migration routes, habitat usage, and inter-species interactions [113]. Sports analytics tracks player movements to evaluate strategy and physical performance [130]. Eye-tracking studies use gaze trajectories recorded by XR headsets and screen-based trackers for applications in education [114], user-experience research [108], and three-dimensional scene analysis [63]. Maritime vessel tracking via Automatic Identification System (AIS) data supports route prediction, anomaly detection, and maritime situational awareness [128]. Air-traffic management depends on accurate trajectory prediction to inform conflict-detection systems that maintain safe separation between aircraft [13]. Crowd simulation reconstructs realistic pedestrian flows from observed trajectory data and underpins tools for evacuation planning and public-space design [121]. These domains share a common computational bottleneck: data sets routinely contain millions of trajectories, and pairwise similarity queries scale quadratically in the number of curves and at least quadratically in curve complexity, making efficient algorithms essential for any analysis at scale.

Fréchet distance

Across all these tasks, measuring how similar two trajectories are is a recurring and central challenge. Curve similarity is central to applications that require matching, clustering, or classifying trajectories. Examples include map matching, detecting entities that move together, and comparing eye-movement traces. Similarity measures are also useful as a bridge between curve problems and better-studied point-set problems.

The study of trajectory similarity thus sits at the intersection of computational geometry and computational movement analysis. Similarity is a geometric distance measure whose efficient computation requires algorithmic techniques from computational geometry, and whose primary motivation comes from the need to compare trajectories in movement data.

But what does it mean for two curves to be similar? The notion of *similarity* is inherently subjective, and we need a formal definition of when two curves are similar (as opposed to dissimilar). One simple approach is to ignore the order along the curve and treat it as the set of points it traces in space. This leads to the (undirected) Hausdorff distance, introduced by Hausdorff [103] and widely used in geometric shape matching [139].

In the Hausdorff distance, for every point on a curve π we find the closest point on σ; the maximum of these closest-point distances gives the directed distance from π to σ. The (undirected) Hausdorff distance takes the maximum of the two directed distances, so that both curves must lie close to each other. This makes the Hausdorff distance a good fit when we only care about geometric proximity (*do the curves lie near the same locations?*) and not about synchronising traversal along the curves.

However, in settings where we care about whether two entities move together over time, the Hausdorff distance is not a suitable measure of similarity. Two curves can lie in close proximity, yet the entities that generated them may follow very different traversal patterns (see Figure 1.1).

The Fréchet distance, by contrast, respects both the geometry and the monotone traversal of continuous curves, making it a natural and widely adopted similarity measure. Maurice

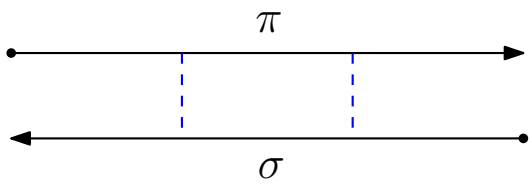

FIGURE 1.1: Two curves that stay close as point sets (small Hausdorff distance) but differ in their movement pattern.

Fréchet introduced this distance in 1906 [80]. In the classical analogy, a person walks along the curve π and a dog walks along the curve σ. Both must move forward, but they may change speed or pause. The Fréchet distance is the minimum leash length needed over all such walks (see Figure 1.2). Its applications range from online handwriting recognition [133] and trademark shape matching [10] to protein structure alignment [109]. In the remainder of this chapter, we formalise the Fréchet distance, introduce the freespace diagram that underpins its computation, and outline the contributions of this thesis.

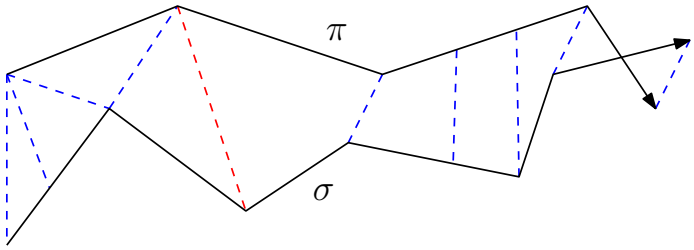

FIGURE 1.2: Intuition for the Fréchet distance as a leash length between two traversals. The red line indicates the longest leash induced by two traversals.

For our purpose, a d-dimensional *point* $p \in \mathbb{R}^d$ is a tuple that represents a location in $\mathbb{R}^d$, and a *segment* pq is a closed straight-line connection between the point p and the point q. A d-dimensional polygonal curve (curve for short) $\pi = \langle \pi_1, \ldots, \pi_n \rangle$ is the union $\{\pi_i\pi_{i+1} \mid i = 1, \ldots, n-1\}$ of segments (see Figure 1.3), and for every $j = 1, \ldots, n$, π_j is a *vertex* of π. The size $|\pi|$ of π is its number of vertices, and the length $\|\pi\|$ of π is the total length of its segments.

Sometimes, it is convenient to describe the curve $\pi : [1, n] \mapsto \mathbb{R}^d$ as a bijective and continuous function that, for every $i = 1, \ldots, n-1$, maps the unit interval $[i, i+1]$ to the segment $\pi_i\pi_{i+1}$. Both representations $\langle \pi_1, \ldots, \pi_n \rangle$ and $\pi : [1, n] \mapsto \mathbb{R}^d$ describe an orientation of the curve that starts at vertex π_1 and ends at vertex π_n. For the rest of the section, a curve is a function.

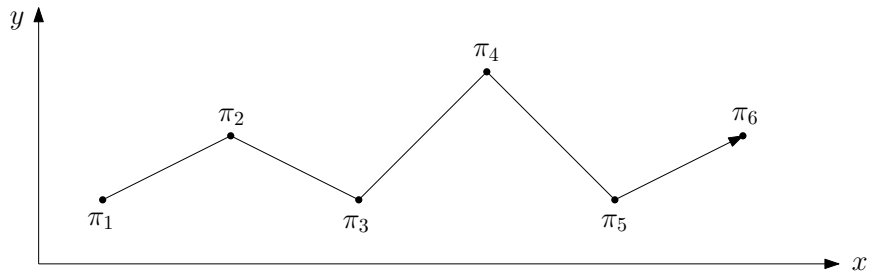

FIGURE 1.3: A curve with six vertices in $\mathbb{R}^2$.

Consider a function f that describes how the person walks on the curve; for some time step $t \in [0, 1]$, $f(t)$ is a number in the range $[1, n]$ and $\pi(f(t))$ is a point indicating the position of the person on the curve π. The person needs to start at the first vertex of the curve and end at the last vertex; therefore $f(0) = 1$ and $f(1) = n$. The person needs to move forward, therefore f must be non-decreasing. The person cannot jump from one point of the curve to another, therefore f must be continuous. Such a continuous and non-decreasing function $f : [0, 1] \mapsto [1, n]$ is called a *reparameterisation*. Analogously, let $g : [0, 1] \mapsto [1, m]$ be a reparameterisation that describes how the dog walks on σ.

For a fixed time $t \in [0, 1]$, in order for the person to be at point $\pi(f(t))$ and the dog to be at $\sigma(g(t))$, the leash needs to be at least as long as the Euclidean distance $\|\pi(f(t)) - \sigma(g(t))\|$ from $\pi(f(t))$ to $\sigma(g(t))$. To complete the walk according to f and g, the required leash length is the maximum distance between the two positions over all times $t \in [0, 1]$.

$$\text{maxLeashLength}(f, g, \pi, \sigma) = \max_{t \in [0,1]} \|\pi(f(t)) - \sigma(g(t))\|$$

The standard *Fréchet distance* $d_{\mathcal{F}}(\pi, \sigma)$ is the minimum leash length required over all possible walks defined by all valid reparameterisations f and g.

$$d_{\mathcal{F}}(\pi, \sigma) = \inf_{\substack{f:[0,1]\mapsto[1,n] \\ g:[0,1]\mapsto[1,m]}} \text{maxLeashLength}(f, g, \pi, \sigma)$$

Two widely used variants are the *weak Fréchet distance*, which relaxes monotonicity by only requiring the reparameterisations to be continuous (allowing backtracking), and the *discrete Fréchet distance*, which compares the vertex sequences of polygonal curves via monotone

couplings and takes the minimum possible maximum vertex distance [71, 102]. In this thesis, we focus on the standard (continuous, monotone) Fréchet distance unless stated otherwise.

Despite the elegance and intuitiveness of the Fréchet distance, it was not until 1995 that Alt and Godau [7] proposed the first polynomial time algorithm that computes the Fréchet distance. Their main contribution was to introduce the notion of the freespace and the freespace diagram. The *δ-freespace* $\mathcal{F}_\delta(\pi, \sigma)$ describes all pairs of points, one from π and one from σ, that are within Euclidean distance δ.

$$\mathcal{F}_\delta(\pi, \sigma) = \{(x, y) \in [1, n] \times [1, m] \mid \|\pi(x) - \sigma(y)\| \leq \delta\}$$

Throughout the remainder of this thesis, when the threshold δ is clear from context, we refer to the δ-freespace and the δ-freespace diagram (introduced later) simply as the freespace and the freespace diagram, respectively. Why do we care about the freespace? The key observation is that an xy-monotone path through $\mathcal{F}_\delta(\pi, \sigma)$ from $(1, 1)$ to (n, m) corresponds to a valid pair of reparameterisations with leash length at most δ. Therefore, the decision problem *is* $d_{\mathcal{F}}(\pi, \sigma) \leq \delta$*?* reduces to deciding whether such a path exists, which is illustrated in Figure 1.4.

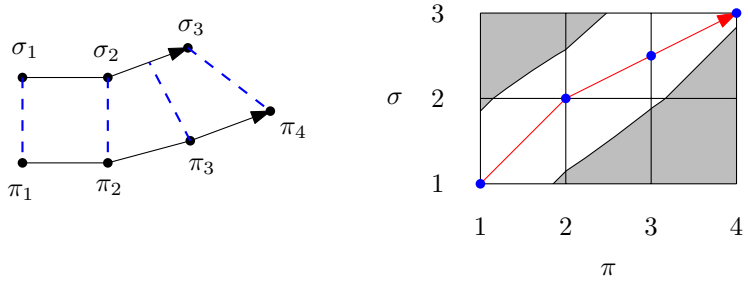

FIGURE 1.4: Two polygonal curves π and σ with their δ-freespace diagram $\mathcal{D}_\delta(\pi, \sigma)$. A red xy-monotone path represents a continuous matching: each point on the path corresponds to a pair of points on the curves within distance δ.

The freespace $\mathcal{F}_\delta(\pi, \sigma)$ can be partitioned into cells; this partition yields the δ-freespace diagram $\mathcal{D}_\delta(\pi, \sigma)$. For $i = 1, \ldots, n-1$ and $j = 1, \ldots, m-1$, each cell $C_{i,j}$ stores the freespace $\mathcal{F}_\delta(\pi_i\pi_{i+1}, \sigma_j\sigma_{j+1})$ induced by the corresponding pair of segments. Alt and Godau [7] observed that the freespace of two segments is the intersection of an ellipse with a rectangle;

consequently, the freespace in every cell has this form. We summarise the key properties of the freespace and the freespace diagram in Fact 1.

FACT 1. *Let π and σ be curves in $\mathbb{R}^d$ with $|\pi| = n$ and $|\sigma| = m$. Then:*

(1) *The freespace diagram $\mathcal{D}_\delta(\pi, \sigma)$ contains $\mathcal{O}(mn)$ cells and can be constructed in $\mathcal{O}(dmn)$ time.*
(2) *The freespace within each cell is convex and has constant description complexity.*
(3) *$d_{\mathcal{F}}(\pi, \sigma) \leq \delta$ if and only if there exists an xy-monotone path in the freespace $\mathcal{F}_\delta(\pi, \sigma)$ from $(1, 1)$ to (n, m).*

The freespace diagram is the core tool for many Fréchet-based problems, because it turns geometric questions about curve similarity into reachability questions in a structured parameter space. Beyond the original decision algorithm of Alt and Godau [7], it is used in algorithms for Fréchet distance with shortcuts [62]. It also underpins approximation schemes for realistic input models like c-packed curves [65], where reachability can be decided in a sparsified freespace. We use this standard version of the freespace diagram in Chapters 2 and 3.

However, not all problems can be handled if we parameterise each edge by unit length and then build the freespace diagram, which makes every cell a square and discards segment lengths. When the length of subtrajectories matters, this loss of information is fatal: subtrajectory clustering imposes a minimum length constraint on the reference subtrajectory [36, 95], partial curve matching [31, 48] depends on the lengths of matched subcurves, and Fréchet distance with speed limits constrains motion by arc length [124]. These settings therefore require a length-preserving freespace and freespace diagram with rectangular cells that reflect the true segment lengths. We use this version in Chapters 4 and 5.

$$\mathcal{F}_\delta(\pi, \sigma) = \{(x, y) \in [0, \|\pi\|] \times [0, \|\sigma\|] \mid \|\pi(x) - \sigma(y)\| \leq \delta\}$$

Contributions

The remainder of the thesis applies the notions introduced here—curves and the freespace diagram—to several Fréchet-based problems. Each chapter focuses on a concrete problem setting and a specific algorithmic improvement. We defer the literature reviews of related problems to individual chapters.

Chapter 2 (Fréchet distance under transformations). We study the decision problem: given two curves π, σ and a threshold δ, decide whether there exists a transformation τ (translation, rotation, scaling, or more general affine maps) such that $d_{\mathcal{F}}(\pi, \tau(\sigma)) \leq \delta$. The chapter first handles translations in $\mathbb{R}^2$, improving the best known running time from $\mathcal{O}(n^8)$ to $\widetilde{\mathcal{O}}(n^{7+\frac{1}{3}})$ by maintaining reachability in related freespace structures while traversing the transformation arrangement. We then generalise the method to any class of rationally parameterised transformations with k degrees of freedom, achieving $\widetilde{\mathcal{O}}(n^{3k+\frac{4}{3}})$ time, and discuss how the freespace diagram changes under these transformations.

Chapter 3 (Fréchet edit distance). We consider the Fréchet edit distance (FED) introduced by Fox et al. [79], where one may delete and/or insert vertices before measuring the continuous Fréchet distance. The decision problem asks whether at most k edits suffice to bring the Fréchet distance below a given threshold. We provide improved algorithms for the main edit models: (i) deletion-only on one or both curves in $\widetilde{\mathcal{O}}(kn^2)$ time with a near-matching lower bound, (ii) insertion-only on one or both curves in $\widetilde{\mathcal{O}}(kn^3)$ time, and (iii) mixed insertions and deletions on a single curve in $\widetilde{\mathcal{O}}(k^2n^3)$ time. These results improve prior bounds by up to a factor of $k \cdot n$.

Chapter 4 (Subtrajectory cluster on c-packed curves). We address the subtrajectory cluster (SC) problem: given a trajectory π, parameters m, δ, l, and approximation factor ε, determine whether there exist at least m non-overlapping subtrajectories whose Fréchet distances are at most $(1+\varepsilon)\delta$ from one reference subtrajectory of length at least l. For general curves, SC has cubic-time barriers; we circumvent these by restricting to c-packed trajectories and designing a simplified freespace diagram that preserves subtrajectory lengths

while keeping the number of non-empty cells near-linear. This yields a near-linear time $(1+\varepsilon)$-approximation algorithm with running time $\mathcal{O}((c^2 n/\varepsilon^2)\log(c/\varepsilon)\log(n/\varepsilon))$, and a simplified freespace construction that may be useful beyond SC.

Chapter 5 ($0/1/\infty$ weighted regions and weak Fréchet similarity). We study the weighted shortest path problem in planar subdivisions where faces have weights in $\{0, 1, \infty\}$ (free regions, unit-cost plane, and obstacles). We build a $(1+\varepsilon)$-approximate spanner for the vertices of such regions in expected near-linear time, enabling fast approximate shortest paths between arbitrary points. By reducing partial weak Fréchet similarity [31] to such weighted shortest path instances, this yields a near-quadratic time approximation for partial weak Fréchet similarity, which is close to the best possible under known lower bounds.

CHAPTER 2

Faster Fréchet Distance under Transformations

2.1 Introduction

Many applications require determining the similarity of two geometric shapes, disregarding their location or orientation. More specifically, *shape matching* asks for the distance between two shapes if we allow the shapes to be transformed to minimise their distance [6, 8, 138]. Transformations may include translations, rotations, scaling, or a combination thereof. In this chapter, we focus on polygonal curves under the Fréchet distance. Curves occur in many applications and need to be matched whenever they describe a local pattern, for example to recognise handwritten characters [133], trademarks [10], or movement patterns [36, 95, 96]. The Fréchet distance is arguably the most popular distance measure for curves in computational geometry. There has been significant algorithmic progress on the Fréchet distance, for instance, most recently on approximation [61, 105, 106], data structures [11, 29, 41, 45, 74, 76, 89, 97], algorithm engineering [14, 27, 28, 37, 67], simplification [24, 54, 115, 118], clustering [30, 33, 39, 43, 95, 127], Fréchet variants [35, 38, 42, 66, 73, 77, 79], and its complexity in general [18, 32, 55, 56].

The Fréchet distance under translation is defined as the minimum Fréchet distance for any translation of the curves. Computing the Fréchet distance under translations was first studied by Efrat, Indyk and Venkatasubramanian [70] and by Alt, Knauer and Wenk [9, 117, 141]. For two curves of total complexity n in $\mathbb{R}^2$, an $\widetilde{\mathcal{O}}(n^{10})$ time algorithm[1] for the Fréchet distance under translation was presented in [70], and an $\widetilde{\mathcal{O}}(n^8)$ time algorithm was presented in [9]. Wenk [141] generalised the approach in [9] to higher dimensions and a wide range

[1] By $\widetilde{\mathcal{O}}(\cdot)$ we hide (poly-)logarithmic factors in n.

of other transformations, e.g., for rotations or scalings in 2D in $\widetilde{\mathcal{O}}(n^5)$ time, or for affine transformations in d dimensions in $\widetilde{\mathcal{O}}(n^{3(d^2+d)+2})$ time.

Despite significant algorithmic progress on various aspects of the Fréchet distance, no faster algorithms for computing the (continuous) Fréchet distance under transformations have been developed until now. In contrast, for computing the discrete Fréchet distance under translation, first an $\widetilde{\mathcal{O}}(n^6)$-time algorithm [109] was developed, then an $\widetilde{\mathcal{O}}(n^5)$-time algorithm [12], and more recently, an $\widetilde{\mathcal{O}}(n^{4+\frac{2}{3}})$-time algorithm [26].

Our Contribution. In this chapter, we present the first progress on the Fréchet distance under transformations since its introduction [9, 70, 117, 141]. Similar to the algorithm in [9, 141], our algorithm can be used for a wide range of classes of transformations. Specifically, given two curves π and σ of total complexity n and a threshold $\delta \geq 0$, we want to determine whether there is a transformation τ from a given class of transformations, such that the Fréchet distance between π and $\tau(\sigma)$ is at most δ. Our algorithm improves the running time for translations in two dimensions from $\mathcal{O}(n^8)$ to $\widetilde{\mathcal{O}}(n^{7+\frac{1}{3}})$.

THEOREM 1. *The Fréchet distance under translation in $\mathbb{R}^2$ can be decided in $\mathcal{O}(n^{7+\frac{1}{3}} \log^2 n)$ time.*

More generally, we improve the running time for the various classes of transformations (and dimensions) given by Wenk [141] by roughly a factor of $n^{2/3}$. For example, for rotations or scalings in $\mathbb{R}^2$, our algorithm runs in $\widetilde{\mathcal{O}}(n^{4+\frac{1}{3}})$ time and for affine transformations in d dimensions in $\widetilde{\mathcal{O}}(n^{3(d^2+d)+\frac{4}{3}})$ time.

THEOREM 2. *The Fréchet distance under transformations rationally represented with k degrees of freedom can be decided in $\mathcal{O}(n^{3k+4/3} \log^2 n)$ time.*

We first present our approach for the special case of translations in two dimensions. Then we generalise our approach to other transformations and higher dimensions. Similarly to the approach in [9], we compute an arrangement in the space of transformations, which in the case of 2D translations has complexity $\mathcal{O}(n^6)$. In [9], the Fréchet distance is computed for each

face of this arrangement in $\mathcal{O}(n^2)$ time per face by using the *freespace diagram*, which results in an overall running time of $\mathcal{O}(n^8)$. In our approach, we instead traverse the arrangement, making use of the fact that the freespace diagram for adjacent faces in the arrangement is similar. The challenge with using this approach is that it requires a dynamic data structure for maintaining reachability in the freespace diagram. However, already on directed graphs this is a difficult problem that to the best of our knowledge mostly saw progress on planar graphs [64, 112, 134] or more specifically grid graphs [12, 26]. In [26], a data structure with efficient updates and queries for reachability in a dynamic directed grid graph is presented, which is used to give a faster algorithm for the *discrete* Fréchet distance under translation. However, while the discrete Fréchet distance naturally reduces to a reachability problem on such a grid, this is not the case for the (continuous) Fréchet distance.

To that end, we present a detailed analysis of the changes in the freespace diagram when traversing the arrangement in the space of translations, which we call *events* (see Section 2.3). We then show how to construct a (directed) *freespace graph* from the freespace diagram, and analyse how the freespace graph changes for the various types of events. Then, we show how each of these events can be modelled as changes in a suitable grid graph, which allows us to utilise the data structure in [26] to maintain reachability in this graph (see Section 2.4). The main challenge here is that an event may cause rows or columns in the freespace graph to swap, appear or disappear, which are not operations that can be handled by the previous data structure of [26]. Using our approach, we show how to handle these row or column events using a linear number of updates to the grid graph per event. Finally, we use an amortised analysis to show that each event only requires a constant number of updates to the grid graph per event, since expensive events do not happen too often.

We note that in an independent work, progress was made on the one-dimensional Fréchet distance under translation or scaling [19]. In that work, the authors present an $\widetilde{\mathcal{O}}(n^{8/3})$ algorithm for both problems, making use of the offline dynamic data structure of [26].

2.2 Preliminaries

A d-dimensional polygonal curve π is a piecewise linear curve represented as a continuous mapping from $[1, n]$ to $\mathbb{R}^d$. For integer i, $\pi_i = \pi[i]$ is a vertex, and $\overline{\pi_i} = \pi_i\pi_{i+1} = \pi[i, i + 1]$ is a segment.

The Fréchet distance is a popular measure of the similarity between two polygonal curves. An *orientation-preserving reparameterisation* is a continuous and bijective function $f : [0, 1] \to [0, 1]$ such that $f(0) = 0$, and $f(1) = 1$. The maxLeashLength(f, g, π, σ) between two curves π and σ with respect to the reparameterisations f and g is defined as follows.

$$\text{maxLeashLength}(f, g, \pi, \sigma) = \max_{t\in[0,1]} \|\pi(f(t)) - \sigma(g(t))\|$$

Imagine the scenario where a person is walking their dog with a leash connecting them: the person needs to stay on π while walking according to f, and the dog needs to stay on σ while walking according to g. The maximum leash length is the width between π and σ with respect to the reparameterisations f and g. The Fréchet distance $d_{\mathcal{F}}(\pi, \sigma)$ is the minimum leash length required over all possible walks (defined by reparameterisations f and g).

$$d_{\mathcal{F}}(\pi, \sigma) = \inf_{f,g\in[0,1]\to[0,1]} \text{maxLeashLength}(f, g, \pi, \sigma)$$

Problems relating to the Fréchet distance are commonly solved in a configuration space called the *freespace diagram*. In our problem, we assume for simplicity that both input curves have exactly n vertices. The *freespace* $\mathcal{F}_\delta(\pi, \sigma)$ is a collection of points in $\mathbb{R}^2$ in the range $[1, n] \times [1, n]$. A point (x, y) from $[1, n] \times [1, n]$ is in the freespace if the Euclidean norm (distance) $\|\pi(x) - \sigma(y)\|$ between $\pi(x)$ and $\sigma(y)$ is at most δ. As opposed to the freespace, we will call $[1, n] \times [1, n] \setminus \mathcal{F}_\delta(\pi, \sigma)$ the *non-free space*.

$$\mathcal{F}_\delta(\pi, \sigma) = \{(x, y) \in [1, n] \times [1, n] \mid \|\pi(x) - \sigma(y)\| \le \delta\}$$

The *freespace diagram* $\mathcal{D} = \mathcal{D}_\delta(\pi, \sigma)$ is the partition of the freespace into $(n - 1) \times (n - 1)$ cells. A cell $C_{i,j}$ in $\mathcal{D}$ contains the freespace in the range $[i, i+1]\times[j, j+1]$. Alt and Godau [7]

made the critical observation that the freespace within a cell $C_{i,j}$ is an intersection between an ellipse $E_{i,j}$ and the square $S_{i,j} = [i, i+1] \times [j, j+1]$; hence it is convex with constant description complexity. By Fact 1, $d_{\mathcal{F}}(\pi, \sigma) \leq \delta$ if and only if there is an xy-monotone path in $\mathcal{D}_\delta(\pi, \sigma)$ from $(1, 1)$ to (n, n) through the freespace.

An intersection between the boundaries $\partial E_{i,j}$ and $\partial S_{i,j} \setminus \{(i, j), (i+1, j), (i, j+1), (i+1, j+1)\}$ is called a *critical point*. A point $(i, j) \in \mathbb{N} \times \mathbb{N}$ is a *corner point*. We say the line $\pi[i] \times [1, n]$ is the *freespace diagram boundary* defined by π_i (and analogous for σ_i). We say the strip $[i, i+1] \times [1, n]$ is the ith *column*, and the strip $[1, n] \times [i, i+1]$ is the ith *row*. We say a critical point p is a row (resp. column) critical point if p lies on a vertical (resp. horizontal) boundary.

2.3 From freespace reachability to graph reachability

In this and the next section, we describe our ideas using an intuitive class of transformations: translations in $\mathbb{R}^2$. Specifically, we determine whether there exists a translation vector $t = \lambda\vec{v}$ such that $d_{\mathcal{F}}(\pi, \sigma + t) \leq \delta$, where $\lambda \in \mathbb{R}_{\geq 0}$ and $\vec{v}$ is a unit vector. For this, we first describe how to transform the freespace reachability problem into a graph reachability problem.

Using the freespace diagram $\mathcal{D} = \mathcal{D}_\delta(\pi, \sigma)$, we construct a *refined freespace diagram* (see Figure 2.1, left). Let $l(\pi_i)$ (resp. $l(\sigma_j)$) be the vertical (resp. horizontal) boundary of $\mathcal{D}$ defined by π_i (resp. σ_j). For every critical point p on the boundaries, we draw a perpendicular *grid line* $l(p)$ through p. Note that, by definition, a critical point does not coincide with a corner point of $\mathcal{D}$. If p lies on the horizontal boundary, $l(p)$ is a vertical line; otherwise, $l(p)$ is a horizontal line. The refined freespace diagram includes all grid lines, the freespace diagram boundaries, and their intersections. For simplicity, we redefine $\mathcal{D}_\delta(\pi, \sigma)$ to denote the refined freespace diagram. Let the intersection between a grid line and a freespace boundary be a *propagated critical point*.

Using the refined freespace diagram $\mathcal{D}_\delta(\pi, \sigma)$, we construct a *refined freespace graph (refined FSG or FSG for short)* $\mathcal{G}^f = \mathcal{G}^f_\delta(\pi, \sigma)$ as follows (see Figure 2.1, right). For every intersection

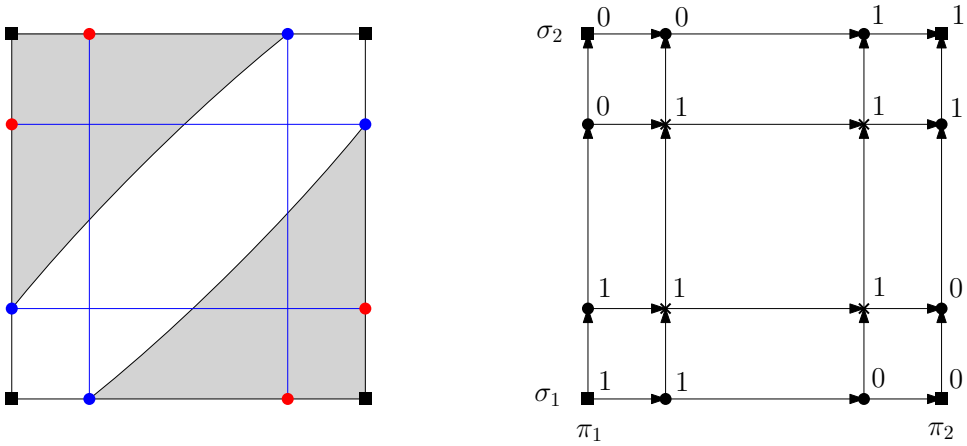

FIGURE 2.1: From the refined freespace diagram to the refined freespace graph. On the left freespace diagram (FSD), we have the corner points, the critical points, and the propagated critical points marked by squares, blue circles, and red circles, respectively. On the right freespace diagram graph (FSG), the corner, boundary, and interior vertices are marked by squares, circles, and crosses, respectively.

between a grid line and a freespace boundary, add a *boundary vertex*. For every intersection between two grid lines, add an *interior vertex*. For every intersection between two freespace boundaries, add a *corner vertex*. Note that each vertex in $\mathcal{G}^f$ is uniquely defined by an ordered pair (p, q), where $l(p)$ is vertical and $l(q)$ is horizontal; p (resp. q) is either a vertex of π (resp. σ) or a critical point in $\mathcal{D}$ — let $\mathbf{v}(p, q)$ denote such vertex in $\mathcal{G}^f$. Each vertex is assigned a weight that is either 1 or 0. For every vertex $u = \mathbf{v}(p, q)$, if $a = l(p) \cap l(q)$ lies on a boundary of the freespace diagram, we set $\mathbf{w}(u) = 1$ if a lies in the freespace or $\mathbf{w}(u) = 0$ otherwise. We set the weights of the rest of the vertices, the interior vertices, to 1. We say a vertex u is *activated* if $\mathbf{w}(u) = 1$ or *deactivated* if $\mathbf{w}(u) = 0$. We say a vertex $\mathbf{v}(\pi_i, \cdot)$ (resp. $\mathbf{v}(\cdot, \sigma_j)$) lies on the freespace graph boundary defined by π_i (resp. σ_j).

To construct edges, we use the geometric positions of the grid lines and the FSG boundaries. For vertices in $\mathcal{G}^f$ defined by every grid line or FSG boundary $l(p)$, add a directed edge $(\mathbf{v}(p, q), \mathbf{v}(p, q'))$ to $\mathcal{G}^f$ if $l(q)$ is immediately below $l(q')$. If $l(q)$ and $l(q')$ overlap, break ties arbitrarily. Analogously, add a directed edge $(\mathbf{v}(p, q), \mathbf{v}(p', q))$ if $l(p)$ is immediately to the left of $l(p')$. A path $P \subseteq \mathcal{G}^f$ is an ordered subset $(a_1, b_1), ..., (a_m, b_m)$ of edges where $b_i = a_{i+1}$ for all $1 \leq i < m$. We say P is a *feasible path* if $a_1 = \mathbf{v}(\pi_1, \sigma_1)$, $b_m = \mathbf{v}(\pi_n, \sigma_n)$, and $\mathbf{w}(a_i) = \mathbf{w}(b_i) = 1$ for all values of i. When a_1 and b_m are specified, P is a feasible path

if P uses exclusively activated vertices. We say an FSG $\mathcal{G}^f$ is *st-reachable* if there exists a feasible path in $\mathcal{G}^f$ from $\mathbf{v}(\pi_1, \sigma_1)$ to $\mathbf{v}(\pi_n, \sigma_n)$. The following lemma is naturally derived from the properties of the freespace diagram.

LEMMA 2. *The FSG $\mathcal{G}^f = \mathcal{G}^f_\delta(\pi, \sigma)$ is st-reachable if and only if $d_\mathcal{F}(\pi, \sigma) \leq \delta$.*

PROOF. By Fact 1, $d_\mathcal{F}(\pi, \sigma) \leq \delta$ if and only if there exists an xy-monotone path through the freespace from $(1, 1)$ to (n, n) in $\mathcal{D}_\delta(\pi, \sigma)$. Such an xy-monotone path can always be transformed into a feasible path P. Every subpath $P_{i,j} = P \cap C_{i,j}$ within the cell $C_{i,j}$ is a line segment ab, where both a and b are either critical points or corner points. We will argue that for every subpath $P_{i,j}$ from the critical point $a = l(p) \cap l(q)$ to $b = l(p') \cap l(q')$, there exists a path $P^f_{i,j} \subseteq \mathcal{G}^f$ from $\mathbf{v}(p, q)$ to $\mathbf{v}(p', q')$ using only vertices of weight 1. By construction, $\mathbf{w}(\mathbf{v}(p, q)) = \mathbf{w}(\mathbf{v}(p', q')) = 1$, and all interior vertices have weight 1. Therefore, a path starting at $\mathbf{v}(p, q)$, turning at $\mathbf{v}(p', q)$, and finally arriving at $\mathbf{v}(p', q')$ is a valid candidate for $P^f_{i,j}$, as required.

Analogously, if a subpath $P^f_{i,j}$ from $\mathbf{v}(p, q)$ to $\mathbf{v}(p', q')$ exists, then we use the segment ab as $P_{i,j}$. The segment ab must lie in the freespace, because the freespace is convex within every cell $C_{i,j}$, as shown by Alt and Godau [7]. □

Now consider translating σ along $\vec{v}$. As σ translates continuously, the FSG also changes, and we would like to know how these changes affect the st-reachability of $\mathcal{G}^f$. To do this, we track the following freespace events. See Figure 2.2 for visualisations of these freespace events.

DEFINITION 3. *We define the following* row freespace events. *Column freespace events are defined analogously.*

(1) *A* Vertex-edge (VE) event *is defined by a pair $(p, \overline{\sigma_w})$, where p is either a vertex of π or a row critical point in row w.*
   (a) *Entering/leaving event: the corner point p enters or leaves the freespace.*
   (b) *Appearing/disappearing event: the grid line $l(p)$ appears or disappears as the critical point p appears or disappears.*

(2) *A* Vertex-vertex-edge (VVE) event *is defined by a triplet* $(p, q, \overline{\sigma_w})$*, where* p *and* q *are row critical points in row* w*.*

(a) *Overlapping event: two grid lines* $l(p)$ *and* $l(q)$ *overlap.*

(b) *Separating event: two overlapping grid lines* $l(p)$ *and* $l(q)$ *no longer overlap.*

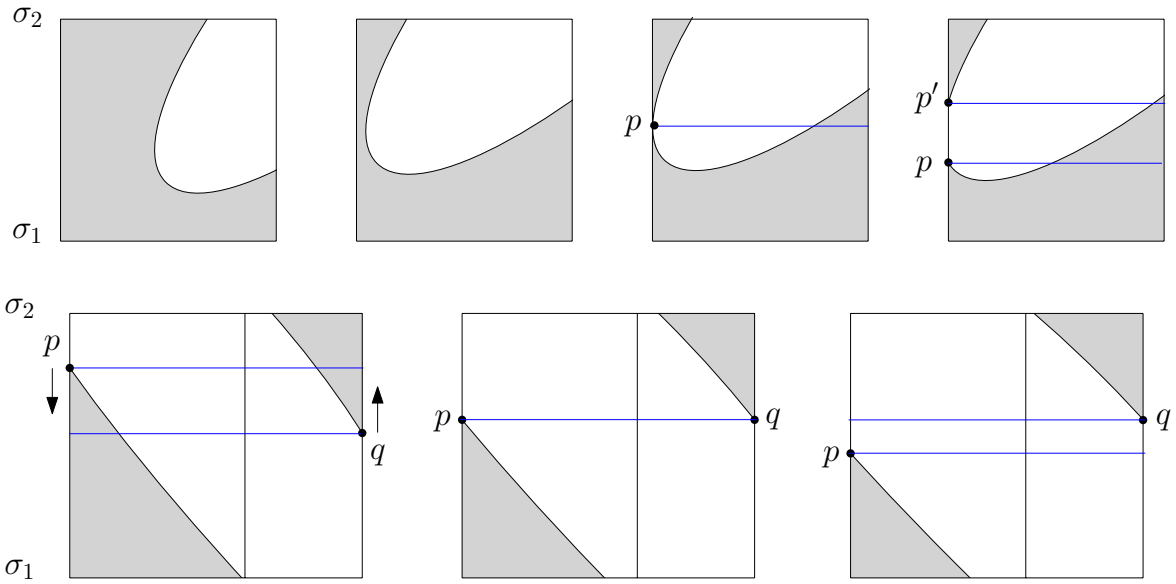

FIGURE 2.2: In the top row of figures, as the segment $\sigma_1\sigma_2$ translates, the critical points p and q move down and up, respectively. As $l(p)$ and $l(q)$ (coloured in blue) move, they overlap and then separate. In the bottom row, as $\sigma_1\sigma_2$ translates, a new critical point p appears, and then a new critical point p' appears.

For now, we assume that we are given an ordered set $T = \{t_1, t_2, ..., t_m\}$ of translations, where $t_i = \lambda_i \vec{v}$, $\lambda_i \in \mathbb{R}_{\geq 0}$, and $\lambda_i \leq \lambda_{i+1}$. We further assume that T is *complete*, which is defined as follows.

DEFINITION 4. *We say the ordered set* $T = \{t_1, ..., t_m\}$ *of translations is* complete *if* $\mathcal{G}_\delta^f(\pi, \sigma + t_i)$ *and* $\mathcal{G}_\delta^f(\pi, \sigma + t_{i+1})$ *differ by exactly one freespace event, for all* $1 \leq i < m$*.*

Here, $\sigma + t$ denotes the curve obtained by adding the translation vector t to σ. To update the freespace graph to reflect the state of the freespace diagram, we define the following freespace graph operations. Let $R(p)$ be the set of vertices defined by $l(p)$ (a row of vertices) , and let $R(p)[i]$ denote the ith vertex, where $i \geq 1$.

DEFINITION 5. *We define the following* freespace graph operations.

- *Vertex operation: either activate or deactivate a vertex of $\mathcal{G}^f$.*
- *Row operation: either insert or delete a row of $\mathcal{G}^f$. Column operations are defined analogously.*
  - *To insert a row $R(p)$ of vertices between adjacent rows $R(p_a)$ and $R(p_b)$, add a vertex $\mathbf{v}(q, p)$ for every q, where q is either a critical point on a horizontal FSG boundary or a vertex of π. For every $1 \leq i < |R(p_a)|$,*
    (1) *remove the edge $(R(p_b)[i], R(p_a)[i])$,*
    (2) *add the horizontal edge $(R(p)[i], R(p)[i+1])$, and*
    (3) *add the vertical edges $(R(p_b)[i], R(p)[i])$ and $(R(p)[i], R(p_a)[i])$.*
  - *To remove the row $R(p)$ of vertices, remove $R(p)$ and its adjacent edges. For every $1 \leq i \leq |R(p_a)|$, add the edge $(R(p_b)[i], R(p_a)[i])$.*

We show that we can update the FSG $\mathcal{G}^f_\delta(\pi, \sigma + t_{i-1})$ to $\mathcal{G}^f_\delta(\pi, \sigma + t_i)$ using a constant number of freespace graph operations, plus processing time. For these updates, we distinguish between a corner vertex operation and a boundary vertex operation. We use corner vertex operations to handle VE event updates, and boundary vertex operations to handle VVE event updates.

LEMMA 6. *Let $\mathcal{G}^f_i = \mathcal{G}^f_\delta(\pi, \sigma + t_i)$. Given $\mathcal{G}^f_{i-1}$, to compute $\mathcal{G}^f_i$, it takes*

- *$\mathcal{O}(1)$ time and $\mathcal{O}(1)$ corner vertex operations if t_i is an entering/leaving event,*
- *$\mathcal{O}(1)$ time and $\mathcal{O}(1)$ boundary vertex operations if t_i is an overlapping or separating event,*
- *$\mathcal{O}(n)$ time plus $\mathcal{O}(1)$ row operations if t_i is an appearing/disappearing event.*

PROOF. If t_i is an entering/leaving event, we simply set the weight of the respective corner vertex to either 1 or 0. If t_i is a VVE event defined by $(p, q, \overline{\sigma_w})$, let p and q lie on the ith and jth freespace boundaries, respectively. Observe that when a VVE event occurs, among all vertices partly defined by p or q, only the weights of vertices $\mathbf{v}(\pi_i, p)$, $\mathbf{v}(\pi_j, p)$,

$\mathbf{v}(\pi_i, q)$, and $\mathbf{v}(\pi_j, q)$ change. It is sufficient to spend $\mathcal{O}(1)$ time to determine and update their weights by computing cells $C_{i,w}$ and $C_{j,w}$ from scratch.

If t_i is an appearing/disappearing event in the jth row, it takes linear time to recompute the row critical points in this row. Then, it takes linear time to compute the grid lines $l(p_a)$ and $l(p_b)$ that $l(p)$ lies between by recomputing the cells in the jth row. Once $l(p_a)$ and $l(p_b)$ are identified, it takes exactly one row insertion or deletion to update $\mathcal{G}^f_{i-1}$ to $\mathcal{G}^f_i$. Therefore, it takes $\mathcal{O}(n)$ time plus a constant number of row operations. □

Using a naive implementation, in the worst case, a row operation can take $\Omega(n^2)$ time since there are $\Omega(n^2)$ vertical grid lines. Determining st-reachability takes $\Omega(n^4)$ time, since there are $\Omega(n^4)$ vertices in the FSG. Hence, using the FSG directly would not result in a faster algorithm. In the next section, we fix these issues by defining an "equivalent" grid graph in which updates and queries can be done more efficiently than the naive implementation.

2.4 From freespace graph reachability to grid graph reachability

In this section, using the refined FSG $\mathcal{G}^f = \mathcal{G}^f_\delta(\pi, \sigma)$, we define a *grid graph* $\mathcal{G}^g = \mathcal{G}^g_\delta(\pi, \sigma)$. We then show that $\mathcal{G}^f$ is st-reachable if and only if $\mathcal{G}^g$ is st-reachable. An advantage of the newly defined grid graph is that its number of vertices does not change under updates. We show that the structural changes implied by the FSG operations can be simulated by simply modifying the weights in the grid graph, without needing to add or remove vertices. These features of the grid graph allow us to use the offline dynamic grid reachability result in [26] to obtain a faster update and query time.

The grid graph $\mathcal{G}^g$ contains all vertices of the freespace graph $\mathcal{G}^f$. In addition to the freespace graph $\mathcal{G}^f$, we add a set of placeholder vertices to the grid graph $\mathcal{G}^g$ so as to maintain the same number of vertices in the grid graph $\mathcal{G}^g$ under updates. Refer to Figure 2.3 for an illustration. We define the placeholder vertices as follows. Let m^r_i (resp. m^c_i) be the number of row critical points in the ith row (resp. column) of $\mathcal{D} = \mathcal{D}_\delta(\pi, \sigma)$. We define a family of ordered sets

$H_1^r, ..., H_{n-1}^r$ of row placeholder points. Each set H_i^r contains exactly $2n - m_i^r + 2$ points, and let $H_i^r[j]$ denote the jth point for $1 \leq j \leq 2n - m_i^r + 2$. The family of ordered sets of column placeholder points $H_1^c, ..., H_{n-1}^c$ is analogously defined. For a row (resp. column) placeholder point h, the *placeholder line* $l(h)$ is horizontal (resp. vertical). Note that the placeholder points and lines are abstractly defined, as they do not exist in $\mathcal{D}$.

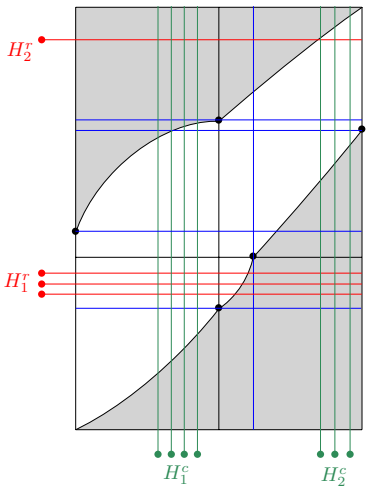

FIGURE 2.3: A visual illustration of the placeholder lines and their relative positions among the grid lines and the freespace diagram boundaries is shown. The row placeholder points and lines (resp. column placeholder points and lines) are coloured in red (resp. green). The grid lines are coloured in blue.

Finally, define $\mathbf{v}(p, q)$ to be a *placeholder vertex* if p or q is a placeholder point. The weight of a placeholder vertex on a boundary matches the weight of the adjacent corner vertex either directly above or directly to its right. Specifically, for every h in every H_i^r and every point p in the union of the vertices of π and the row critical points, we set $\mathbf{w}(\mathbf{v}(p, h)) = \mathbf{w}(\mathbf{v}(\pi_i, \sigma_{j+1}))$ if $p = \pi_i$. Analogously, for every $h \in H_i^c$, we set $\mathbf{w}(\mathbf{v}(h, q)) = \mathbf{w}(\mathbf{v}(\pi_{i+1}, \sigma_j))$ if $q = \sigma_j$. For all remaining placeholder vertices, we set their weights to 1. This completes the definition of placeholder vertices.

To define the edges in $\mathcal{G}^g$, we define the ordering of the placeholder lines among the grid lines and the freespace boundaries. The lowest placeholder line $l(H_i^r[1])$ lies above all grid lines $l(p)$, where p is a row critical point in the ith row. The placeholder line $l(H_i^r[j+1])$ is above $l(H_i^r[j])$. The freespace boundary $l(\sigma_{i+1})$ lies above the highest placeholder line $l(H_i^r[2n - m_i^r + 2])$. The ordering involving vertical placeholder lines is defined analogously.

The set of placeholder lines defined by H_i^c lies between $l(\pi_{i+1})$ and the rightmost grid line in the ith column.

For vertices in $\mathcal{G}^g$ defined by $l(p)$, add a directed edge $(\mathbf{v}(p,q), \mathbf{v}(p,q'))$ to $\mathcal{G}^g$ if $l(q)$ is immediately below $l(q')$, and add a directed edge $(\mathbf{v}(q,p), \mathbf{v}(q',p))$ to $\mathcal{G}^g$ if $l(q)$ is immediately to the left of $l(q')$. If $l(q)$ and $l(q')$ overlap, then break ties using the same ordering as in $\mathcal{G}^f$. Add an additional diagonal directed edge $(\mathbf{v}(p,q), \mathbf{v}(p',q'))$ if p is immediately to the left of p' and q is immediately below q'. Note that a vertex defined by two placeholder points is simultaneously a placeholder vertex as well as an interior vertex.

Next, we check that our construction of the freespace grid graph fits the grid graph definition of [26]. An $N \times N$ grid graph consists of vertices numbered from $(1,1)$ to (N,N) and edges from vertex (i,j) to each of vertices $(i+1,j)$, $(i,j+1)$, and $(i+1,j+1)$, where the weight of a vertex is either 1 or 0. By construction, each of the $n-1$ rows in $\mathcal{D}_\delta(\pi,\sigma)$ contains exactly $2n$ critical points and placeholder points combined. Together with n horizontal boundaries, there are $N = 2n \cdot (n-1) + n = 2n^2 - n$ horizontal lines. For the same reason, there are N vertical lines. Clearly, $\mathcal{G}_\delta^g(\pi,\sigma)$ is an $N \times N$ grid graph.

Before proving that $\mathcal{G}_\delta^f(\pi,\sigma)$ and $\mathcal{G}_\delta^g(\pi,\sigma)$ are equivalent in terms of st-reachability, we first show that if there exists a feasible path $P \subseteq \mathcal{G}^g$, then there exists a feasible path P' using only rectilinear edges.

OBSERVATION 7. *Let P be a feasible path in $\mathcal{G}^g$. If P contains a corner vertex $\mathbf{v}(\pi_i, \sigma_j)$, then let $l(p_l)$ and $l(p_r)$ be the first grid lines that are not placeholder lines to the left and right of $l(\pi_i)$, respectively. Similarly, let $l(p_a)$ and $l(p_b)$ be the first grid lines above and below $l(\sigma_j)$, respectively. Then, we have either $\mathbf{w}(\mathbf{v}(p_l, \sigma_j)) = 1$ or $\mathbf{w}(\mathbf{v}(\pi_i, p_b)) = 1$. Analogously, we have either $\mathbf{w}(\mathbf{v}(\pi_i, p_a)) = 1$ or $\mathbf{w}(\mathbf{v}(p_r, \sigma_j)) = 1$.*

PROOF. See Figure 2.4 for an illustration. For the sake of contradiction, assume that the weights of both $\mathbf{v}(p_l, \sigma_j)$ and $\mathbf{v}(\pi_i, p_b)$ are 0. Since the freespace is convex within a cell, $l(\pi_i) \cap l(\sigma_j)$ is the only point in $C_{i-1,j-1}$ that lies in the freespace. By construction, this implies that for all grid lines $l(p)$ lying between $l(\pi_{i-1})$ and $l(\pi_i)$, we have $\mathbf{w}(\mathbf{v}(p, \sigma_{j-1})) = 0$. For all $l(q)$ lying between $l(\sigma_{j-1})$ and $l(\sigma_j)$, we have $\mathbf{w}(\mathbf{v}(\pi_{i-1}, q)) = 0$. Therefore, there

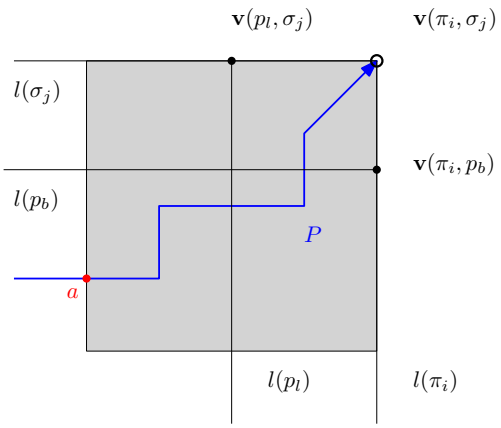

FIGURE 2.4: If vertices $\mathbf{v}(\pi_i, p_b)$ and $\mathbf{v}(p_l, \sigma_j)$ are deactivated, point (i, j) is the only point that lies in the freespace in $C_{i-1,j-1}$. Due to monotonicity, the path P must use at least one deactivated vertex (say, a), which contradicts the fact that P is feasible.

exists at least one vertex of weight 0 in P, contradicting the assumption that P uses exclusively vertices of weight 1. An analogous argument holds for the case where the weights of both $\mathbf{v}(p_r, \sigma_j)$ and $\mathbf{v}(\pi_i, p_a)$ are 0. □

Due to the properties of $\mathcal{G}^g$, a diagonal edge in a path P can be replaced by the rectilinear edges in the same "cube". Furthermore, if the final vertex b of a subpath $P_j \subseteq P$ is a placeholder vertex, we can transform P_j so that it ends at a non-placeholder vertex. We have the following lemma.

LEMMA 8. *If there is a feasible path P in a grid graph $\mathcal{G}^g$, then there is a feasible path P' in $\mathcal{G}^g$ with the following properties.*

(1) *P' does not contain diagonal edges.*
(2) *For every $1 \leq j \leq n$, the first vertex of P' partly defined by σ_j is not a placeholder vertex.*

PROOF. We first replace the diagonal edges in P. Consider a diagonal edge $(a, b) \in P$ in the "cube", where (a, c_t) and (c_b, b) are the vertical edges, and (a, c_b) and (c_t, b) are the horizontal edges. Observe that, by construction, either both a and b lie on the boundaries, or at least one of them is an interior vertex. If both a and b lie on the boundaries, then either

c_b or c_t must be activated, since the opposite would contradict either Observation 7 or the convexity of the freespace in a cell.

If a is an interior vertex, we consider the following cases. If b is also an interior vertex, then so are c_t and c_b with weight 1, and we replace (a, b) with (a, c_t) and (c_t, b). If b is a boundary vertex, then either c_b or c_t is an interior vertex, and we replace (a, b) with either (a, c_b) and (c_b, b) or (a, c_t) and (c_t, b). The more interesting case is when $b = \mathbf{v}(\pi_{i+1}, \sigma_{j+1})$ is a corner vertex. In this case, $c_b = \mathbf{v}(\pi_{i+1}, h \in H_i^r)$, and by construction, $\mathbf{w}(c_b) = \mathbf{w}(b) = 1$. We replace (a, b) with (a, c_b) and (c_b, b).

If b is an interior vertex, we use similar case distinctions. If a is an interior vertex, then c_b must also be an interior vertex. If a is a boundary vertex lying on the left (resp. bottom) boundary, then c_b (resp. c_t) is an interior vertex. The more interesting case is where $a = \mathbf{v}(\pi_i, \sigma_j)$ is a corner vertex. In this case, both c_b and c_t are boundary vertices. By Observation 7, at least one of them (say, c_b) has weight 1, and we replace (a, b) with (a, c_b) and (c_b, b).

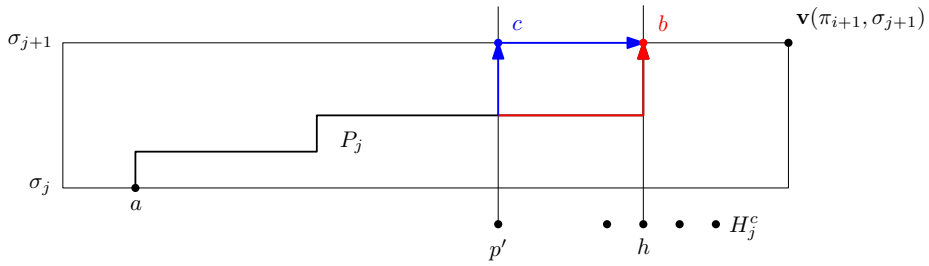

FIGURE 2.5: A path P_{j-1} ending at a placeholder vertex b can be transformed to end at a non-placeholder vertex c.

With P containing exclusively rectilinear edges, we partition P into subpaths lying on different rows (see Figure 2.5). Specifically, let $P_j \subseteq P$ be the subpath containing the first vertex a defined by σ_j and the first vertex b defined by σ_{j+1}. We first transform P_j to guarantee that b is not defined by a placeholder point. Let $b = \mathbf{v}(h \in H_i^c, \sigma_{j+1})$, and note that $\mathbf{w}(b) = \mathbf{w}(\mathbf{v}(\pi_{i+1}, \sigma_{j+1})) = 1$. Let $l(p')$ be immediately to the left of $l(H_i^c[1])$. By Observation 7, $\mathbf{w}(c = \mathbf{v}(p', \sigma_{j+1})) = 1$. Combining this argument with the fact that all interior vertices have weight 1 and the rectilinearity of P, we can transform P_j to end at c, and P_{j+1} to start at c. Since the first vertex $\mathbf{v}(\pi_1, \sigma_1)$ and the final vertex $\mathbf{v}(\pi_n, \sigma_n)$ of P are

not defined by placeholder vertices, once we apply the transformation above, the first vertex of every subpath P_j is not defined by a placeholder vertex. The proof is complete. □

We next observe that if there is a path in $\mathcal{G}^g$ that does not use a placeholder vertex, the path also exists in $\mathcal{G}^f$. Indeed, excluding the placeholder lines, $\mathcal{G}^g$ and $\mathcal{G}^f$ use the same set of grid lines and FSG boundaries, and the same ordering.

OBSERVATION 9. *If there is a path P in $\mathcal{G}^g = \mathcal{G}^g_\delta(\pi, \sigma)$ such that P does not use any placeholder vertices or diagonal edges, then P also exists in $\mathcal{G}^f = \mathcal{G}^f_\delta(\pi, \sigma)$.*

To demonstrate that $\mathcal{G}^f$ and $\mathcal{G}^g$ are equivalent with respect to st-reachability, we first note that any feasible path in $\mathcal{G}^g$ corresponds to a feasible path in $\mathcal{G}^f$. Specifically, given a subpath $P \subseteq \mathcal{G}^g$ that traverses the "strip" defined by a single set of placeholder points, we can always construct a corresponding subpath $Q \subseteq \mathcal{G}^f$ such that Q starts and ends at the same vertices as P. We have Lemma 10 and Lemma 11.

LEMMA 10. *Let P be a path in $\mathcal{G}^g = \mathcal{G}^g_\delta(\pi, \sigma)$. Let P start at a non-placeholder vertex $\mathbf{v}(p, \sigma_j)$. Let $(\mathbf{v}(p', q'), \mathbf{v}(p', H^r_j[1]))$ be the last edge of P, where $q' \neq H^r_j[1]$. Then there exists a path Q from $\mathbf{v}(p, \sigma_j)$ to $\mathbf{v}(p', q')$ in $\mathcal{G}^f = \mathcal{G}^f_\delta(\pi, \sigma)$.*

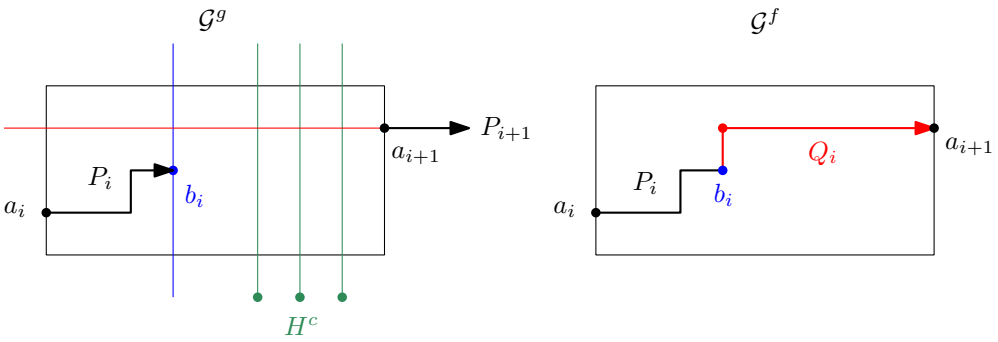

FIGURE 2.6: A path Q_i in $\mathcal{G}^f$ can be constructed by connecting the vertex b_i where P_i ends and the vertex a_{i+1} where P_{i+1} starts.

PROOF. Let $P_1, ..., P_u$ be the subpaths of P obtained by removing all placeholder vertices from P (see Figure 2.6). For $1 \leq i \leq u$, let P_i be the path starting at a_i and ending at b_i. By Observation 9, the path P_i also exists in $\mathcal{G}^f$. Since the placeholder vertices are removed, each

a_i and b_i is either a boundary vertex or an interior vertex. Because every interior vertex has weight 1, a path Q_i from b_i to a_{i+1} exists in $\mathcal{G}^f$. We set

$$Q = \Big(\bigcup_{1 \leq i \leq u-1} (P_i \cup Q_i) \Big) \cup P_u$$

to complete the proof. □

LEMMA 11. *Let P be a path in $\mathcal{G}^g = \mathcal{G}^g_\delta(\pi, \sigma)$. Let $(\mathbf{v}(p,q), \mathbf{v}(p, H^r_j[1]))$ be the first edge of P, and assume that P ends at the non-placeholder vertex $\mathbf{v}(p', \sigma_{j+1})$. Then there exists a path Q from $\mathbf{v}(p,q)$ to $\mathbf{v}(p', \sigma_{j+1})$ in $\mathcal{G}^f = \mathcal{G}^f_\delta(\pi, \sigma)$.*

PROOF. If P does not contain a vertex defined by any vertical boundary, then the lemma holds trivially, as all interior vertices have weight 1. Otherwise, P contains a set of vertices defined by $\pi_u, ..., \pi_w$ in increasing order of indices. By Observation 7 and by the convexity of the freespace within a cell, there exists a path P_1 in $\mathcal{G}^f$ from $\mathbf{v}(p,q)$ to $\mathbf{v}(\pi_u, \sigma_{j+1})$, and a path P_2 from $\mathbf{v}(\pi_w, \sigma_{j+1})$ to $\mathbf{v}(p', \sigma_{j+1})$.

Since P uses only activated vertices, for every $u \leq i \leq w$, we have $\mathbf{w}(\mathbf{v}(\pi_i, h)) = 1$ for some $h \in H^r_j$. By the construction of $\mathcal{G}^g$, the vertex $\mathbf{v}(\pi_i, h)$ is activated if and only if $\mathbf{v}(\pi_i, \sigma_{j+1})$ is activated, and therefore the latter must also have weight 1. By the convexity of the freespace within a cell, since both $\mathbf{v}(\pi_i, \sigma_{j+1})$ and $\mathbf{v}(\pi_{i+1}, \sigma_{j+1})$ are activated, every vertex $\mathbf{v}(p, \sigma_{j+1})$ with $l(p)$ lying between $l(\pi_i)$ and $l(\pi_{i+1})$ is also activated. Hence, for all $u \leq i \leq w$, there exists a path $P_i \subseteq \mathcal{G}^f$ from $\mathbf{v}(\pi_i, \sigma_{j+1})$ to $\mathbf{v}(\pi_{i+1}, \sigma_{j+1})$ using only activated vertices. We set

$$Q = P_1 \cup \Big(\bigcup_{u \leq i \leq w} P_i \Big) \cup P_2$$

to complete the proof. □

We are finally ready to show that $\mathcal{G}^f$ and $\mathcal{G}^g$ are equivalent in terms of st-reachability.

LEMMA 12. *There exists a feasible path P^f in the freespace graph $\mathcal{G}^f_\delta(\pi, \sigma)$ if and only if there exists a feasible path P^g in the grid graph $\mathcal{G}^g_\delta(\pi, \sigma)$.*

PROOF. First, we observe that if there is a feasible path $P^f \subseteq \mathcal{G}^f$, then there is a feasible path $P^g \subseteq \mathcal{G}^g$. Consider a partition of P^f into subpaths lying in different cells of the freespace diagram. Let the subpath $P^f_{i,j} \subseteq P^f$ start at the vertex a and end at the vertex b, where a lies on the bottom or left boundary of $C_{i,j}$ and b lies on the top or right boundary of $C_{i,j}$. Since all interior vertices have weight 1, and a diagonal edge can be used when b is a corner vertex, there exists a path $P^g_{i,j} \subseteq \mathcal{G}^g$ from a to b using activated vertices.

Second, we show that if there is a feasible path $P^g \subseteq \mathcal{G}^g$, then there is a feasible path $P^f \subseteq \mathcal{G}^f$. We use an analogous argument, constructing a feasible path P^f from the subpaths of P^g. By Lemma 8, P^g uses exclusively rectilinear edges. Furthermore, by the same lemma, P^g can be partitioned into subpaths $P^g_1, \ldots, P^g_{n-1}$ such that for every $1 \leq j \leq n-1$, the subpath P^g_j starts at a non-placeholder vertex $a = \mathbf{v}(p, \sigma_j)$ and ends at another non-placeholder vertex $b = \mathbf{v}(p', \sigma_{j+1})$.

We further partition P^g_j using its first edge (a', b'), where the vertex a' is partly defined by a placeholder point $H^r_j[1]$. By Lemma 10, there exists a path P^{fa}_j in $\mathcal{G}^f$ from a to a'. By Lemma 11, there exists a path P^{fb}_j in $\mathcal{G}^f$ from a' to b. We set

$$P^f_j = P^{fa}_j \cup P^{fb}_j,$$

$$P^f = \bigcup_j P^f_j,$$

which completes the proof. □

Now, we show that freespace graph operations can be implemented efficiently in the grid graph. Recall from Definition 5 and Lemma 6 that corner vertex operations activate or deactivate a vertex of $\mathcal{G}^f$ given an entering or leaving event, boundary vertex operations activate or deactivate a vertex of $\mathcal{G}^f$ given an overlapping or separating event, and row operations insert or delete a row of $\mathcal{G}^f$ given an appearing or disappearing event.

LEMMA 13. *Let $\mathcal{G}^g_i$ be the grid graph associated with the freespace graph $\mathcal{G}^f_i$. Let u be a freespace graph operation that updates $\mathcal{G}^f_1$ to $\mathcal{G}^f_2$. To update $\mathcal{G}^g_1$ to $\mathcal{G}^g_2$, it is sufficient to update the weights of at most*

- $\mathcal{O}(n)$ *vertices if* u *is a corner vertex operation,*
- $\mathcal{O}(1)$ *vertices if* u *is a boundary vertex operation, or*
- $\mathcal{O}(n)$ *vertices if* u *is a row operation.*

PROOF. If u is a corner vertex operation activating a corner vertex $\mathbf{v}(\pi_i, \sigma_j)$, we activate $\mathbf{v}(\pi_i, \sigma_j)$ in $\mathcal{G}^g$. We then activate $\mathbf{v}(\pi_i, h_1)$ for every $h_1 \in H^r_{j-1}$, and activate $\mathbf{v}(h_2, \sigma_j)$ for every $h_2 \in H^c_{i-1}$. Since there are at most a linear number of placeholder points defined per row and per column, this operation requires changing the weights of $\mathcal{O}(n)$ vertices in $\mathcal{G}^g$. If u activates a boundary vertex a, we simply activate a in $\mathcal{G}^g$. If u deactivates a vertex, we use an analogous procedure.

If u is a boundary vertex operation, then to insert a row of vertices while maintaining the properties of the grid graph, we take advantage of the fact that the intersection between the freespace and a cell boundary is a single interval. Specifically, to insert $R(p)$ of vertices below $R(p_a)$ and above $R(p_b)$ in row j, let the critical point p lie on the i'th vertical boundary. For $1 \leq i \leq n$, if $i = i'$, then set $\mathbf{w}(\mathbf{v}(\pi_{i'}, H^r_j[1])) = 1$. For every other value of i, set $\mathbf{w}(\mathbf{v}(\pi_i, H^r_j[1])) = 1$ if both $\mathbf{v}(\pi_i, p_a)$ and $\mathbf{v}(\pi_i, p_b)$ have weight 1. Otherwise, set $\mathbf{w}(\mathbf{v}(\pi_i, H^r_j[1])) = 0$. Finally, reduce the size of H^r_j by one by removing $H^r_j[1]$.

We prove the correctness of the boundary vertex operation by showing that this insertion process maintains the properties of the grid graph. First, the total number of row critical points plus the placeholder points remains the same. Second, it is sufficient to determine the weight of $\mathbf{v}(\pi_i, p)$ by checking the weights of $\mathbf{v}(\pi_i, p_a)$ and $\mathbf{v}(\pi_i, p_b)$. Both intersections $l(\pi_i) \cap l(p_a)$ and $l(\pi_i) \cap l(p_b)$ must lie in the freespace for $l(\pi_i) \cap l(p)$ to lie in the freespace, since the opposite would imply the existence of a grid line between $l(p_a)$ and $l(p_b)$, contradicting the assumption that $l(p_a)$ and $l(p_b)$ are adjacent.

If u is a row operation, to delete $R(p)$ lying in the jth row, let $l(q)$ be the first grid line below $l(H^r_j[1])$. For all $1 \leq i \leq n$, set $\mathbf{w}(\mathbf{v}(\pi_i, q)) = \mathbf{w}(\mathbf{v}(\pi_i, \sigma_j))$. Insert a new placeholder point at the beginning of H^r_j by setting $H^r_j[1] = q$. This process also maintains the properties of the grid graph. Column operations use analogous arguments. □

Given $\mathcal{G}^g_{i-1} = \mathcal{G}^g_\delta(\pi, \sigma+t_{i-1})$ and the event t_i, we can now transform $\mathcal{G}^g_{i-1}$ into $\mathcal{G}^g_i$. Specifically, in Lemma 6, we have shown that if t_i is a VE event, the update requires $\mathcal{O}(n)$ time plus $\mathcal{O}(1)$ corner vertex operations or row operations. If t_i is a VVE event, the update requires $\mathcal{O}(1)$ time plus $\mathcal{O}(1)$ boundary vertex operations. By combining Lemma 6 with Lemma 13, we can bound the number of vertex-weight changes for each type of freespace event.

LEMMA 14. *Given $\mathcal{G}^g_\delta(\pi, \sigma + t_{i-1})$ and the next event t_i, to compute $\mathcal{G}^g_\delta(\pi, \sigma + t_i)$, it takes*

- *$\mathcal{O}(n)$ vertex-weight changes if t_i is a VE event, and*
- *$\mathcal{O}(1)$ vertex-weight changes if t_i is a VVE event.*

We can now summarise Sections 2.3 and 2.4 and state the main lemma of this section. In Lemma 2, we showed that the Fréchet distance $d_\mathcal{F}(\pi, \sigma)$ is at most δ if and only if the refined freespace graph $\mathcal{G}^f = \mathcal{G}^f_\delta(\pi, \sigma)$ is st-reachable. In Section 2.4, for each $\mathcal{G}^f$, we defined an associated grid graph $\mathcal{G}^g = \mathcal{G}^g_\delta(\pi, \sigma)$. In Lemma 12, we showed that $\mathcal{G}^g$ is st-reachable if and only if $\mathcal{G}^f$ is st-reachable. Combining the above with Lemma 14, we obtain the following.

LEMMA 15. *Let $T = \{t_1, \ldots, t_m\}$ be a complete set of freespace events containing exactly m_{vve} VVE events and m_{ve} VE events. Let $N = 2n^2 - n$, and let $T_u(N)$ (resp. $T_q(N)$) be the time complexity to update (resp. query st-reachability in) an $N \times N$ grid graph. It takes*

$$\mathcal{O}(N^2 + (m_{ve} \cdot n + m_{vve}) \cdot T_u(N) + m \cdot T_q(N))$$

time to determine whether there exists $t_i \in T$ such that $d_\mathcal{F}(\pi, \sigma + t_i) \leq \delta$.

Using the results of Alt, Knauer, and Wenk [9], we can build an arrangement in the translation space. From this arrangement, we can compute a complete set of events (translations) T containing $\mathcal{O}(n^6)$ VVE events and $\mathcal{O}(n^5)$ VE events. They showed that it is sufficient to consider only translations in T to determine whether there exists a translation t such that $d_\mathcal{F}(\pi, \sigma + t) \leq \delta$.

To obtain fast updates and queries in the grid graph, we use the offline dynamic grid-reachability result of Bringmann, Künnemann, and Nusser [26]. The problem is defined as

follows. We start with a directed $N \times N$ grid graph, and we are given a set $u_1, \ldots, u_U$ of updates such that each update u_i activates or deactivates a vertex. For $1 \leq i \leq U$, the goal is to decide, after each update u_i, whether there is a feasible path from vertex $\mathbf{v}(1,1)$ to vertex $\mathbf{v}(N,N)$. Their result is as follows.

FACT 16 ([26, Theorem 3.4]). *Offline dynamic grid reachability can be solved in time* $\mathcal{O}(N^2 + UN^{2/3}\log^2 N)$.

We analyse the running time. In total, we require $\mathcal{O}(n^6)$ vertex updates for $\mathcal{G}^g$, and we can update a vertex and perform st-reachability queries in amortised $\mathcal{O}(N^{2/3} \cdot \log^2 N) = \mathcal{O}(n^{4/3} \cdot \log^2 n)$ time. Plugging these running times into Lemma 15 we obtain the following theorem.

THEOREM 1. *The Fréchet distance under translation in* $\mathbb{R}^2$ *can be decided in* $\mathcal{O}(n^{7+\frac{1}{3}}\log^2 n)$ *time.*

2.5 Fréchet distance under transformation

In this section, we consider a class $\mathcal{T}$ of transformations that is *rationally parameterised* or *rationally represented with k degrees of freedom*, as defined by Wenk [141]. The set of rationally parameterised transformations is a subset of affine transformations, and an affine transformation is composed of a linear transformation and a translation. In the definition below, the pair (A,t) represents the affine transformation obtained by composing a linear transformation A, given as a $d \times d$ matrix, and a translation t, given as a d-dimensional vector.

DEFINITION 17 ([141, Definition 25]). *Let* $1 \leq k \leq d^2 + d$, *and let* $p_1, \ldots, p_{d^2+d}, q_1, \ldots, q_{d^2+d} \in \mathbb{R}[X_1, \ldots, X_k]$ *be* $2(d^2+d)$ *polynomials of constant degree in* k *variables, such that* $q_i(x) \neq 0$ *for all* $1 \leq i \leq d^2+d$ *and for all* $x \in \mathbb{R}^k$. *Let* $r_i := p_i/q_i$ *for*

all $1 \leq i \leq d^2 + d$*, such that* $r_i(x) := p_i(x)/q_i(x)$ *for all* $x \in \mathbb{R}^k$*. If*

$$\mathcal{T} = \left\{ \begin{pmatrix} \begin{pmatrix} r_1(x) & \dots & r_d(x) \\ r_{d+1}(x) & \dots & r_{2d}(x) \\ \vdots & \ddots & \vdots \\ r_{d^2-d+1}(x) & \dots & r_{d^2}(x) \end{pmatrix}, \begin{pmatrix} r_{d^2+1}(x) \\ r_{d^2+2}(x) \\ \vdots \\ r_{d^2+d}(x) \end{pmatrix} \end{pmatrix} \middle| \; x \in \mathbb{R}^k \right\},$$

then we call $\mathcal{T}$ rationally parameterised *or* rationally represented with k degrees of freedom *(dof).* $\mathbb{R}^k$ *is called the parameter space of* $\mathcal{T}$*.*

Let $\mathbb{R}^k$ be the parameter space of $\mathcal{T}$. For $x \in \mathbb{R}^k$, let τ_x denote the transformation defined by the k-tuple x of parameters. Let π and σ be two d-dimensional polygonal curves. Let $\tau_x(\sigma)$ be the curve obtained by applying the transformation τ_x to σ.

In $\mathbb{R}^k$, a *vertex-vertex-edge (VVE) critical transformation* $\mathcal{T}_\delta^{vve}(\pi_i, \pi_j, \overline{\sigma_w})$ is the set of all points $x \in \mathbb{R}^k$ such that the segment $\tau_x(\overline{\sigma_w})$ lies on the intersection of the boundaries of the two d-spheres centred at π_i and π_j, respectively. It is formally defined as follows.

$$\mathcal{T}_\delta^{vve}(\pi_i, \pi_j, \overline{\sigma_w}) = \{x \in \mathbb{R}^k \mid \exists z \in \overline{\sigma_w}, \|\tau_x(z) - \pi_i\| = \|\tau_x(z) - \pi_j\| = \delta\}$$

Analogously, a *vertex-edge (VE) critical transformation* $\mathcal{T}_\delta^{ve}(\pi_i, \overline{\sigma_w})$ is the set of all points $x \in \mathbb{R}^k$ such that the segment $\tau_x(\overline{\sigma_w})$ lies on the boundary of the d-ball centred at π_i. It is formally defined as follows.

$$\mathcal{T}_\delta^{ve}(\pi_i, \overline{\sigma_w}) = \{x \in \mathbb{R}^k \mid \exists z \in \overline{\sigma_w}, \|\tau_x(z) - \pi_i\| = \delta\}$$

Every critical transformation is a semi-algebraic set in $\mathbb{R}^k$. Using $\mathcal{O}(n^3)$ VVE critical transformations and $\mathcal{O}(n^2)$ VE critical transformations, Wenk [141, Proof of Theorem 8] showed that one can build an arrangement $\mathcal{A}_\delta$ in $\mathbb{R}^k$ in time $\mathcal{O}(n^{3k})$ and using $\mathcal{O}(n^{3k})$ space. Furthermore, $\mathcal{A}_\delta$ contains at most $\mathcal{O}(n^{3k})$ k'-dimensional faces for $0 \leq k' \leq k-1$. Let $\tau_F = \tau_x$ be the transformation represented by an arbitrary parameter $x \in F$. In [141, Lemma 24], Wenk showed that if there exists a transformation τ such that $d_{\mathcal{F}}(\pi, \tau(\sigma)) < \delta$, then there

exists some k'-dimensional face $F \in \mathcal{A}_\delta$ such that for any τ_F, we have $d_{\mathcal{F}}(\pi, \tau_F(\sigma)) \leq \delta$. Their results are summarised as follows.

FACT 18. *Given a pair of polygonal curves π and σ, a real number $\delta \geq 0$, and a class $\mathcal{T}$ of transformations that is rationally represented with k degrees of freedom, one can build an arrangement $\mathcal{A}_\delta = \mathcal{A}_\delta(\pi, \sigma, \mathcal{T})$ using at most $\mathcal{O}(n^3)$ VVE critical transformations and $\mathcal{O}(n^2)$ VE critical transformations. The arrangement $\mathcal{A}_\delta$ has total complexity $\mathcal{O}(n^{3k})$, and it can be constructed in $\mathcal{O}(n^{3k})$ time using $\mathcal{O}(n^{3k})$ space. To determine whether there exists a transformation $\tau \in \mathcal{T}$ such that $d_{\mathcal{F}}(\pi, \tau(\sigma)) \leq \delta$, it is sufficient to check exactly one transformation τ_F for every k'-dimensional face $F \in \mathcal{A}_\delta$, where $0 \leq k' \leq k - 1$.*

Once the arrangement $\mathcal{A}_\delta$ is constructed, the remainder of the algorithm in Wenk [141] is straightforward. For every face $F \in \mathcal{A}_\delta$, sample a point $x \in F$ and determine whether $d_{\mathcal{F}}(\pi, \tau_x(\sigma)) \leq \delta$ using a classic algorithm (Alt and Godau [7], for example). In total, this takes $\widetilde{\mathcal{O}}(n^{3k} \cdot n^2)$ time.

To obtain a running-time improvement, we use an approach similar to the previous sections. We generate a complete set of events as follows. Initialise an empty graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. For every face $F \in \mathcal{A}_\delta$, add a vertex v_F to $\mathcal{V}$. For every two adjacent faces F and F', add an edge $(v_F, v_{F'})$ to $\mathcal{E}$. For each vertex v_F, record a transformation τ_F.

Next, we compute a complete set of events using $\mathcal{G}$. Initialise an empty event set T. Then perform a DFS to compute a spanning tree of $\mathcal{G}$, and carry out an Euler tour of the spanning tree starting from an arbitrary vertex. For each directed edge $e = (v_{F'}, v_F)$ in the tour, we add an event only if we enter or leave a critical transformation. More specifically, let $B(F)$ be the set of critical transformations adjacent to face F. If $B(F) \setminus B(F') = \{\mathcal{T}_c\}$, we say that e *traverses into* the critical transformation $\mathcal{T}_c$ via F. If $B(F') \setminus B(F) = \{\mathcal{T}_c\}$, we say that e *traverses out of* the critical transformation $\mathcal{T}_c$ via F.

Let $\mathbf{b}(p)$ be the d-sphere of radius δ centred at p. Depending on whether e traverses into or out of a VVE or VE critical transformation, we add the respective freespace events defined in Definition 3.

(1) If $\mathcal{T}_c$ is a VVE critical transformation $\mathcal{T}_\delta^{vve} = \mathcal{T}_\delta^{vve}(\pi_i, \pi_j, \overline{\sigma_w})$, we compute $p = \mathbf{b}(\pi_i) \cap \tau_F(\overline{\sigma_w})$ and $q = \mathbf{b}(\pi_j) \cap \tau_F(\overline{\sigma_w})$, and append an event t defined by $(p, q, \overline{\sigma_w})$ to T. If e traverses into $\mathcal{T}_c$, t is an overlapping event. If e traverses out of $\mathcal{T}_\delta^{vve}$, t is a separating event.

(2) If $\mathcal{T}_c$ is a VE critical transformation $\mathcal{T}_\delta^{ve} = \mathcal{T}_\delta^{ve}(\pi_i, \overline{\sigma_w})$, we compute $p = \mathbf{b}(\pi_i) \cap \overline{\sigma_w}$, and we append an event t represented by $(p, \overline{\sigma_w})$ to T. If e traverses into $\mathcal{T}_c$, t is an entering event if $p = \pi_i$, or an appearing event if $p \neq \pi_i$. Analogously, if e traverses out of $\mathcal{T}_c$, t_e is a leaving event if $p = \pi_i$, or a disappearing event if $p \neq \pi_i$.

We say an event t is associated with the critical transformation $\mathcal{T}_c$ and the edge e if t is computed from e traversing into or out of $\mathcal{T}_c$. We show that T has two desired properties.

LEMMA 19. *Given the arrangement $\mathcal{A}_\delta = \mathcal{A}_\delta(\pi, \sigma, \mathcal{T})$, one can compute a complete set $T = \{t_1, \ldots, t_{\mathcal{O}(n^{3k})}\}$ of freespace events in $\mathcal{O}(n^{3k})$ time with the following properties:*

(1) *Every face in $\mathcal{A}_\delta$ is associated with at least one event in T.*
(2) *For each event t_i associated with an edge $(v_F, v_{F'}) \in \mathcal{E}$, the graphs $\mathcal{G}_\delta^f(\pi, \tau_F(\sigma))$ and $\mathcal{G}_\delta^f(\pi, \tau_{F'}(\sigma))$ differ by exactly one freespace event.*

PROOF. Note that $\mathcal{G}$ is connected and contains a vertex v_F for every face $F \in \mathcal{A}_\delta$. The Euler tour that we constructed over $\mathcal{G}$ visits every vertex of $\mathcal{G}$, and hence every face of $\mathcal{A}_\delta$, at least once. Moreover, a pair of adjacent faces in $\mathcal{A}_\delta$ is separated by exactly one critical transformation, which we recall is a semi-algebraic set in $\mathbb{R}^k$. Therefore, any edge $(v_F, v_{F'})$ in the Euler tour traverses exactly one critical transformation—namely, the critical transformation separating the adjacent faces F and F'.

We next argue that the freespace graph does not change unless a freespace event occurs. It is clear that, unless an appearing or disappearing event occurs, neither the vertices nor the edges in a freespace graph $\mathcal{G}^f$ change. What remains is to argue that the vertex weights do not change unless an event occurs. More specifically, unless a freespace event occurs, no intersection $a = l(p) \cap l(q)$ enters or leaves the freespace.

For the sake of contradiction, suppose that a either enters or leaves the freespace while no freespace event occurs. Clearly, $\mathbf{v}(p, q)$ cannot be a corner vertex, since such a weight change is explicitly captured by an entering or leaving event. Moreover, $\mathbf{v}(p, q)$ cannot be an interior vertex, because every interior vertex has weight 1 regardless.

Therefore, $\mathbf{v}(p, q)$ must be a boundary vertex, and a must be a critical point. Without loss of generality, let p be a vertex of π. If a enters the freespace, then a must coincide with a critical point $b = l(p') \cap l(q)$, implying that the grid lines $l(p)$ and $l(p')$ overlap. If a leaves the freespace, then a must first coincide with (again) such a critical point b, implying that the grid lines $l(p)$ and $l(p')$ separate. In both cases, either an overlapping event or a separating event occurs, contradicting our assumption.

Let p (resp. q) be a critical point on the ith (resp. jth) freespace boundary. We next argue that no freespace event occurs unless we traverse into or out of a critical transformation. This is true by definition. An entering/appearing (resp. leaving/disappearing) event defined by $(p, \overline{\sigma_w})$ occurs only when the associated edge $(v_F, v_{F'})$ traverses into (resp. out of) the critical transformation $\mathcal{T}_\delta^{ve}(\pi_i, \overline{\sigma_w})$. An overlapping (resp. separating) event defined by $(p, q, \overline{\sigma_w})$ occurs only when $(v_F, v_{F'})$ traverses into (resp. out of) the critical transformation $\mathcal{T}_\delta^{vve}(\pi_i, \pi_j, \overline{\sigma_w})$.

Observe that exactly one freespace event occurs every time we traverse into or out of a critical transformation, and this is captured by the Euler tour. Therefore, every face is associated with at least one event in T, and for every edge $(v_F, v_{F'})$ associated with an event $t \in T$, the graphs $\mathcal{G}^f(\pi, \tau_F(\sigma))$ and $\mathcal{G}^f(\pi, \tau_{F'}(\sigma))$ differ by exactly one event. □

We also observe that fewer faces are adjacent to VE critical transformations than to VVE critical transformations.

OBSERVATION 20. *In the arrangement $\mathcal{A}_\delta = \mathcal{A}_\delta(\pi, \sigma, \mathcal{T})$, the number of faces adjacent to VVE critical transformations is at most $\mathcal{O}(n^{3k})$, whereas the number of faces adjacent to VE critical transformations is at most $\mathcal{O}(n^{3k-1})$.*

PROOF. Wenk [141] proved that there are at most $\mathcal{O}(n^{3k})$ faces in $\mathcal{A}_\delta$, which upper-bounds the number of faces adjacent to VVE critical transformations. They also showed that each critical transformation is a semi-algebraic set with constant description complexity [141, Lemma 24]. Therefore, every face in $\mathcal{A}_\delta$ is the intersection of at most k critical transformations. If a face is adjacent to a VE critical transformation, there are at most $n^{3(k-1)}$ ways to choose the remaining $k-1$ critical transformations from the $\Theta(n^3)$ total. Since there are at most n^2 VE critical transformations, in total at most $\mathcal{O}(n^{3k-1})$ faces are adjacent to VE critical transformations. □

We are now ready to put the pieces together to obtain a faster algorithm for computing the Fréchet distance under rationally parameterised transformations. Given a real number $\delta \geq 0$, a class $\mathcal{T}$ of transformations rationally represented with k degrees of freedom, and a pair of polygonal curves π and σ, each with n vertices, Wenk [141] showed that it is sufficient to construct an arrangement $\mathcal{A}_\delta$ in the parameter space $\mathbb{R}^k$ and to sample a single transformation from each face. In Lemma 19, we showed that this arrangement can be traversed to generate a complete set of $\mathcal{O}(n^{3k})$ freespace events in $\mathcal{O}(n^{3k})$ time.

Our results in Sections 2.3 and 2.4 for handling freespace events under translations extend directly to the case of rationally parameterised transformations. Therefore, the number of updates produced by the complete event sequence can be plugged into Lemma 15. Altogether, our algorithm runs in $\mathcal{O}(n^4 + n^{3k} \cdot (T_u(n^2) + T_q(n^2)))$ time, where $T_u(n^2)$ denotes the time to update a vertex and $T_q(n^2)$ the time to query st-reachability in an $\mathcal{O}(n^2) \times \mathcal{O}(n^2)$ grid graph. For a fast query bound $T_q(n^2)$, we again employ the offline dynamic grid-reachability result of [26], which we restate for convenience below.

FACT 21 ([26, Theorem 3.4], restate of Fact 16). *Offline dynamic grid reachability can be solved in time* $\mathcal{O}(N^2 + UN^{2/3} \log^2 N)$.

By Fact 21, $T_u(n^2) + T_q(n^2)$ takes amortised $\mathcal{O}(n^{4/3} \log^2 n)$ time. For classes of rationally parameterised transformations with at least one degree of freedom, we state our final result.

THEOREM 2. *The Fréchet distance under transformations rationally represented with k degrees of freedom can be decided in $\mathcal{O}(n^{3k+4/3}\log^2 n)$ time.*

2.6 Conclusion and future work

Our algorithm provides the first progress in over 20 years for computing the (continuous) Fréchet distance under transformations. The running time comes from traversing an arrangement in the space of transformations, performing an update to a dynamic grid graph data structure in each step.

We conclude with open questions. Can the update time be reduced? Improving the update time for the data structure of [26] would directly improve our running time. But it may also be possible to tailor the data structure for our setting. For instance, we could prune the lower levels of the data structure, since reachability in the inside of freespace cells does not carry any information.

Can we prove a non-trivial conditional lower bound for computing the Fréchet distance under translations? The complexity of the arrangement, $\Omega(n^6)$, would seem like a natural lower bound. However, even transferring the $n^{4-o(1)}$ conditional lower bound for the discrete Fréchet distance under translation [26] to the (continuous) Fréchet distance seems difficult, since the lower bound construction crucially relies on the fact that the discrete Fréchet traversal can only stay on the vertices and not on the edges.

CHAPTER 3

Faster Algorithms for Fréchet Edit Distance

3.1 Introduction

Determining the similarity of two trajectories is a central problem in numerous domains — ranging from geographic mapping to activity recognition and robotics. In computational geometry, the *Fréchet distance* is arguably the most popular metric for measuring curve similarity: it models two entities traversing the trajectories (e.g., a person and a dog) and captures the minimum leash length required to keep them connected under continuous motion. Since the classic algorithm by Alt and Godau [7], extensive progress has been made. Recent works have improved the exact algorithm [55, 56], approximation algorithms [57, 61, 105, 106], query structures [11, 29, 40, 45, 53, 74, 75, 88, 97], algorithm engineering [14, 27, 28, 37, 67], simplification [24, 54, 115, 118], clustering [30, 39, 43, 95, 127], and several other Fréchet-based similarity measurements [38, 42, 66, 73, 77]. Lower bounds have also been a subject of extensive research [23, 32, 56].

Unfortunately, the Fréchet distance is sensitive to outliers. For a curve π, an outlier point $p \in \pi$ that lies far from the remaining points may cause the Fréchet distance between π and another curve σ to be dominated by the distance from p to σ. Real-world data sets, such as GPS traces, eye-tracking movement, and motion capture feeds, often suffer from outliers.

To address this, several robust variants of the Fréchet distance have been suggested. The k-outlier Fréchet distance [44] allow us to ignore up to k points and compute the minimum Fréchet distance for all such deletions. The shortcut Fréchet distance [62] enables skipping subcurves on one curve via shortcuts to mitigate local distortions. Partial curve matching [31]

relaxes the requirement for full alignment by matching only subcurves. Fréchet distance under uncertainty [38] uses points in a curve as uncertainty regions.

In 2024, Fox, Nayyeri, Perry, and Raichel [79] introduced the *Fréchet Edit Distance* (FED), a similarity measure that augments the classical Fréchet distance with edit operations, specifically, insertions and deletions of vertices. They considered a range of variants, depending on whether the discrete or continuous Fréchet distance is used, whether edits are allowed on one or both curves, and whether the edits are restricted to insertions, deletions, or both. For many of these settings, they provided polynomial-time algorithms, and they also established NP-hardness for certain variants, including the case where both insertions and deletions are permitted under the weak Fréchet metric.

In [79], the central insight was to model the problem as a purely combinatorial one, largely avoiding the use of geometric structure. In contrast, we take a fundamentally different approach: our key technical contribution is a detailed study of the problem within the freespace diagram framework of Alt and Godau [7]. We show that the freespace diagram can be scanned greedily while maintaining only a small number of candidate paths. Moreover, we prove that these paths can be maintained using purely local updates during the scan. When combined with geometric properties of the underlying curves (see e.g. Lemma 50), this approach leads to faster algorithms and naturally extends to more general settings, including insert-only operations on both curves for which no result existed previously.

Using this framework, we present improved algorithms for computing the continuous FED under a range of edit models, summarised in Table 3.1. When deletions are allowed on either one or both curves, we obtain an $\widetilde{\mathcal{O}}(kn^2)$-time algorithm (Theorem 6), together with a near-matching lower bound (Section 3.3.5), where $\widetilde{\mathcal{O}}(\cdot)$ suppresses polylogarithmic factors in k and n.

For the insertion-only setting, we show that FED can be solved in $\widetilde{\mathcal{O}}(kn^3)$ time, regardless of whether insertions are allowed on one curve or on both (Theorems 8 and 10). Finally, when both insertions and deletions are allowed on a single curve, we give a $\widetilde{\mathcal{O}}(k^2n^3)$-time algorithm

(Section 3.4.5). In general, our results improve the previously best-known bounds by up to a factor of kn.

	Edit on one curve		Edit on both curves	
	Ours	[79]	Ours	[79]
Delete only	$\widetilde{\mathcal{O}}(kn^2)$	$\widetilde{\mathcal{O}}(k^2n^2)$	$\widetilde{\mathcal{O}}(kn^2)$	$\widetilde{\mathcal{O}}(k^4n^2)$
Insert only	$\widetilde{\mathcal{O}}(kn^3)$	$\widetilde{\mathcal{O}}(n^4(k^2+n))$	$\widetilde{\mathcal{O}}(kn^3)$	—
Insert and delete	$\widetilde{\mathcal{O}}(k^2n^3)$	$\widetilde{\mathcal{O}}(kn^4(k^2+n))$	—	—

TABLE 3.1: A comparison between the running times of our algorithms and the algorithms in [79], where n is the total number of vertices and k is the number of allowed edits.

3.2 Preliminaries

A d-dimensional polygonal curve $\pi : [1, n] \to \mathbb{R}^d$ is a continuous mapping from $[1, n]$ to $\mathbb{R}^d$ such that for any valid integer i, $\pi[i, i+1]$ is a linear function, $\pi_i = \pi[i]$ is a vertex and $\pi_i\pi_{i+1} = \pi[i, i+1]$ is a segment. The integer n is the number of vertices in π, and we say $|\pi| = n$.

Let n and m be the number of vertices in the polygonal curves π and σ, respectively. We use $d_{\mathcal{F}}(\pi, \sigma)$ to denote the Fréchet distance between π and σ (see Chapter 1 for an overview). Let $\mathcal{F}_\delta(\pi, \sigma)$ be the δ-freespace between π and σ, and we call $\mathcal{F}'_\delta(\pi, \sigma) = [1, n] \times [1, m] \setminus \mathcal{F}_\delta(\pi, \sigma)$ the non-free space.

$$\mathcal{F}_\delta(\pi, \sigma) = \{(x, y) \in [1, n] \times [1, m] \mid \|\pi[x] - \sigma[y]\| \leq \delta\}.$$

Let $\mathcal{D}_\delta(\pi, \sigma)$ denote the δ-freespace diagram, which is a partition of $\mathcal{F}_\delta(\pi, \sigma)$ into $(|\pi| - 1) \cdot (|\sigma| - 1)$ cells. We have the following classic result by Alt and Godau [7].

FACT 1. *Let π and σ be curves in $\mathbb{R}^d$ with $|\pi| = n$ and $|\sigma| = m$. Then:*

(1) *The freespace diagram $\mathcal{D}_\delta(\pi, \sigma)$ contains $\mathcal{O}(mn)$ cells and can be constructed in $\mathcal{O}(dmn)$ time.*
(2) *The freespace within each cell is convex and has constant description complexity.*

(3) $d_{\mathcal{F}}(\pi, \sigma) \leq \delta$ *if and only if there exists an* xy*-monotone path in the freespace* $\mathcal{F}_\delta(\pi, \sigma)$ *from* $(1, 1)$ *to* (n, m).

We say two curves π and σ are *similar* if $d_{\mathcal{F}}(\pi, \sigma) \leq \delta$, and a path P in the freespace diagram $\mathcal{D}_\delta(\pi, \sigma)$ is *feasible* if P is an xy-monotone path traversing through the freespace $\mathcal{F}_\delta(\pi, \sigma)$. By Fact 1, $d_{\mathcal{F}}(\pi, \sigma) \leq \delta$ if and only if there exists a feasible path P from $(1, 1)$ to (n, m). The feasible path P corresponds to a continuous matching $M\colon [1, n] \to [1, m]$, which maps a point on π to a point on σ. We say M (or P) realises $d_{\mathcal{F}}(\pi, \sigma) \leq \delta$.

3.3 Delete only

We will start with the model where k vertices can be deleted, either from one curve or from both curves. When deleting a vertex π_i, we remove vertex π_i as well as segments $\pi_{i-1}\pi_i$ and $\pi_i\pi_{i+1}$, and adjacent vertices π_{i-1} and π_{i+1} are connected by a segment. When deleting π_1, π_2 is set as the new starting vertex, and when deleting π_n, π_{n-1} is set as the new ending vertex. A k*-deletions curve* of π is a curve obtained by deleting k vertices from π. If π is the original curve of the deletion curve π', in which consecutive vertices $\{\pi_{i+1}, \pi_{i+2}, ..., \pi_{j-1}\}$ are deleted, then π' maps the interval $[i, j]$ to the segment $\pi_i\pi_j$. For completeness, we define π to be its own 0-deletion curve.

The delete-only Fréchet edit distance (FED) problem asks for the minimum number of vertex deletions required to bring two curves within a given Fréchet distance threshold. The decision version of the delete-only FED problem asks the following.

PROBLEM 1 (Delete-only FED problem). *Given two polygonal curves* π *and* σ*, a non-negative real number* δ*, and a non-negative integer* k*, determine whether there exists a deletion curve* π' *of* π *and a deletion curve* σ' *of* σ *such that* $d_{\mathcal{F}}(\pi', \sigma') \leq \delta$ *and* $|\pi| + |\sigma| - |\pi'| - |\sigma'| \leq k$.

We denote the delete-only Fréchet edit distance between π and σ by $\mathrm{Ded}_\delta(\pi, \sigma)$: the minimum number of vertex deletions required to obtain deletion curves π' and σ' such that $d_{\mathcal{F}}(\pi', \sigma') \leq \delta$. We are also interested in the case in which deletions are allowed only from π.

Let $\text{Ded}^1_\delta(\pi, \sigma)$ denote the minimum number of vertex deletions required from π such that $d_{\mathcal{F}}(\pi', \sigma) \leq \delta$. It is possible that no deletions can bring π and σ within Fréchet distance δ, and they have a delete-only FED distance ∞.

In this section, we will present more general deciders for the delete-only FED problem. To simplify the presentation, we impose the restriction that π_1, π_n, σ_1, and σ_m are not allowed to be deleted. Our approach can easily be generalised to handle these special cases (see Section 3.3.4). This restriction allows us to assume that $\|\pi_1 - \sigma_1\| \leq \delta$ and $\|\pi_n - \sigma_m\| \leq \delta$. This restriction also implies that both (n, m) and $(1, 1)$ lie in the freespace determined by any deletion curve of π and any deletion curve of σ.

3.3.1 Delete-only distance between a segment and a polygonal curve

In this section, we study the restricted case of computing the delete-only edit distance between a segment and a polygonal curve. The observations and algorithms developed here will later enable us to solve the delete-only edit distance problem between two polygonal curves. To this end, we first introduce notation for the cells defined by two deletion curves—more general than what is strictly required for the segment-to-curve case.

For integers $i < j$, let π' be a deletion curve of π, where consecutive vertices $\pi_{i+1}, ..., \pi_{j-1}$ (if they exist) are deleted but vertices π_i and π_j are retained. For integers $a < b$, let σ' be a deletion curve of σ, where consecutive vertices $\sigma_{a+1}, ..., \sigma_{b-1}$ (if they exist) are deleted but σ_a and σ_b are retained. Note that $\pi_i\pi_j$ is a segment in π' and $\sigma_a\sigma_b$ is a segment in σ'.

We discuss the connection between the freespace diagrams $\mathcal{D}_\delta(\pi', \sigma')$ and $\mathcal{D}_\delta(\pi, \sigma)$ by defining the notations for a cell in the freespace diagram $\mathcal{D}_\delta(\pi', \sigma')$ (see Figure 3.1, left). Let $C = C_{(i,j),(a,b)}$ be the freespace diagram cell in $\mathcal{D}_\delta(\pi', \sigma')$, determined by segments $\pi_i\pi_j$ in π' and $\sigma_a\sigma_b$ in σ'. Let $L_{i,(a,b)}$ be the left boundary of C, and let $B_{(i,j),a}$ be the bottom boundary. Let $L^f_{i,(a,b)} \subseteq L_{i,(a,b)}$ and $B^f_{(i,j),a} \subseteq B_{(i,j),a}$ be the portions that lie in the freespace. Finally, let $\mathbf{b}(p, \delta)$ denote the disk centred at point p with radius δ; for brevity, we write $\mathbf{b}(p) = \mathbf{b}(p, \delta)$.

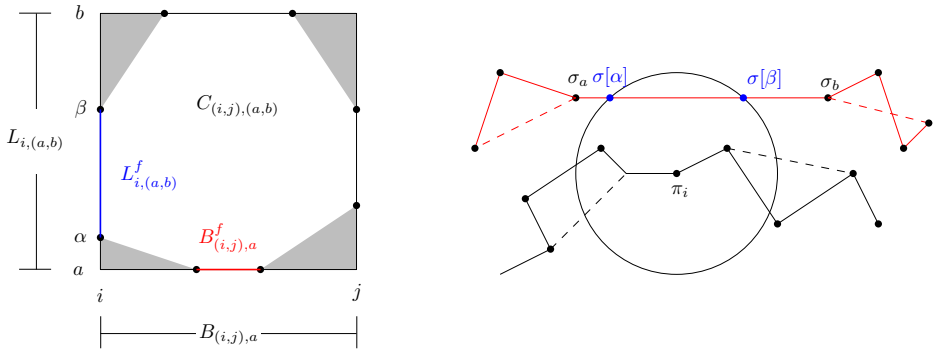

FIGURE 3.1: The left figure shows the cell in the freespace diagram determined by segments $\pi_i\pi_j$ and $\sigma_a\sigma_b$. In the right figure, we observe that $I = [i] \times [\alpha, \beta]$ lies in the freespace $\mathcal{F}_\delta(\pi', \sigma')$, where π' can be any deletion curve of π that retains π_i, and σ' can be any deletion curve of σ that contains the segment $\sigma_a\sigma_b$.

Next, we establish an observation and a lemma that provide additional structure to the problem, showing that it suffices to consider only the information along the boundaries of the freespace diagram.

OBSERVATION 22. *For an integer i and real numbers $\alpha \leq \beta$, if $I = [i] \times [\alpha, \beta]$ lies in $\mathcal{F}_\delta(\pi, \sigma)$, then for any deletion curve π' of π that retains π_i, I lies in $\mathcal{F}_\delta(\pi', \sigma)$.*

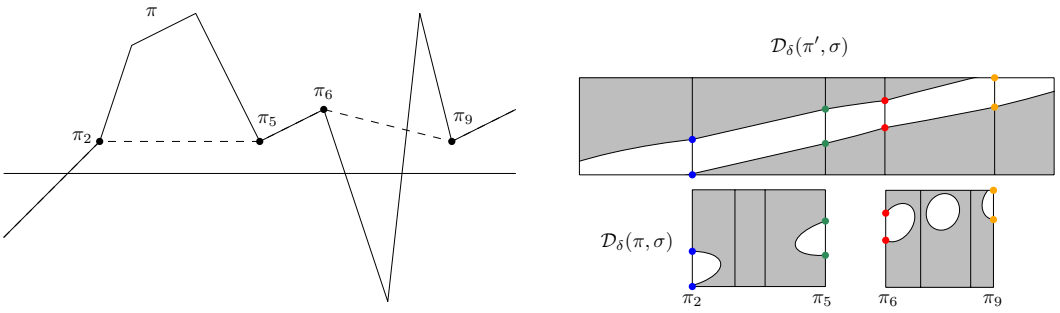

FIGURE 3.2: The left figure shows a horizontal line segment together with a polygonal curve π and a deletion curve of π obtained by deleting the vertices between π_2 and π_5 and between π_6 and π_9. The right figure shows the freespace diagrams between the segment and π, and between the segment and the deletion curve. The coloured critical points on the vertical boundaries defined by π_2, π_5, π_6, and π_9 coincide in both freespace diagrams.

PROOF. See Figure 3.1, right, and 3.2 for an illustration. The interval I lies in the freespace $\mathcal{F}_\delta(\pi, \sigma)$ if the subcurve $\sigma[\alpha, \beta]$ lies entirely in the disk $\mathbf{b}(\pi_i)$. The right-hand side remains true for any deletion curve π' that retains π_i. □

LEMMA 23. *Consider two points $p = (i, \alpha)$ and $q = (j, \beta)$ in $\mathcal{F}_\delta(\pi, \sigma)$, where $p \in L^f_{i,(a,a+1)}$ and $q \in L^f_{j,(a,a+1)}$ for integers $i < j$ and real numbers $\alpha \leq \beta$. Under the restriction that both π_i and π_j must be retained, let an xy-monotone path from p to q be one that intersects the minimum number of vertical boundaries in the non-free space $\mathcal{F}'_\delta(\pi, \sigma)$, and let this minimum be k. Then* $\text{Ded}^1_\delta(\pi[i, j], \sigma[\alpha, \beta]) = k$.

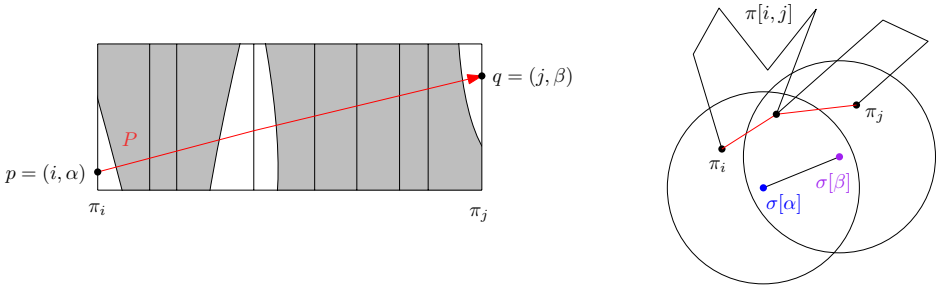

FIGURE 3.3: If a path P intersects vertical boundaries in the non-free space, we can delete the vertices of π that define these boundaries to obtain a deletion curve π' such that $d_{\mathcal{F}}(\pi_i\pi_j, \sigma[\alpha, \beta]) \leq \delta$.

PROOF. See Figure 3.3 for an illustration. We first prove that if there exists an xy-monotone path P from p to q that intersects k vertical boundaries in the non-free space $\mathcal{F}'_\delta(\pi, \sigma)$, then $\text{Ded}^1_\delta(\pi[i, j], \sigma[\alpha, \beta]) \leq k$. Let B be the maximal set of integer indices in increasing order such that for $i' \in B$, $P \cap L_{i',(a,a+1)}$ lies in the freespace $\mathcal{F}_\delta(\pi, \sigma)$ (B includes i and j). Let π' be the k-deletions curve of $\pi[i, j]$ that deletes all vertices π_w for all $w = i + 1, ..., j - 1$ with $w \notin B$.

For every adjacent pair $b, c \in B$, by Observation 22, the intersections $p_b = P \cap L_{b,(a,a+1)}$ and $p_c = P \cap L_{c,(a,a+1)}$ lie in the freespace $\mathcal{F}_\delta(\pi', \sigma)$. Since both p_b and p_c lie in the freespace $\mathcal{F}_\delta(\pi', \sigma)$, by Fact 1, the entire subpath of P from p_b to p_c lies in the freespace. Since this holds for each consecutive pair of such boundaries, P also lies entirely in the freespace. Thus, $d_{\mathcal{F}}(\pi', \sigma[\alpha, \beta]) \leq \delta$, and hence $\text{Ded}^1_\delta(\pi[i, j], \sigma[\alpha, \beta]) \leq k$.

We next prove that if $\text{Ded}^1_\delta(\pi[i, j], \sigma[\alpha, \beta]) = k$, then there exists an xy-monotone path P from p to q that intersects at most k vertical boundaries in the non-free space $\mathcal{F}'_\delta(\pi, \sigma)$. Let $\pi^*[i, j]$ be the k-deletions curve that realises $\text{Ded}^1_\delta(\pi[i, j], \sigma[\alpha, \beta])$ and retains vertices $\pi_{a_0}, ..., \pi_{a_h}$, where $a_0 = i$ and $a_h = j$. By Fact 1, there exists a feasible path $P^* \subseteq \mathcal{F}_\delta(\pi^*, \sigma)$

from p to q. For integer $g = 0, ..., h$, the intersection $p_g = P^* \cap L^f_{a_g,(a,a+1)}$ and the intersection $p_{g+1} = P^* \cap L^f_{a_{g+1},(a,a+1)}$ must lie in the freespace $\mathcal{F}_\delta(\pi^*, \sigma)$ determined by the deletion curve. By Observation 22, both p_g and p_{g+1} must also lie in the freespace $\mathcal{F}_\delta(\pi, \sigma)$ determined by the original curve.

In the worst case, any xy-monotone path $P_g \subseteq \mathcal{D}_\delta(\pi, \sigma)$ from p_g to p_{g+1} intersects all vertical boundaries in the non-free space except for those two determined by π_{a_g} and $\pi_{a_{g+1}}$. Note that $\sum_g (a_{g+1} - a_g - 1) = k$, as the total number of deleted vertices is exactly k. A path P can be constructed by concatenating the paths P_g for all values of g, and this path intersects at most k vertical boundaries in the non-free space. □

Lemma 23 effectively reduces the problem of computing the delete-only Fréchet edit distance between a segment and a curve to finding an xy-monotone path that intersects as few vertical boundaries in the non-free space as possible. We say that a path P *stabs* a vertical cell boundary $L_{i,(a,b)}$ if their intersection $P \cap L_{i,(a,b)}$ lies in the non-free space $L_{i,(a,b)} \setminus L^f_{i,(a,b)}$. The *deletion cost* $\mathbf{w}(p)$ of a point p is the minimum number of vertical boundaries an xy-monotone path must stab to reach p — we say that p is $\mathbf{w}(p)$*-reachable*.

Our approach iterates over i from 1 to n, and we are interested in the minimum deletion cost among all points lying on every $L_{i,(1,2)}$. However, it is insufficient to maintain the minimum deletion cost for each cell boundary. Compared to a best path ending at a point $q \in L_{i,(1,2)}$, a best path ending at a lower point p may stab more vertical boundaries initially, but it has greater freedom to stab fewer vertical boundaries as it continues. For the same reason, if p has a lower deletion cost than a higher point q, there is no need to record q. We say a point p is an *extreme reachable point* if it is the lowest point such that there exists an xy-monotone path from $(1, 1)$ to p that stabs at most $\mathbf{w}(p)$ vertical boundaries.

Motivated by these observations, Algorithm 1 iteratively maintains, for fixed a and b, a set $R_{i,(a,b)}$ of extreme reachable points (henceforth, simply reachable points) on $L_{i,(a,b)}$. Each set of reachable points is *monotone*: when the points are sorted from low to high, their deletion costs decrease. Observe that the reachable points in $R_{i,(a,b)}$ share the same x-coordinate i, and it is sufficient to store $R_{i,(a,b)}$ as a set of real values in the range $[a, b]$.

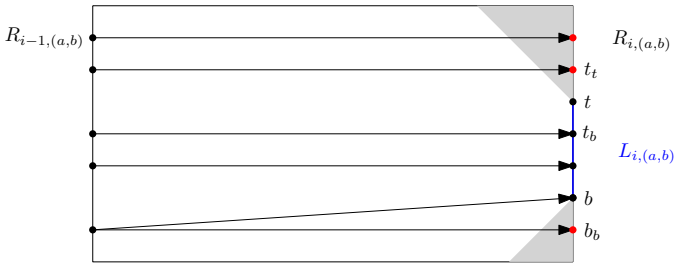

FIGURE 3.4: An illustration of deletion cost updates for points in $R_{i-1,(a,b)}$, the weights of the red reachable points are incremented by 1.

Algorithm 1 Horizontal Propagation. We iterate over i from 1 to n, and compute the set $R_{i,(1,2)}$ of reachable points. We initialise $R_{1,(1,2)}$ to contain a single point $(1, 1)$ with deletion cost 0.

Given that we have computed $R_{i-1,(1,2)}$, we compute $R_{i,(1,2)}$. To avoid needless duplication, we first set $R_{i,(1,2)} = R_{i-1,(1,2)}$ before updating $R_{i,(1,2)}$ using $L^f_{i,(1,2)}$. If $L_{i,(1,2)}$ lies entirely in the freespace, no action is required. If $L_{i,(1,2)}$ lies entirely in the non-free space, increment the deletion cost of every point in $R_{i,(1,2)}$ by 1.

If neither of the above cases occurs, let t and b be the top and bottom points of $L^f_{i,(1,2)}$, respectively (see Figure 3.4). Let b_b (resp. t_b) be the topmost point in $R_{i,(1,2)}$ lower than b (resp. t), and let b_t (resp. t_t) be the bottommost point above b (resp. t). We first insert b in $R_{i,(1,2)}$ and set $\mathbf{w}(b) = \mathbf{w}(b_b)$. Then, for every point $r \in R_{i,(1,2)}$ that is either above t or lower than b, increment $\mathbf{w}(r)$ by 1. As a final step, if $\mathbf{w}(t_t) = \mathbf{w}(t_b)$, we remove t_t from $R_{i,(1,2)}$. If we are solving the decision version, remove the lowest point in $R_{i,(1,2)}$ if it has a deletion cost of $k + 1$. We stop if $R_{i,(1,2)}$ is empty and report $\text{Ded}^1_\delta(\pi, \sigma) > k$. If $R_{n,(1,2)}$ is non-empty, we report $\text{Ded}^1_\delta(\pi, \sigma) \leq k$. To solve the optimisation version, report the deletion cost of the topmost point in $R_{n,(1,2)}$ as $\text{Ded}^1_\delta(\pi, \sigma)$.

LEMMA 24. *Given two polygonal curves π and σ with $|\pi| = n$ and $|\sigma| = 2$, it takes $\mathcal{O}(n \log k)$ time to determine if $\text{Ded}^1_\delta(\pi, \sigma) \leq k$, and $\mathcal{O}(n \log n)$ time to compute $\text{Ded}^1_\delta(\pi, \sigma)$.*

PROOF. To prove the correctness of Algorithm 1, we show that $R_{i,(1,2)}$ contains every extreme reachable point on $L^f_{i,(1,2)}$, for all $i = 1, ..., n$.

We prove this claim using induction. Clearly, $R_{1,(1,2)} = \{s\}$ with $s = (1, 1)$ and $\mathbf{w}(s) = 0$ since the degenerate path (the single point) $(1, 1)$ stabs no vertical boundaries. Given that $R_{i-1,(1,2)}$ is correct, we argue that $R_{i,(1,2)}$ is also correct. For any c-reachable point $(i-1, r) \in R_{i-1,(1,2)}$, due to the monotonicity requirement of a path, $p = (i, r)$ is at most c-reachable if $p \in L^f_{i,(1,2)}$ or at most $(c + 1)$-reachable if $p \notin L^f_{i,(1,2)}$. If $p = b_b$, then b is c-reachable, since $R_{i-1,(1,2)}$ is monotone, and the higher point is c'-reachable for some smaller c'. Once these steps have taken place in Algorithm 1, we remove t_t if $\mathbf{w}(t_t) = \mathbf{w}(t_b)$ because they are the only pair that may have the same deletion cost.

We next analyse the time complexity. If we are solving the decision version, we are required to keep only the reachable points that are at most k-reachable. Therefore, $|R_{i,(1,2)}| \leq k + 1$, and it takes $\mathcal{O}(\log k)$ time to compute $R_{i,(1,2)}$ from $R_{i-1,(1,2)}$. We use a balanced binary search tree T to store the reachable points in increasing order. When computing $R_{i,(1,2)}$, the deletion costs for a continuous subset $R' \subseteq R_{i,(1,2)}$ of points are either unchanged or incremented by 1. To avoid explicitly maintaining the deletion costs of $\mathcal{O}(k)$ points, for each node of T, we store a counter initialised to 0. When incrementing the deletion costs of points in R', we first compute a minimal subset T' of nodes whose canonical subset stores R'. Then we increment the counters stored at each node of T' by 1. The size of T' is at most $\mathcal{O}(\log k)$.

During the update, our algorithm queries the exact deletion costs of at most three points: the lowest point in $R_{i,(1,2)}$, t_b and b_b, and updates at most two points: the lowest point in $R_{i,(1,2)}$ and b. These operations involve a constant number of nodes, and they take a total of $\mathcal{O}(\log k)$ time since the depth of T is at most $\mathcal{O}(\log k)$. The decision algorithm in total takes $\mathcal{O}(n \log k)$ time due to the n vertical boundaries for which we perform the updates. The optimisation version takes $\mathcal{O}(n \log n)$ time, since $|R_{i,(1,2)}| \leq n$. □

3.3.2 Deletions from one of the two curves

Algorithm 1 does not trivially extend to the case where $|\sigma| > 2$. The algorithm uses Observation 22, and computes the deletion costs of at most $k + 1$ points one vertical boundary

at a time. However, when $|\sigma| > 2$, a horizontal cell boundary lying in the non-free space may lie in the freespace once a set of vertices is deleted. See Figure 3.5 for an example.

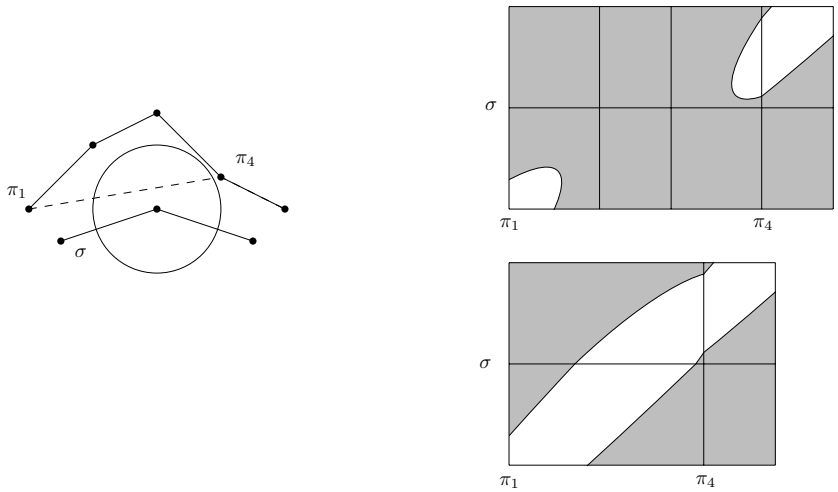

FIGURE 3.5: By removing π_2 and π_3, part of the horizontal boundary now lies in the freespace.

From a freespace diagram perspective, consider an xy-monotone path P^* through $\mathcal{D}_\delta(\pi^*, \sigma)$, where π^* is an optimal deletion curve that realises $\mathrm{Ded}^1_\delta(\pi, \sigma)$. Gudmundsson and Wong [95] observed that P^* can be transformed and partitioned into two types of subpaths — a subpath either lies in a single row or crosses at least one horizontal boundary. We need only a weaker version of their result.

FACT 25 ([95, Lemma 19 and Theorem 22]). *Let P be an xy-monotone path in the freespace diagram $\mathcal{D}_\delta(\pi', \sigma)$, where π' is a deletion curve such that $d_{\mathcal{F}}(\pi', \sigma) \leq \delta$. P can be transformed and partitioned into a set of maximal subpaths such that for some valid integers $i < j$ and $a < a'$, each subpath is exactly one of the following two types:*

(1) ***Row subpath.*** *The subpath lies entirely within a single row determined by a segment $\sigma_a\sigma_{a+1}$. It starts at the lowest point on $L^f_{i,(a,a+1)}$ and ends at a point on $L^f_{j,(a,a+1)}$.*
(2) ***Column subpath.*** *The subpath lies entirely within a single column determined by a segment $\pi_i\pi_j$. It starts at a point on $L^f_{i,(a,a+1)}$ and ends at the lowest point on $L^f_{j,(a',a'+1)}$.*

The existence of a row subpath requires us to compute the edit distance between a subsegment of σ and a subcurve of π, which is done in Algorithm 1. The existence of a column subpath requires us to compute additional horizontal boundaries. For fixed integers i and a, we compute $B_{(i,j),a}$ for all j only if $j - i - 1 \leq k$, since at most k deletions are allowed.

For a point q lying on some $L^f_{j,(a',a'+1)}$, we want to compute the point $p \in L^f_{i,(a,a+1)}$ with lowest deletion cost such that there exists an xy-monotone path through $\mathcal{F}_\delta(\pi_i\pi_j, \sigma)$ from p to q. Naturally, any column subpath reaching $L^f_{j,(a,a+1)}$ must first arrive at some $B^f_{(i,j),a'}$, for which we pay an additional $j - i - 1$ deleted vertices. This information is computed efficiently using Algorithm 2 and its correctness is proved in Lemma 26.

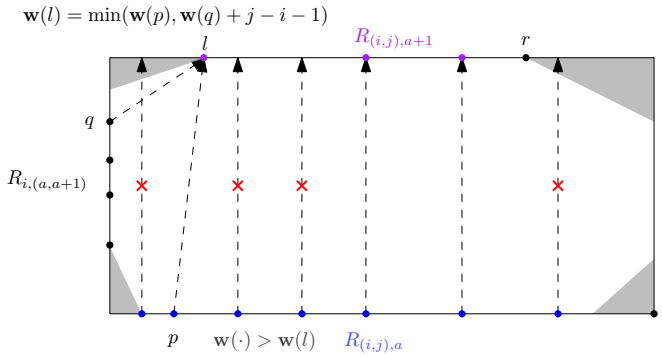

FIGURE 3.6: An illustration of Algorithm 2.

Algorithm 2 Vertical Propagation. See Figure 3.6 for an illustration. Given every set $R_{i,(a,a+1)}$ of monotone reachable points as the input, we first initialise $R_{(i,j),1}$ as an empty set, and then iterate $a = 1, ..., m - 1$ and compute $R_{(i,j),a+1}$ in the following manner.

Let l and r be the leftmost and rightmost points on $B^f_{(i,j),a+1}$, respectively. Store points p and q, if each exists: $p \in R_{(i,j),a}$ is the rightmost point that lies to the left[1] of l, and q is the topmost point on $L^f_{i,(a,a+1)}$. Set $R_{(i,j),a+1} = R_{(i,j),a}$.

First, remove the points in $R_{(i,j),a+1}$ that are to the right of r or to the left of l. Next, if neither p nor q exists, we are done. Otherwise, let $c' = \min\{\mathbf{w}(p), \mathbf{w}(q) + j - i - 1\}$ and remove all points in $R_{(i,j),a+1}$ whose deletion cost exceeds c'. Finally, insert l into $R_{(i,j),a+1}$ with $\mathbf{w}(l) = c'$.

[1]For brevity, we say that p is to the left (resp. on top) of q if p has a smaller or equal x-coordinate (resp. y-coordinate).

LEMMA 26. *For fixed integers i and j, it takes $\mathcal{O}(m \log k)$ time to compute, for every lowest critical point q on all $L^f_{i,(a,a+1)}$, $1 \leq a \leq m$, the point p with the lowest deletion cost such that there exists an xy-monotone path from p to q through $\mathcal{F}_\delta(\pi_i\pi_j, \sigma)$.*

PROOF. Since $R_{(i,j),a}$ (resp. $R_{i,(a,a+1)}$) is monotone, the points to the right (resp. top) have smaller deletion cost. Therefore, $\min\{\mathbf{w}(p), \mathbf{w}(q)\}$ is the smallest deletion cost for which there exists an xy-monotone path from any point in $R_{(i,j),a} \cup R_{i,(a,a+1)}$ to l.

We now analyse the time complexity. It is sufficient to store the set R of reachable points as an array of sorted real values with decreasing deletion costs. When updating R, we either remove a continuous subset or insert a new point l. All operations can be done with a binary search on an array of size $\mathcal{O}(k)$ and a constant number of index manipulations. □

When σ is a segment, in Algorithm 1, we computed the set $R_{j,(a,a+1)}$ of reachable points using only $R_{j-1,(a,a+1)}$. When σ contains more than one segment, computing $R_{j,(a,a+1)}$ needs to take into consideration the set $R_{(i,j),a}$ of reachable points on the horizontal boundaries, for all valid values of i.

Algorithm 3 Compute $\mathcal{D}_\delta(\pi, \sigma)$. Initialise both $R_{1,(1,2)}$ and $R_{(1,2),1}$ to contain a single point $(1, 1)$ with deletion cost 0. Initialise $R_{1,(a,a+1)}$ to be empty for all $a = 2, ..., m-1$. For all $a = 1, ..., m-1$, $i = 2, ..., n-1$, and $j = i+1, ..., i+k+1$, compute $B_{(i,j),a}$ and initialise $R_{(i,j),1}$ to be empty.

We use Algorithm 2 to compute the set of reachable points on horizontal boundaries. For every $j = 2, ..., n$ and $a = 2, ..., m-1$, we use the following steps to compute $R_{j,(a,a+1)}$ using the set $R_{j-1,(a,a+1)}$, and $R_{(i,j),a}$ for all $i = j-k-1, ..., j-1$. First, compute $R_{j,(a,a+1)}$ using Algorithm 1 with $R_{j-1,(a,a+1)}$ as input. Next, let c be the lowest deletion cost of points in $R_{(i,j),a}$ for all $i = j-k-1, ..., j-1$ and let p be the lowest point on $L^f_{j,(a,a+1)}$. Set $\mathbf{w}(p) = \min\{c, \mathbf{w}(p)\}$, and remove all points above p in $R_{j,(a,a+1)}$ with deletion cost greater than or equal to $\mathbf{w}(p)$.

Report $\text{Ded}^1_\delta(\pi, \sigma) \leq k$ if $R_{n,(m-1,m)}$ is non-empty. To solve the optimisation version, set $k = n$ and report the minimum deletion cost of points in $R_{n,(m-1,m)}$ as $\text{Ded}^1_\delta(\pi, \sigma)$.

LEMMA 27. *Let $R_{n,(m-1,m)}$ be the set of reachable points computed by Algorithm 3 on input curves π, σ, and a threshold δ. Then the minimum deletion cost among the points in $R_{n,(m-1,m)}$ equals* $\mathrm{Ded}^1_\delta(\pi,\sigma)$.

PROOF. We first prove that if $\mathrm{Ded}^1_\delta(\pi,\sigma)=k$, the minimum deletion cost of points in $R_{n,(m-1,m)}$ is at most k. Consider an optimal k-deletion curve π^* of π that realises $\mathrm{Ded}^1_\delta(\pi,\sigma)$, and a feasible path $\mathcal{P}\subseteq\mathcal{F}_\delta(\pi^*,\sigma)$ from $(1,1)$ to (n,m). Using Fact 25, we know there is a partition of $\mathcal{P}$ such that each subpath P_g is either a column subpath or a row subpath.

Consider the case where P_g is a row subpath traversing from $p=(i,\alpha)$ to $q=(j,\beta)$, and π^* deletes k_g vertices between π_i and π_j. By Lemmas 23 and 24, p is a reachable point in $R_{i,(a,a+1)}$ and there is a reachable point q' at or below q with deletion cost $\mathbf{w}(q')\le\mathbf{w}(p)+k_g$.

If P_g is a column subpath traversing from $p\in L^f_{i,(a,a+1)}$ to q, then π^* deletes all vertices between π_i and π_j. By Lemma 26, there exists a reachable point p' at or below p and $\mathbf{w}(q)\le\mathbf{w}(p')+k_g$, where $k_g=j-i-1$, since P_g is an xy-monotone path through the freespace $\mathcal{F}_\delta(\pi_i\pi_j,\sigma)$. In total, the minimum deletion cost of points in $R_{n,(m-1,m)}$ is at most $\sum_g k_g\le k$.

We next prove that if the minimum deletion cost $\mathbf{w}(u)$ of points in $R_{n,(m-1,m)}$ is k, then $\mathrm{Ded}^1_\delta(\pi,\sigma)\le k$ — by backtracking from u to construct a deletion curve π' such that $d_{\mathcal{F}}(\pi',\sigma)\le\delta$. Observe that in Algorithm 3, the deletion cost of a reachable point $q=(j,\beta)$ in $R_{j,(a',a'+1)}$ is realised either by a reachable point p_h through horizontal propagation or a reachable point $p_v=(i,\alpha)$ in $R_{i,(a,a+1)}$ through vertical propagation.

If $\mathbf{w}(q)$ is realised by p_v, then we use the deletion curve $\pi_i\pi_j$ in π' and $d_{\mathcal{F}}(\pi_i\pi_j,\sigma[\alpha,\beta])$ by Lemma 26. If $\mathbf{w}(q)$ is realised by p_h, we continue backtracking from p_h until a point $p=(i,\alpha)\in L^f_{i,(a,a+1)}$ in the freespace. We attach $\pi_i\pi_j$ to π', and by Lemma 23, $d_{\mathcal{F}}(\pi_i\pi_j,\sigma[\alpha,\beta])\le\delta$. Repeat this process until we reach the reachable point $(1,1)$, at which point we have $d_{\mathcal{F}}(\pi',\sigma)\le\delta$. □

LEMMA 28. *Given two curves π and σ with $|\pi|=n$ and $|\sigma|=m$, it takes $\mathcal{O}(kmn\log k)$ time to determine if* $\mathrm{Ded}^1_\delta(\pi,\sigma)\le k$ *and $\mathcal{O}(mn^2\log n)$ time to compute* $\mathrm{Ded}^1_\delta(\pi,\sigma)$.

PROOF. The correctness of Algorithm 3 is shown in Lemma 27, and the algorithm computes the reachable points on $\mathcal{O}(mn)$ vertical cell boundaries and $\mathcal{O}(kmn)$ horizontal cell boundaries. By Lemma 24, it takes $\mathcal{O}(\log k)$ time to first compute $R_{j,(a,a+1)}$ using $R_{j-1,(a,a+1)}$. It takes $\mathcal{O}(\log k)$ time to find the point with minimum deletion cost in $R_{(i,j),a}$ for all $i = j-k-1, ..., j-1$, since the reachable points are monotone and their minima can be stored in a min-heap. It takes $\mathcal{O}(\log k)$ time to compute every $R_{(i,j),a}$ by Lemma 26. In total, Algorithm 3 takes $\mathcal{O}(kmn \log k)$ time. We have $k = n$ for the optimisation version, which takes $\mathcal{O}(mn^2 \log n)$ time. □

Note that we can perform exponential search to obtain a value k that is at most $2 \cdot \mathrm{Ded}^1_\delta(\pi, \sigma)$. Combined with the decision algorithm, this yields an $\mathcal{O}(kmn \log^2 k)$-time optimisation algorithm. This applies to all special cases of the Fréchet edit distance problem discussed in this chapter.

THEOREM 3. *When deletions are allowed on only one of the two curves, the delete-only Fréchet edit distance problem can be solved in $\widetilde{\mathcal{O}}(kn^2)$ time.*

3.3.3 Deletions in both curves

When deletions are allowed on both curves, we compute the sets of reachable points on all valid horizontal and vertical boundaries.

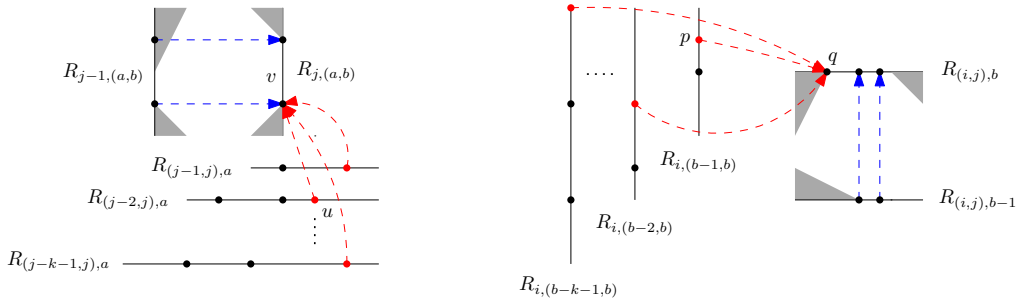

FIGURE 3.7: An illustration of Algorithm 4. The blue arrows illustrate the update by Algorithm 1. The red arrows illustrate taking the minimum deletion cost on the sets of reachable points on the orthogonal boundaries.

Algorithm 4 We iteratively compute the sets $R_{i,(a,b)}$ and $R_{(i,j),a}$ of reachable points for all $1 \le i < j \le n$ and $1 \le a < b \le m$. During initialisation, compute $L_{i,(a,b)}$ and $B_{(i,j),a}$ for all

valid integers $i < j$ and $a < b$. For $b = 2, ..., k + 1$, initialise two sets $R_{1,(1,b)}$ and $R_{(1,b),1}$ of reachable points, both containing a single point $(1, 1)$ with deletion cost $b - 2$. For other sets of reachable points determined by π_1 and σ_1, initialise them to be empty.

Once the initialisation is complete, use Algorithm 1 to compute every set of reachable points on every horizontal boundary or vertical boundary, followed by an update using sets of reachable points on adjacent orthogonal boundaries (see Figure 3.7). Specifically, once $R_{j-1,(a,b)}$ is updated to $R_{j,(a,b)}$ by Algorithm 1, update the deletion cost of v to $\mathbf{w}(v) = \min\{\mathbf{w}(v), \mathbf{w}(u) + b - a - 1\}$, where $v \in R_{j,(a,b)}$ is the lowest point in $L^f_{j,(a,b)}$ and u is the minimum deletion cost in $R_{(i,j),a}$ for all $i = j - k - 1, ..., j - 1$. Analogously, once $R_{(i,j),b-1}$ is updated to $R_{(i,j),b}$ by Algorithm 1, update the deletion cost of q to $\mathbf{w}(q) = \min\{\mathbf{w}(q), \mathbf{w}(p) + j - i - 1\}$, where $q \in R_{(i,j),b}$ is the leftmost point in $B^f_{(i,j),b}$ and p is the point with the minimum deletion cost in $R_{i,(a,b)}$ for all $a = b - k - 1, ..., b - 1$.

With a procedure to compute a set of reachable points, our algorithm iterates over the indices in the following manner. We iterate $a = 1, ..., m - 1$. For each value of a, first compute $R_{i,(a,b)}$ for each $i = 1, ..., n$ and $b = a + 1, ..., a + k + 1$, and next compute $R_{(i,j),a+1}$ for each $i = 1, ..., n$ and $j = i + 1, ..., i + k + 1$.

Once every $R_{(i,n),m}$ has been computed for $i \geq n - k - 1$, and every $R_{n,(a,m)}$ has been computed for $a \geq m - k - 1$, if either set is non-empty, report $\text{Ded}_\delta(\pi, \sigma) \leq k$. For the optimisation version, set $k = n + m$ and report the minimum deletion cost as $\text{Ded}_\delta(\pi, \sigma)$.

LEMMA 29. *Given two curves π and σ with $|\pi| = n$ and $|\sigma| = m$, it takes $\mathcal{O}(kmn \log k)$ time to determine if $\text{Ded}_\delta(\pi, \sigma) \leq k$ and $\mathcal{O}((m^2n + mn^2) \log(m + n))$ time to compute $\text{Ded}_\delta(\pi, \sigma)$.*

PROOF. The correctness follows directly from the same arguments in Lemma 27, and we focus on analysing the running time. In total, the algorithm computes $\mathcal{O}(kmn)$ sets of reachable points. By Lemma 23, each set takes $\mathcal{O}(\log k)$ time to compute and the algorithm takes $\mathcal{O}(kmn \log k)$ time in total. To solve the optimisation version, we set $k = m + n$, and the time complexity is $\mathcal{O}((m + n)mn \log(m + n)) = \mathcal{O}((m^2n + mn^2) \log(m + n))$. □

3.3.4 When deletions are allowed for leading and trailing vertices

We remove the restriction that the leading and trailing vertices are not allowed to be deleted.

Remark 4 In the case that only vertices of π are allowed to be deleted, observe that the first vertex of any deletion curve of π must match to σ_1. Therefore, we claim that for any optimal deletion curve π^* that realises the delete-only Fréchet edit distance, π^* must retain the first vertex π_i of π such that $\|\pi_i - \sigma_1\| \leq \delta$. Indeed, otherwise the curve $\pi_i \circ \pi^*$ is still within Fréchet distance δ from σ since the segment connecting π_i and the first vertex of π^* lies entirely in the disk $\mathbf{b}(\sigma_1, \delta)$, which means that π^* is not optimal. For analogous reasons, π^* must retain the last vertex of π that is within distance δ from σ_m. To compute or determine $\text{Ded}^1_\delta(\pi, \sigma)$, it is sufficient to first identify the first vertex $\pi_i \in \mathbf{b}(\sigma_1)$ and the last vertex $\pi_j \in \mathbf{b}(\sigma_m)$. Then, use the algorithm in Lemma 28 with σ, $\pi[i, j]$, and an updated target $k' = k - (i-1) - (n-j)$ as inputs. It takes $\mathcal{O}(n)$ time to identify π_i and π_j, and the time complexity in Lemma 24 and Lemma 28 still hold.

Remark 5 In the case that deletions are allowed on both curves, let (π^*, σ^*) be a pair of optimal deletion curves that realises the delete-only Fréchet distance between π and σ. Observe that the first vertex of π^* and the first vertex of σ^* must be within distance δ. Therefore, we claim that the first vertex $\pi_i \in \pi^*$ and $\sigma_j \in \sigma^*$ must satisfy the following properties: $\|\pi_i - \sigma_j\| \leq \delta$ and no pair (π_a, σ_b) of vertices are within distance δ, where either $a \leq i$ and $b < j$, or $a < i$ and $b \leq j$. Indeed, as otherwise $d_{\mathcal{F}}(\pi_a \circ \pi^*, \sigma_b \circ \sigma^*) \leq \delta$, and (π^*, σ^*) do not realise $\text{Ded}_\delta(\pi, \sigma)$. If π_i and σ_j are the last vertex of π^* and σ^*, respectively, an analogous property must hold: $\|\pi_i - \sigma_j\| \leq \delta$ and no pair (π_a, σ_b) of vertices are within distance δ, where either $a \geq i$ and $b > j$, or $a > i$ and $b \geq j$.

Let A (resp. B) be the set containing valid pairs of first (resp. last) vertices. To determine if $\text{Ded}_\delta(\pi, \sigma) \leq k$, it is sufficient to run Algorithm 4 using the pairs in A as the starting points and stop at each ending pair in B. Each starting pair (π_i, σ_j) incur an additional deletion cost of $i-1+j-1$, and each ending pair (π_a, σ_b) incur an additional deletion cost of $n-a+m-b$. Identifying A and B takes at most $\mathcal{O}(nm)$ time, and the time complexity in Lemma 29 still holds.

THEOREM 6. *The delete-only Fréchet edit distance problem can be solved in $\widetilde{\mathcal{O}}(kn^2)$ time.*

3.3.5 A cubic lower bound for deletions from one curve

We will perform a reduction from the ∀∀∃-OV problem to the FED problem when deletions are only allowed from one curve. In an instance of the ∀∀∃-OV problem, we are given three sets of vectors A, B, and C. Each set contains n vectors of dimension d. Each entry of a vector is either 0 or 1. We are to answer the following: is it true that for all pairs $a \in A$ and $b \in B$ of vectors, there exists $c \in C$ such that a, b and c are orthogonal? To report 'no' to this question, it is sufficient to do the following: identify at least one pair $(a_i \in A, b_j \in B)$ such that for all $c \in C$, we have $a_i[\ell] = b_j[\ell] = c[\ell] = 1$ for some ℓ. It is conjectured that there is no $\mathcal{O}(n^{3-\varepsilon})$-time algorithm for the ∀∀∃-OV problem for $\varepsilon > 0$ [24, 82].

We say two points a and b are close if $\|a - b\|_p \leq \delta$, and they are far if $\|a - b\|_p > \delta$, where $\|\cdot\|_p$ is the $\mathcal{L}_p$ norm. Let $\mathbf{m}(ab) = \frac{a+b}{2}$ be the midpoint of segment ab. Given an instance (A', B', C') of ∀∀∃-OV and $p \neq 2$, Bringmann and Chaudhury [24, Section 4.1] showed how to compute a threshold δ, a point s, and three sets A, B, and C of d-dimensional points with the following properties.

(1) For any $a \in A$, $b \in B$, $c \in C$, there is a point x on the line segment ab with $\|x - c\|_p \leq \delta$ if and only if $\|\mathbf{m}(ab) - c\|_p \leq \delta$.
(2) For any $a \in A$, $b \in B$, $c \in C$, we have $\|\mathbf{m}(ab) - c\|_p \leq \delta$ if and only if there exists $l \in [d]$ such that $a[l] = b[l] = c[l] = 1$.
(3) For any $p, q \in A$, $p, q \in B$, or $p, q \in C$, $\|p - q\|_p \leq \delta$.
(4) For any $p, q \in \{s\} \cup B \cup C$ and $x \in pq$, $\|x - a\|_p > \delta$ for all $a \in A$.
(5) For any $p, q \in \{s\} \cup A \cup C$ and $x \in pq$, $\|x - b\|_p > \delta$ for all $b \in B$.
(6) For any $p \in A \cup B$ and $x \in sp$, $\|x - c\|_p > \delta$ for all $c \in C$.

LEMMA 30. *Let π and σ be a pair of polygonal curves in high dimension. Let $|\pi| = 2n + 2$ and $|\sigma| = n + 4$. Let the distance between two points be measured in the $\mathcal{L}_p$ metric, where $p \in [1, \infty)$ and $p \neq 2$. Let $\delta > 0$ and $k \in \mathbb{N}$. Conditioned on ∀∀∃-OV, for $\varepsilon > 0$, there is no $\mathcal{O}(n^{3-\varepsilon})$-time algorithm to decide whether* $\mathrm{Ded}^1_\delta(\pi, \sigma) \leq 2n - 2$.

PROOF. Let $\langle p_1, p_2, ..., p_n\rangle$ denote the polygonal curve connecting points $p_1, p_2, ..., p_n$. For the reduction from the $\forall\forall\exists$-OV problem to the FED problem, we set $\pi = \langle s, a_1, ..., a_n, b_1, ..., b_n, s\rangle$, $\sigma = \langle s, a_1, c_1, ..., c_n, b_1, s\rangle$ and $k = 2n - 2$. The decision problem $\text{Ded}^1_\delta(\pi, \sigma) \leq k$ asks whether we can delete at most k vertices from π such that the resulting curve π' is similar to σ.

Suppose that $\text{Ded}^1_\delta(\pi, \sigma) \leq k$. By Properties 4 and 5, we cannot delete the start or end point s from π. By Property 4, we cannot delete the entire set A. By Property 5, we cannot delete the entire set B. Therefore, at least one segment a_ib_j must be in π', since $k = 2n - 2$. By Property 4, the point $a_i \in \sigma$ must match a point in the subcurve $\langle s, a_1, c_1\rangle$. By Property 5, the point $b_j \in \sigma$ must match to a point in the subcurve $\langle c_n, b_1, s\rangle$. Therefore, the subcurve $\langle c_1, ..., c_n\rangle$ must match to a subsegment of $a_ib_j \in \pi'$. By Property 1, every $c_k \in C$ must be close to $\mathbf{m}(a_ib_j)$. By Property 2, for every c_k, there exists $l \in [d]$ such that $a_i[l] = b_j[l] = c_k[l] = 1$ and the triple (a_i, b_j, c_k) is non-orthogonal. We can therefore report that (A', B', C') is a NO-instance of $\forall\forall\exists$-OV.

Suppose that (A', B', C') is a NO-instance of $\forall\forall\exists$-OV. Therefore, there exists a pair $a_i \in A$ and $b_j \in B$ such that the triple (a_i, b_j, c_k) is non-orthogonal for all $c_k \in C$. Delete $k = 2n-2$ vertices from π to obtain the curve $\pi' = \langle s, a_i, b_j, s\rangle$. The Fréchet distance between π' and σ is $\leq \delta$, since s matches to s, a_i matches to a_1 (Property 3), $\mathbf{m}(a_ib_j)$ matches to c_k for all $c_k \in C$ (Property 2), b_j matches to b_1 (Property 3), and s matches to s. Therefore, $\text{Ded}^1_\delta(\pi, \sigma) \leq k$.

Therefore, (A', B', C') is a NO-instance of $\forall\forall\exists$-OV if and only if $\text{Ded}^1_\delta(\pi, \sigma) \leq 2n - 2$. Conditioned on the $\forall\forall\exists$-OV conjecture, there is no $\mathcal{O}(n^{3-\varepsilon})$ time algorithm for $\forall\forall\exists$-OV, so there is also no $\mathcal{O}(n^{3-\varepsilon})$-time algorithm for $\text{Ded}^1_\delta(\pi, \sigma) \leq 2n - 2$. □

To connect the time complexity in Theorem 3 to the cubic lower bound, observe that our algorithm indeed works under any $\mathcal{L}_p$ metric. The geometric observations, Observation 22 and Lemma 23, still hold, and our algorithm primarily operates in the freespace diagram, which easily generalises to any $\mathcal{L}_p$ metric. Lemma 30 provides evidence that Theorem 3 is close to optimal.

3.4 Edits on one of the two curves

Above we showed an algorithm for the delete-only FED problem. In this section, we focus on the special case that both deletions and insertions are allowed only on σ. In Section 3.4.1, we describe notations and some useful tools. In Section 3.4.2, we give an algorithm for the case where only insertions are allowed. The main part of Section 3.4.2 focuses on the correctness of our algorithm. In Section 3.4.3, we show how the running time can be improved. In Section 3.4.5, we prove that our approach applies even when deletions are also allowed.

3.4.1 Definitions and notations

Let σ' be a polygonal curve with $|\sigma| + k$ vertices. We say σ' is a *k-insertions curve* of σ if the vertices of σ are a subset of the vertices of σ', and the vertices of σ appear in the same order in σ' as they appear in σ. Recall that $\langle p_1, ..., p_n\rangle$ is the polygonal curve connecting vertices $p_1, ..., p_n$. When a curve $\langle\sigma'_1, ..., \sigma'_a\rangle$ is inserted between two vertices σ_j and σ_{j+1}, $\sigma'[j, j+1]$ becomes the polygonal curve $\langle\sigma_j, \sigma'_1, ..., \sigma'_a, \sigma_{j+1}\rangle$ instead of a segment.

PROBLEM 2 (Insert-only FED problem). *Given two polygonal curves π and σ, a non-negative real number δ, and a non-negative integer k, determine if there exists an insertion curve π' of π and an insertion curve σ' of σ such that $d_{\mathcal{F}}(\pi', \sigma') \le \delta$ and $|\pi'| + |\sigma'| - |\pi| - |\sigma| \le k$.*

We use $\text{Ied}^1_\delta(\pi, \sigma)$ to denote the minimum number of inserted vertices in σ to obtain an insertion curve σ' such that $d_{\mathcal{F}}(\pi, \sigma') \le \delta$. Analogously, $\text{Ied}_\delta(\pi, \sigma)$ denotes the minimum number of vertices required to obtain insertion curves π' and σ' such that $d_{\mathcal{F}}(\pi', \sigma') \le \delta$. We use $s \circ s'$ to denote the *concatenation* of the curve s and the curve s'. If the last point s_b on s is the first point s'_a on s', then s and s' are simply merged and $s \circ s' = \langle s_a, ..., s_b = s'_a, ..., s'_b\rangle$. If s_b and s'_a are different, then the segment $s_b s'_a$ is added in between and $s \circ s' = \langle s_a, ..., s_b, s'_a, ..., s'_b\rangle$.

Consider the case when σ is a single segment st and π is a polygonal curve. The *minimum vertex curve* $\sigma' = mv(s, t, \pi)$ is a polygonal curve with minimum number of vertices such that $s \circ \sigma' \circ t$ first starts at point s, then stabs $\mathbf{b}(\pi_1)$, $\mathbf{b}(\pi_2)$, ..., $\mathbf{b}(\pi_n)$ in order, and finally

ends at point t. Note that $mv(s,t,\pi)$ contains neither s nor t, and it is empty if $d_{\mathcal{F}}(st,\pi) \leq \delta$. Under the right definition of ordered stabbing and containment condition [99, Definition 4], σ' is the polygonal curve with minimum number of vertices such that $d_{\mathcal{F}}(\pi, s \circ \sigma' \circ t) \leq \delta$. By combining previous results from Guibas, Hershberger, Mitchell, Snoeyink, Hsu, and Lee [99] and [79, Appendix A], one can compute, in quadratic time, the minimum vertex curves defined by any prefix curve of π in $\mathbb{R}^2$. In subsequent sections, the curves under discussion are in $\mathbb{R}^2$.

FACT 31. *In the plane, given a point s, a point t, and a polygonal curve π, for a fixed integer $i \in [1,n]$, it takes $\mathcal{M}(n) = \mathcal{O}(n^2 \log^2 n)$ time to compute every minimum vertex curve $mv(s,t,\pi[i,j])$ for every $j \in [i,n]$.*

In the following sections, we transform the insert-only FED problem into a weighted shortest path problem. We will build a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with directed and weighted edges using the critical points of the freespace diagram $\mathcal{D}_\delta(\pi,\sigma)$. For all i,j, the extreme points of $L^f_{i,j}$ and $B^f_{i,j}$ are critical points. Each critical point $a \in \mathcal{D}_\delta(\pi,\sigma)$ corresponds to a vertex $v(a)$ in $\mathcal{G}$. Once we have defined $\mathcal{G}$, we prove that the weight $\mathbf{w}(P)$ of the weighted shortest path P from $v(s)$ to $v(t)$ is the smallest number of edits required, where $s = (1,1)$ and $t = (n,m)$ are critical points.

To simplify the presentation, we impose the restriction that no vertices are allowed to be inserted before the first or after the last vertex of any curve (see Remark 7). This implies that $\|\pi_1 - \sigma_1\| \leq \delta$ and $\|\pi_n - \sigma_m\| \leq \delta$; otherwise, $\text{Ied}_\delta(\pi,\sigma) = \text{Ied}^1_\delta(\pi,\sigma) = \infty$ and our algorithms can immediately stop.

3.4.2 Insert-only

In this section, we analyse the case where only insertions are allowed, and only in σ. We first describe how to transform the insert-only FED problem into a weighted shortest path problem in a graph. Using the critical points in $\mathcal{D}_\delta(\pi,\sigma)$, we construct a directed graph, and an edge weight describes the number of inserted vertices required to "jump" through the non-free space from one critical point to another.

To traverse the freespace when no edits are required, we initialise $\mathcal{G}$ using the data structure by Gudmundsson and Wong [95] and assign these edges weight 0.

FACT 32 ([95, Lemma 19 and Theorem 22]). *The* freespace graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ *contains the following vertices and edges. For each critical point* a *in* $\mathcal{D}_\delta(\pi, \sigma)$*, there is a corresponding vertex* $v(a)$ *in* $\mathcal{V}$*. For every pair* (a, b) *of critical points, there is a* 0*-weighted path from* $v(a)$ *to* $v(b)$ *in* $\mathcal{G}$ *if and only if there is an* xy*-monotone path from* a *to* b *through* $\mathcal{F}_\delta(\pi, \sigma)$*. The freespace graph has* $\mathcal{O}(mn \log(m+n))$ *edges and can be constructed in* $\mathcal{O}(mn \log(m+n))$ *time.*

Consider a pair (σ_j, σ_{j+1}) of vertices of σ, an *exit point* $\pi[\alpha]$ where π exits $\mathbf{b}(\sigma_j)$, and an *entry point* $\pi[\beta]$ where π enters $\mathbf{b}(\sigma_{j+1})$. [79] proved the following. In order to compute an insertion curve of σ, it is sufficient to consider $P = mv(\sigma_j, \sigma_{j+1}, \pi[\alpha, \beta])$ for all $j \in [1, m)$ and all pairs of entry and exit points. Note that P can be empty, in which case we require no insertions between σ_j and σ_{j+1}.

FACT 33 ([79, Lemma 10 and Corollary 12]). $\mathrm{Ied}^1_\delta(\pi, \sigma) \leq k$ *if and only if the following is true. There exists a* k*-insertion curve* σ' *of* σ *such that* $d_{\mathcal{F}}(\pi, \sigma') \leq \delta$ *and* $\sigma'[j, j+1] = \sigma_j \circ mv(\sigma_j, \sigma_{j+1}, \pi[\alpha, \beta]) \circ \sigma_{j+1}$ *for every integer* $j \in [1, m)$*, where* $\pi[\alpha]$ *is an exit point of* $\mathbf{b}(\sigma_j)$ *and* $\pi[\beta]$ *is an entry point of* $\mathbf{b}(\sigma_{j+1})$.

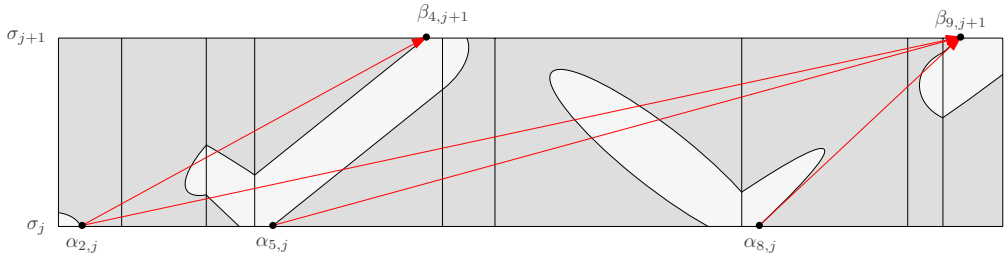

FIGURE 3.8: We construct every edge $e = (v(\mathbf{r}_{i,j}), v(\mathbf{l}_{i',j+1}))$, then assign $\mathbf{w}(e) = |mv(\sigma_j, \sigma_{j+1}, \pi[\alpha_{i,j}, \beta_{i',j+1}])|$.

Each entry or exit point corresponds to a unique critical point in the freespace diagram. Let $1 \leq i \leq n$ and $1 \leq j \leq m$ be integers. If $(i, j) \notin \mathcal{F}_\delta(\pi, \sigma)$, then let $\mathbf{r}_{i,j} = (j, \beta_{i,j})$ be the leftmost point of $B^f_{(i,i+1),j}$, and let $\mathbf{l}_{i,j} = (j, \alpha_{i,j})$ be the rightmost point of $B^f_{(i-1,i),j}$ if these critical points exist (see Figure 3.9). Using Fact 31, 32, and 33, we present an algorithm that

adds an edge $e = (v(\mathbf{r}_{i,j}), v(\mathbf{l}_{i',j+1}))$ with weight $|mv(\sigma_j, \sigma_{j+1}, \pi[\alpha_{i,j}, \beta_{i',j+1}])|$ for all values $i' \geq i$ and j (see Figure 3.8). However, this method is no longer sufficient when insertions are allowed on both curves. Therefore, we present a more general approach, which can be extended to use in later sections. We first observe the following.

OBSERVATION 34. *Let $\alpha = \alpha_{i,j}$ and $\beta = \beta_{i',j+1}$, where $i' \geq i$. Let $P = mv(\sigma_j, \sigma_{j+1}, \pi[\alpha, \beta])$ and $Q = mv(\sigma_j, \sigma_{j+1}, \pi[i, i'])$. We have $|P| = |Q|$, and $d_{\mathcal{F}}(Q, \pi[\alpha, \beta]) \leq \delta$.*

Using Fact 33 and Observation 34, we have the following.

LEMMA 35. *$\mathrm{Ied}^1_\delta(\pi, \sigma) \leq k$ if and only if there exists a k-insertions curve σ' of σ such that $d_{\mathcal{F}}(\pi, \sigma') \leq \delta$ and for all integers $j \in [1, m)$ and any integers $i \leq i'$, $\sigma'[j, j+1] = \sigma_j \circ mv(\sigma_j, \sigma_{j+1}, \pi[i, i']) \circ \sigma_{j+1}$.*

DEFINITION 36 (Augmented graph for σ-insertions). *See Figure 3.9. For each $(i, j) \notin \mathcal{F}_\delta(\pi, \sigma)$, an* augmented graph for σ-insertions *determined by $\mathcal{D}_\delta(\pi, \sigma)$ contains*

- *two* dummy vertices $\sigma^{\text{in}}_{i,j}$ *and* $\sigma^{\text{out}}_{i,j}$,
- *two* 0*-weighted edges* $(\sigma^{\text{in}}_{i,j}, v(\mathbf{r}_{i,j}))$ *and* $(v(\mathbf{l}_{i,j}), \sigma^{\text{out}}_{i,j})$, *and*
- *for all $i' \geq i$, an edge* $(\sigma^{\text{out}}_{i,j}, \sigma^{\text{in}}_{i',j+1})$ *with weight* $|mv(\sigma_j, \sigma_{j+1}, \pi[i, i'])|$.

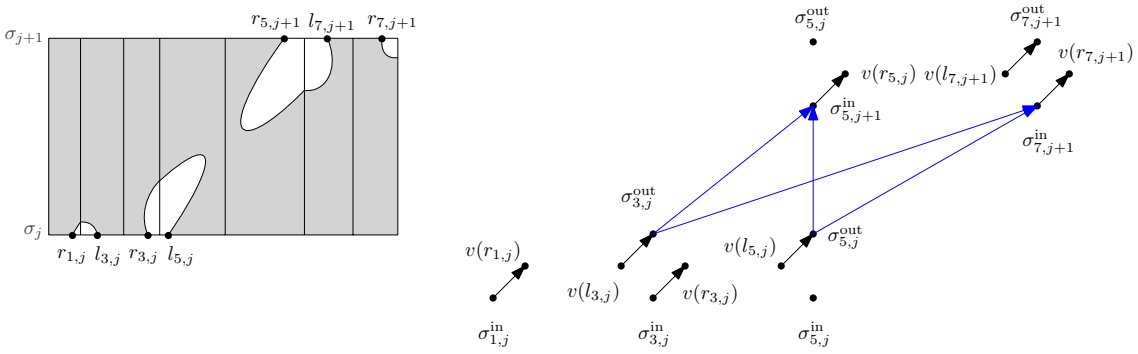

FIGURE 3.9: The augmented graph for σ-insertions (right), determined by the freespace diagram (left). Black edges have weight 0, and blue edges have positive weights. The weights of the blue edges are computed according to Definition 36.

With both the freespace graph and the augmented graph for insertions defined, we first describe Algorithm 5, which uses the union of the two graphs. Then, in Lemma 39, we prove

that a weighted shortest path in this rather dense graph corresponds to the minimum number of inserted vertices required. Following the correctness argument, we will then prove that a subset of edges is sufficient. We say the weight $\mathbf{w}(u)$ of a vertex u is the minimum weight of any path from vertex $v((1,1))$ to u.

Algorithm 5 Construct the union $\mathcal{G}$ of the freespace graph and the augmented graph for σ-insertions, both determined by $\mathcal{D}_\delta(\pi,\sigma)$. Report the weight of the weighted shortest path from $v(s)$ to $v(t)$ as $\mathrm{Ied}^1_\delta(\pi,\sigma)$, where $s=(1,1)$ and $t=(n,m)$ are critical points in $\mathcal{D}_\delta(\pi,\sigma)$.

To show correctness, we first make an observation on the matching that realises the Fréchet distance involving an insertion curve (see Figure 3.10).

LEMMA 37. *Let σ^* be an optimal insertion curve of σ that realises* $\mathrm{Ied}^1_\delta(\pi,\sigma)$. *For any adjacent pair (σ_s,σ_t) of original vertices in σ^* such that $\sigma^*[s,t]$ contains an inserted curve that matches $\pi[a,b]$, there exist two* well-defined *points $\pi[\alpha]$ and $\pi[\beta]$, where $\pi[\alpha]$ is the first exit point of $\mathbf{b}(\sigma_s)$ after $\pi[a]$, $\pi[\beta]$ is the last entry point of $\mathbf{b}(\sigma_t)$ before $\pi[b]$, and $\alpha \le \beta$. Furthermore, there exists a matching M that realises $d_{\mathcal{F}}(\pi,\sigma^*)$ and matches σ_s to $\pi[\alpha]$ and σ_t to $\pi[\beta]$.*

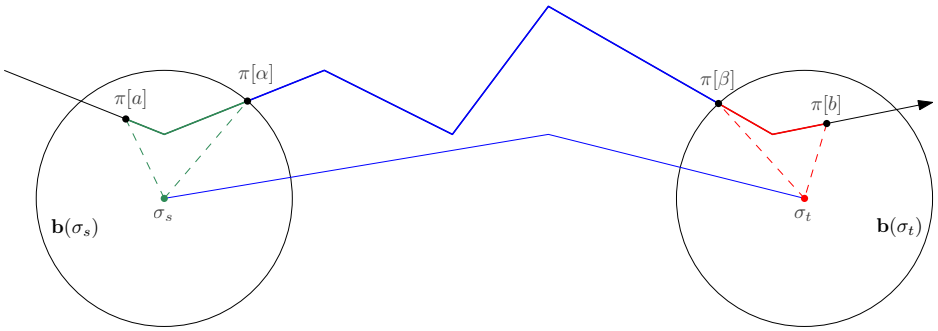

FIGURE 3.10: If there are inserted vertices between a pair (σ_s,σ_t) of vertices, we change the matching so that the subcurves of the same colour are matched.

PROOF. We will prove the lemma by contradiction. Two points $\pi[\alpha]$ and $\pi[\beta]$ are not well-defined when at least one of the following three statements is true: $\pi[a,b]$ does not exit $\mathbf{b}(\sigma_s)$, $\pi[a,b]$ does not enter $\mathbf{b}(\sigma_t)$, or $\alpha > \beta$ for every pair $(\pi[\alpha],\pi[\beta])$ of exit and entry points. If $\pi[a,b]$ does not exit $\mathbf{b}(\sigma_s)$, then $d_{\mathcal{F}}(\pi[a,b],\sigma_s) \le \delta$ and $d_{\mathcal{F}}(\pi[b],\sigma_t) \le \delta$. Hence,

no insertions between σ_s and σ_t are required, contradicting the assumption that σ^* is optimal and $\sigma^*[s,t]$ contains inserted vertices. For analogous reasons, $\pi[a,b]$ must enter $\mathbf{b}(\sigma_t)$.

If $\alpha > \beta$ for every pair $(\pi[\alpha], \pi[\beta])$ of exit and entry points, we have an ordered set of real numbers $\beta_1, ..., \beta_b, \alpha_1, ..., \alpha_a$ such that each $\pi[\alpha_i]$ is an exit point and each $\pi[\beta_i]$ is an entry point. Every $\pi[\beta_i]$ must lie in $\mathbf{b}(\sigma_s)$, and every $\pi[\alpha_i]$ must lie in $\mathbf{b}(\sigma_t)$. We have $d_{\mathcal{F}}(\pi[a,\beta_b], \sigma_s) \leq \delta$, $d_{\mathcal{F}}(\pi[\beta_b, \alpha_1], \sigma_s\sigma_t) \leq \delta$, and $d_{\mathcal{F}}(\pi[\alpha_1, \alpha_a], \sigma_t) \leq \delta$. Thus, no insertions are required. All three statements contradict either $d_{\mathcal{F}}(\pi[a,b], \sigma^*[s,t]) \leq \delta$ or the optimality of σ^*. Therefore, there must exist a pair $\pi[\alpha]$ and $\pi[\beta]$ of well-defined exit and entry points.

With well-defined entry and exit points, we adjust M such that it first matches $\pi[a,\alpha]$ to σ_s, then $\pi[\alpha,\beta]$ to $\sigma[s,t]$, and finally $\pi[\beta,b]$ to σ_t. The matching M is still valid because $d_{\mathcal{F}}(\pi[a,\alpha], \sigma_s) \leq \delta$, $d_{\mathcal{F}}(\pi[\alpha,\beta], \sigma^*[s,t]) \leq \delta$, and $d_{\mathcal{F}}(\pi[\beta,b], \sigma_t) \leq \delta$. □

Since each exit point corresponds to a critical point $\mathbf{l}_{i,j}$ and each entry point corresponds to a critical point $\mathbf{r}_{i,j}$, Lemma 37 implies the following properties of a freespace path P in $\mathcal{D}_\delta(\pi, \sigma)$.

COROLLARY 38. *Let σ^* be an insertion curve of σ that realises $\mathrm{Ied}^1_\delta(\pi,\sigma)$. Let $\mathbf{l}_{1,1} = (1,1)$ and let $\mathbf{r}_{n,m} = (n,m)$ be critical points in $\mathcal{D}_\delta(\pi,\sigma^*)$. There exists a freespace path in $\mathcal{D}_\delta(\pi,\sigma^*)$ that visits a maximal sequence $\{\mathbf{l}_{a_1,b_1}, \mathbf{r}_{a_2,b_2}, \mathbf{l}_{a_3,b_3}, ..., \mathbf{r}_{a_h,b_h}\}$ of critical points satisfying, for $i = 1, 2, ..., h-1$:*

(1) *a_i and b_i are integers,*
(2) *$a_1 = b_1 = 1$, $a_h = n$, $b_h = m$, $a_i \leq a_{i+1}$, and $b_i \leq b_{i+1}$,*
(3) *every $\sigma^*[b_i]$ is a vertex of the original curve,*
(4) *for each subpath that connects $\mathbf{l}_{a_i,b_i}$ to $\mathbf{r}_{a_{i+1},b_{i+1}}$, $\sigma^*[b_i, b_{i+1}]$ contains at least one inserted vertex, and*
(5) *for each subpath that connects $\mathbf{r}_{a_i,b_i}$ to $\mathbf{l}_{a_{i+1},b_{i+1}}$, $\sigma^*[b_i, b_{i+1}]$ contains no inserted vertices.*

We call a freespace path *well-structured* if it fulfils the properties in Corollary 38. We prove that if a well-structured freespace path exists, then a corresponding graph path must exist in the union of the freespace graph and the augmented insertion graph.

LEMMA 39. *Let $\mathcal{G}$ be the union of the freespace graph and the augmented graph for σ-insertions, both determined by $\mathcal{D}_\delta(\pi, \sigma)$. The weighted shortest path from $v((1,1))$ to $v((n,m))$ has weight k if and only if* $\mathrm{Ied}^1_\delta(\pi, \sigma) = k$.

PROOF. Let σ^* be an insertion curve of σ that realises $\mathrm{Ied}^1_\delta(\pi, \sigma)$. By Corollary 38, we know that there exists a well-structured path $\mathcal{P}$ visiting a maximal sequence $\{\mathbf{l}_{a_1,b_1}, \mathbf{r}_{a_2,b_2}, \mathbf{l}_{a_3,b_3}, ..., \mathbf{r}_{a_h,b_h}\}$ of critical points in $\mathcal{D}_\delta(\pi, \sigma^*)$ that satisfy the following.

If a freespace path $P' \subseteq \mathcal{F}_\delta(\pi, \sigma^*)$ connecting $\mathbf{r}_{a_i,b_i}$ to $\mathbf{l}_{a_{i+1},b_{i+1}}$ exists, and $\sigma^*[b_i, b_{i+1}]$ contains no inserted vertices, then the freespace diagrams $\mathcal{D}_\delta(\pi[a_i, a_{i+1}], \sigma^*[b_i, b_{i+1}])$ determined by the insertion curve σ^* and $\mathcal{D}_\delta(\pi[a_i, a_{i+1}], \sigma[b_i, b_{i+1}])$ determined by the original curve σ are exactly the same, which implies $P' \subseteq \mathcal{F}_\delta(\pi, \sigma)$. By Fact 32, there exists a graph path $P'_\mathcal{G}$ in the freespace graph that connects $v(\mathbf{r}_{a_i,b_i})$ to $v(\mathbf{l}_{a_{i+1},b_{i+1}})$ with $\mathbf{w}(P'_\mathcal{G}) = 0$.

If a freespace path $P \subseteq \mathcal{F}_\delta(\pi, \sigma^*)$ connecting $\mathbf{l}_{a_i,b_i}$ to $\mathbf{r}_{a_{i+1},b_{i+1}}$ exists, and $\sigma^*[b_i, b_{i+1}]$ contains k_i inserted vertices, then for any insertion curve $\sigma'[b_i, b_{i+1}]$ with $d_\mathcal{F}(\pi[a_i, a_{i+1}], \sigma'[b_i, b_{i+1}]) \leq \delta$, the minimum vertex curve $Q = mv(\sigma_{b_i}, \sigma_{b_{i+1}}, \pi[a_i, a_{i+1}])$ has the minimum number of vertices by definition. We have $|Q| = k_i$ as otherwise σ^* is not optimal. Using the construction in Definition 36, there is a path $P_\mathcal{G}$ with three edges connecting $v(\mathbf{l}_{a_i,b_i})$ to $\sigma^{\text{out}}_{a_i,b_i}$, then to $\sigma^{\text{in}}_{a_{i+1},b_{i+1}}$, and finally to $\mathbf{r}_{a_{i+1},b_{i+1}}$. The total weight of $P_\mathcal{G}$ is the weight of the edge $(\sigma^{\text{out}}_{a_i,b_i}, \sigma^{\text{in}}_{a_{i+1},b_{i+1}})$, which is exactly $|Q|$.

Summing the weights of all the subpaths in $\mathcal{P}$ gives a weighted shortest path of weight exactly k whenever such an optimal insertion curve σ^* exists. Conversely, by computing the weighted shortest path $\mathcal{P}_\mathcal{G}$, we can reconstruct σ^*. The correctness proof is now complete. □

In Lemma 39, we have shown the correctness of Algorithm 5. The augmented graph for σ-insertions contains at most $\widetilde{\mathcal{O}}(mn^2)$ edges in total. Computing the weight of each edge takes $\mathcal{M}(n) = \widetilde{\mathcal{O}}(n^2)$ time by Fact 31.

3.4.3 Improving the time complexity

To improve the running time of the algorithm above, we make two geometric observations. When combined, these observations imply that (1) it is sufficient to add and compute the weight of a smaller subset of edges, and (2) the weights of several edges can be computed in batches.

LEMMA 40. *For any two points s and t, and vertex indices $a < b$ and i, we have $|mv(s,t,\pi[i,a])| \leq |mv(s,t,\pi[i,b])|$. Furthermore,* $\text{Ied}^1_\delta(\pi[1,\lambda],\sigma) \leq \text{Ied}^1_\delta(\pi[1,\mu],\sigma)$ *for real values $\lambda \leq \mu$, provided that $\pi[\lambda], \pi[\mu] \in \mathbf{b}(\sigma_m)$, where σ_m is the last vertex of σ.*

PROOF. We take the minimum vertex curve $\sigma' = mv(s,t,\pi[i,b])$ and remove its trailing subcurve, ensuring that $s \circ \sigma'[1,\gamma] \circ t$ is within Fréchet distance δ from $\pi[i,a]$. Since $\sigma'[1,\gamma]$ is a subcurve of σ', we have $|\sigma'[1,\gamma]| \leq |\sigma'|$, and by the definition of the minimum vertex curve, $|mv(s,t,\pi[i,a])| \leq |\sigma'[1,\gamma]|$.

Analogous argument applies to the second statement. Take any insertion curve σ'' that realises $\text{Ied}^1_\delta(\pi[1,\mu],\sigma)$, and remove its trailing subcurve until what remains, $\sigma''[1,\gamma]$, is still within Fréchet distance δ from $\pi[1,\lambda]$. If $\sigma''[\gamma,m]$ is a segment, then $d_{\mathcal{F}}(\pi[1,\mu],\sigma'') \leq \delta$ by linear interpolation. If $\sigma''[\gamma,m]$ contains at least one vertex besides σ_m, then $d_{\mathcal{F}}(\sigma''[1,\gamma],\pi[1,\lambda]) \leq \delta$ and $|\sigma''[1,\gamma]| \leq |\sigma''|$. The curve that realises $\text{Ied}^1_\delta(\pi[1,\lambda],\sigma)$ must have at most as many vertices as $\sigma''[1,\gamma]$. □

Using the above lemma, we can establish monotonicity for the weights of the critical points defined by the same vertex σ_j. Using this observation, we can determine the weight of a fixed critical point $\mathbf{r}_{i,j+1}$ by considering at most $k+2$ critical points determined by σ_j. Let $\mathcal{A}_j$ (resp. $\mathcal{B}_j$) be the set of integers i such that the critical point $\mathbf{l}_{i,j}$ (resp. $\mathbf{r}_{i,j}$) exists in $\mathcal{D}_\delta(\pi,\sigma)$. We say that $\mathbf{r}_{i,j}$ and $v(\mathbf{r}_{i,j})$ (resp. $\mathbf{l}_{i,j}$ and $v(\mathbf{l}_{i,j})$) are determined by $\mathcal{A}_j$ (resp. $\mathcal{B}_j$) if $i \in \mathcal{A}_j$ (resp. $i \in \mathcal{B}_j$).

LEMMA 41. *Let $\mathcal{G}$ be the union of the freespace graph and the augmented graph for σ-insertions, both determined by $\mathcal{D}_\delta(\pi,\sigma)$. For a fixed integer i', let $\mathcal{A}_j^{\leq i'} \subseteq \mathcal{A}_j$ denote the subset with the following two properties: (i) for all $i \in \mathcal{A}_j^{\leq i'}$, $i \leq i'$, and (ii) every $i \in \mathcal{A}_j^{\leq i'}$ is*

either the largest value in $\mathcal{A}_j$ such that $i \leq i'$, or i is the largest value for which $\mathbf{w}(v(\mathbf{l}_{i,j})) = c$, where $c \in [0, k]$ is arbitrary. We have

$$\mathbf{w}(v(\mathbf{r}_{i',j+1})) = \min\{i \in \mathcal{A}_j^{\leq i'} \mid \mathbf{w}(v(\mathbf{l}_{i,j})) + |mv(\sigma_j, \sigma_{j+1}, \pi[i, i'])|\}$$

unless every minimum-weight path arriving at $v(\mathbf{r}_{i',j+1})$ ends with edges from the freespace graph.

PROOF. The correctness follows from Lemma 40. Consider the curve π', which is a copy of π but in the reverse direction, and observe that $|mv(\sigma_{j+1}, \sigma_j, \pi'[i, i'])| = |mv(\sigma_j, \sigma_{j+1}, \pi[i', i])|$. Combining this with Lemma 40, we obtain $|mv(\sigma_j, \sigma_{j+1}, \pi[a, i])| \leq |mv(\sigma_j, \sigma_{j+1}, \pi[b, i])|$ for integers $a < b$. Therefore, when $\mathbf{w}(v(\mathbf{l}_{a,j})) = \mathbf{w}(v(\mathbf{l}_{b,j}))$, to determine $\mathbf{w}(v(\mathbf{r}_{i',j+1}))$ it is sufficient to compute $mv(\sigma_j, \sigma_{j+1}, \pi[b, i])$, since its cardinality plus $\mathbf{w}(v(\mathbf{l}_{b,j}))$ must be smaller. □

We are interested in the minimum weight of any path that reaches every vertex determined by $\mathcal{B}_{j+1}$. In Algorithm 5, for a fixed j, we computed every edge $(\sigma_{i,j}^{\text{out}}, \sigma_{i',j+1}^{\text{in}})$ for all $i \leq i'$ in $\mathcal{O}(n^2\mathcal{M}(n))$ time. Lemma 41 allows us to compute a smaller subset of these edges, and we show that both the edges and their weights can be computed much faster.

LEMMA 42. *Given $\mathbf{w}(v(\mathbf{l}_{i,j}))$ for all $i \in \mathcal{A}_j$, it takes $\mathcal{O}(k\mathcal{M}(n))$ time to compute all edges $(\sigma_{i,j}^{\text{out}}, \sigma_{i',j+1}^{\text{in}})$ that realise $\mathbf{w}(v(\mathbf{r}_{i',j+1}))$ for every $i' \in \mathcal{B}_{j+1}$.*

PROOF. For a fixed i', we consider the subset $\mathcal{A}_j^{\leq i'} \subseteq \mathcal{A}_j$ such that for every $i \in \mathcal{A}_j^{\leq i'}$, the critical point $\mathbf{l}_{i,j}$ exists. By Lemma 41, for every $i' \in \mathcal{B}_{j+1}$, it is sufficient to consider two types of values $i \in \mathcal{A}_j$ to upper bound $\mathbf{w}(v(r))$.

The first type is when i is the largest value at most i', and we claim that it takes $\mathcal{M}(n)$ time to compute $mv(\sigma_j, \sigma_{j+1}, \pi[i, i'])$ for every valid pair i and i' in total. Indeed, consider partitioning $\mathcal{B}_{j+1}$ into contiguous subsets $B_1, ..., B_h$ such that, for $g = 1, ..., h$, all values in B_g share the same largest value $a_g \in \mathcal{A}_j$. Recall that Fact 31 uses the algorithm in [99], which computes a polygonal stabber visiting n disks in order in $c_1 n^2 \log^2 n = \mathcal{O}(n^2 \log^2 n)$ time, and the reduction in [79, Appendix A], which forces the stabber to start and end at prescribed

points by adding a constant c_2 number of disks. Therefore, computing $mv(\sigma_j, \sigma_{j+1}, \pi[a_g, b_g])$ takes at most $c_1 \cdot (c_2 + b_g - a_g + 1)^2 \log^2 n$ operations. Since $\sum_g (b_g - a_g) \leq 2n$, the total time is

$$\begin{aligned}
&\sum_{1 \leq g \leq h} c_1 \cdot \big((c_2 + b_g - a_g + 1)^2 \log^2 n\big) \\
&\leq c_1 \log^2 n \sum_{1 \leq g \leq h} (c_2 + b_g - a_g + 1)^2 \\
&= c_1 \log^2 n \sum_{1 \leq g \leq h} \big((b_g - a_g)^2 + 2 \cdot (b_g - a_g) \cdot (c_2 + 1) + (c_2 + 1)^2\big) \\
&= \mathcal{O}(n^2 \log^2 n).
\end{aligned}$$

The second type occurs when i is the largest index such that $\mathbf{w}(v(\mathbf{l}_{i,j})) = c$ for some $c \in [0, k]$. For each such index i and each valid index $i' \geq i$, we compute $mv(\sigma_j, \sigma_{j+1}, \pi[i, i'])$ using the minimum-link path algorithm. There are k such indices i and at most $\mathcal{O}(n)$ valid choices of i' for each. Thus, a direct application of Algorithm 5 requires $\mathcal{O}(kn \cdot \mathcal{M}(n))$ time. A more careful inspection of the minimum-link path algorithm of [99], together with a refined analysis, removes an additional factor of n.

Their algorithm computes a minimum-link path by iteratively maintaining the *wedge* $\mathcal{W}(i, i')$, a convex region of complexity at most $\mathcal{O}(n)$. This wedge contains all possible locations of the last vertex of a minimum-link path that starts at σ_j, stabs the disks $\mathbf{b}(\pi_i), \ldots, \mathbf{b}(\pi_{i'})$, and is not required to terminate at σ_{j+1} [99, Lemma 13]. Instead of recomputing the minimum-link path from scratch for each pair (i, i') with $i < i'$, we maintain $\mathcal{W}(i, i')$ incrementally. Once $\mathcal{W}(i, i')$ has been constructed, we determine whether extending the path to σ_{j+1} requires zero or one additional link. Using the technique in [79, Appendix A], this step can be performed in $\mathcal{O}(n \log^2 n)$ time by letting the minimum-link path continue to stab a constant number of disks.

We analyse the total running time in two parts. First, for each of the k indices i, we iteratively construct $\mathcal{W}(i, i')$ for all valid indices i', which requires $\mathcal{O}(k \cdot \mathcal{M}(n))$ time in total. Second, for each valid index i', we compute $mv(\sigma_j, \sigma_{j+1}, \pi[i, i'])$ from $\mathcal{W}(i, i')$ in $\mathcal{O}(n \log^2 n)$ time.

Since there are at most $\mathcal{O}(n)$ valid indices i' for each of the k choices of i, this contributes $\mathcal{O}(kn \log^2 n)$ time overall. Consequently, the total running time is $\mathcal{O}(k \cdot \mathcal{M}(n))$. □

Instead of constructing the entire graph before computing a shortest path, the algorithm in Lemma 43 uses Lemma 42 to compute vertex weights in a row-by-row fashion.

LEMMA 43. *Given two curves π and σ with $|\pi| = n$ and $|\sigma| = m$ in $\mathbb{R}^2$, it takes $\mathcal{O}(km\mathcal{M}(n))$ time to determine whether* $\text{Ied}^1_\delta(\sigma, \pi) \leq k$ *and $\mathcal{O}(nm\mathcal{M}(n))$ time to compute* $\text{Ied}^1_\delta(\sigma, \pi)$.

PROOF. We use Algorithm 5 with two modifications. The first modification is that instead of constructing the entire graph $\mathcal{G}$ before using Dijkstra's, we first build the freespace graph and then iterate j from 1 to m to compute every $\mathbf{w}(\sigma^{\text{out}}_{i,j})$. The second modification is that we do not compute every $mv(\sigma_j, \sigma_{j+1}, \pi[i, i'])$ separately. Instead, we run just enough passes of the minimum vertex path algorithm to obtain the weights using Lemma 42. At the end, if $\mathbf{w}((n, m)) \leq k$, we report $\text{Ied}^1_\delta(\pi, \sigma) \leq k$. For the optimisation version, we report $\mathbf{w}((n, m))$ as $\text{Ied}^1_\delta(\pi, \sigma)$. The correctness is implied by Fact 32, Fact 33, Lemma 39, and Lemma 41. See Remark 7 for a discussion of inserting leading and trailing vertices.

We analyse the time complexity. Initially, $\mathcal{A}_1$ contains a single value 1 since $\mathbf{w}((1, 1)) = 0$. Given the weights of vertices determined by $\mathcal{A}_j$, we compute the weights of vertices determined by $\mathcal{A}_{j+1}$. Lemma 42 lets us compute the necessary edges and their weights in the augmented graph for σ-insertions in $\mathcal{O}(k\mathcal{M}(n))$ time. Each vertex determined by σ_{j+1} requires inspecting the weights of $\mathcal{O}(n)$ critical points in the freespace graph. The time complexity is dominated by computing the minimum vertex paths. The decision version takes $\mathcal{O}(km\mathcal{M}(n))$ time, and the optimisation version takes $\mathcal{O}(nm\mathcal{M}(n))$ time. □

3.4.4 When insertions are allowed before the leading and after the trailing vertices

Remark 7 In the above algorithm, we assumed for simplicity of the description that the Fréchet distance between σ_1 and π (the whole curve) is at most δ as otherwise, $\text{Ied}^1_\delta(\pi, \sigma) = \infty$. If insertions are required before σ_1, then σ_1 is not matched to π_1. For every entry point $i \in \mathcal{B}_1$,

we can therefore simply compute $mv(\mathbb{R}^2, \sigma_1, \pi[1, i])$ and assign its size as an additive weight $\omega(\sigma^{\text{out}}_{i,1})$. Then, we can use the algorithm in Lemma 43 as is except that a starting vertex $\sigma^{\text{out}}_{i,1}$ has an additive cost $\omega(\sigma^{\text{out}}_{i,1})$. Analogously, for every $i \in \mathcal{A}_m$, we can impose an additive weight $\omega(i) = |mv(\sigma_m, \mathbb{R}^2, \pi[\alpha, n])|$. The algorithm can stop at every $v(\mathbf{r}_{i,m})$, and report the minimum $\omega(\alpha) + \mathbf{w}(\alpha)$ as $\text{Ied}^1_\delta(\pi, \sigma)$. Additive weights can be computed in $\mathcal{O}(\mathcal{M}(n))$ time, and embedded in the graph in $\mathcal{O}(n)$ time.

THEOREM 8. *When insertions are allowed on only one of the two curves, the insert-only Fréchet edit distance in $\mathbb{R}^2$ can be solved in $\widetilde{\mathcal{O}}(kn^3)$ time.*

3.4.5 With both insertions and deletions

In this section, we allow both insertions and deletions when edits are allowed only on σ. For integers $k_1, k_2 > 0$, we say π'' is an *edit curve* of π if π'' is a k_1-insertions curve of π', where π' is a k_2-deletions curve of π. We have performed $k = k_1 + k_2$ edits to π, and π'' is a k-edits curve. We solve the following problem.

PROBLEM 3 (FED problem). *Given two polygonal curves π and σ, a non-negative real number δ, and a non-negative integer k, determine if there exists an edit curve π' of π and an edit curve σ' of σ with at most k edits in total such that $d_{\mathcal{F}}(\pi', \sigma') \leq \delta$.*

Previously in Section 3.4.2, we described an augmented graph for insertions, explained how we combined it with the freespace graph, and how each edge weight describes the minimum number of inserted vertices required to "jump" from one critical point to another.

Since an edit curve is also an insertion curve, the results in Section 3.4.2 are still valid. However, we require additional edges to take into consideration when consecutive vertices between (say) σ_j and $\sigma_{j'}$ are deleted. The weights of the new edges include the cost of deleting the $j' - j - 1$ consecutive vertices $\sigma_{j+1}, ..., \sigma_{j'-1}$.

DEFINITION 44 (augmented graph for σ-edits). *An* augmented graph for σ-edits *determined by $\mathcal{D}_\delta(\pi, \sigma)$ contains the augmented graph for σ-insertions. Additionally, for each $(i, j) \notin$*

$\mathcal{F}_\delta(\pi,\sigma)$ and for all $i' > i$, $j' > j$ such that $(i',j') \notin \mathcal{F}_\delta(\pi,\sigma)$ and $j' - j - 1 \leq k$, it includes an edge $(\sigma^{\text{out}}_{i,j}, \sigma^{\text{in}}_{i',j'})$ with weight $|mv(\sigma_j, \sigma_{j'}, \pi[i,i'])| + (j' - j - 1)$.

We also need an augmented graph for deletions, because the augmented graph for insertions alone does not handle the case when a portion of σ must be simplified to match a subsegment of π. Recall that we solved this problem in Algorithm 1, and we can transform the sets of reachable points into a graph. Each reachable point p is a vertex $v(p)$. There is an edge $(v(p), v(q))$ if the deletion cost of q is computed from p, and the edge weight is either 0 or 1 depending on whether q lies in the freespace. There are also edges between pairs of vertices defined by adjacent reachable points.

LEMMA 45. *Given two curves π and σ with $|\pi| = n$ and $|\sigma| = m$, and an integer $k > 0$, there exists a union $\mathcal{G}$ of the freespace graph and the augmented graph for σ-deletions, both determined by $\mathcal{D}_\delta(\pi,\sigma)$, with the following properties. Let $\mathbf{r}_{i,j} = (\beta_{i,j}, i)$, $\mathbf{l}_{i,j} = (\alpha_{i,j}, i)$, $\mathbf{r}_{1,1} = (1,1)$ and $\mathbf{l}_{n,m} = (n,m)$ be critical points in $\mathcal{D}_\delta(\pi,\sigma)$.*

(1) *For any integers $i \leq i'$ and $j \leq j'$, there exists a graph path P in $\mathcal{G}$ from $v(\mathbf{r}_{i,j})$ to $v(\mathbf{l}_{i',j'})$ with weight $\mathbf{w}(P) = k$ if and only if $\text{Ded}^1_\delta(\sigma[j,j'], \pi[\beta_{i,j}, \alpha_{i',j'}]) \leq k$.*
(2) *The augmented graph for σ-deletions has at most $\mathcal{O}(kmn)$ vertices and edges.*

Using both augmented graphs for deletions and insertions, we have an algorithm. Let $\text{Ed}^1_\delta(\pi,\sigma)$ denote the minimum number of edits required on σ to generate an edit curve σ' with $d_{\mathcal{F}}(\pi,\sigma') \leq \delta$.

Algorithm 6 Compute the union $\mathcal{G}$ of the freespace graph, the augmented graph for σ-edits, and the augmented graph for σ-deletions, all determined by $\mathcal{D}_\delta(\pi,\sigma)$. Iterate $j = 1, ..., m$, and compute the weights of every vertex $v(\mathbf{l}_{i,j})$ for all $i \in [1,n]$. Report $\mathbf{w}(t)$ as $\text{Ed}^1_\delta(\pi,\sigma)$, where $t = (n,m)$ is a critical point.

Since an edit curve is an insertion curve of a deletion curve, Lemma 39 implies the following corollary.

COROLLARY 46. *Let σ' be a deletion curve of σ. Let $\mathcal{G}$ be the union of the freespace graph and the augmented graph for σ'-insertions, both determined by $\mathcal{D}_\delta(\pi, \sigma')$. There is a weighted path of weight at most k from $v((1,1))$ to $v((n,m))$ in $\mathcal{G}$ if and only if* $\text{Ied}^1_\delta(\pi, \sigma') \leq k$.

LEMMA 47. *Given two curves π and σ in $\mathbb{R}^2$ with $|\pi| = n$ and $|\sigma| = m$, and an integer k, it takes $\mathcal{O}(k^2 m \mathcal{M}(n))$ time to determine if* $\text{Ed}^1_\delta(\pi, \sigma) \leq k$, *and $\mathcal{O}(mn^2\mathcal{M}(n))$ time to compute* $\text{Ed}^1_\delta(\pi, \sigma)$.

PROOF. We first prove the correctness of Algorithm 6, and we claim that $\text{Ed}^1_\delta(\pi, \sigma)$ equals the weight of the weighted shortest path from $v((1,1))$ to $v((n,m))$ in $\mathcal{G}$.

Let σ^* be the edit curve that realises $\text{Ed}^1_\delta(\pi, \sigma)$. If σ^* contains no inserted vertices, then this is the delete-only Fréchet edit distance problem. Lemma 45 applies and there exists a graph path in the union of the freespace graph, and the augmented graph for σ-deletions. Otherwise, σ^* contains at least one inserted vertex. Using Corollary 38, in $\mathcal{D}_\delta(\pi, \sigma')$ there exists a freespace path $\mathcal{P}$ that visits a sequence $\mathbf{l}_{a_1,b_1}, \mathbf{r}_{a_2,b_2}, \mathbf{l}_{a_3,b_3}, ..., \mathbf{r}_{a_h,b_h}$ of critical points, where $\mathbf{r}_{1,1} = (1,1)$ and $\mathbf{l}_{n,m} = (n,m)$.

For a subpath $P \subseteq \mathcal{P}$ connecting $\mathbf{l}_{a_i,b_i}$ to $\mathbf{r}_{a_{i+1},b_{i+1}}$, $\sigma^*[b_i, b_{i+1}]$ contains inserted vertices. We have a corresponding graph path $P_\mathcal{G} \subseteq \mathcal{G}$ for P by construction. The graph path $P_\mathcal{G}$ has three edges: from $\mathbf{l}_{a_i,b_i}$ to $\sigma^{\text{out}}_{a_i,b_i}$, then from $\sigma^{\text{out}}_{a_i,b_i}$ to $\sigma^{\text{in}}_{a_{i+1},b_{i+1}}$, and finally from $\sigma^{\text{in}}_{a_{i+1},b_{i+1}}$ to $\mathbf{r}_{a_{i+1},b_{i+1}}$. The subpath $P_\mathcal{G}$ has weight $|mv(\sigma_{b_i}, \sigma_{b_{i+1}}, \pi[a_i, a_{i+1}])| + b_{i+1} - b_i - 1$. Consecutive vertices $\sigma_{b_i+1}, \ldots, \sigma_{b_{i+1}-1}$ are deleted, and this costs exactly $b_{i+1} - b_i - 1$ edits. The minimum vertex curve $mv(\sigma_{b_i}, \sigma_{b_{i+1}}, \pi[a_i, a_{i+1}])$ uses the minimum number of inserted vertices to match $\sigma^*[b_i, b_{i+1}]$ to $\pi[a_i, a_{i+1}]$. Such an edit uses the fewest number of edits.

For a subpath $P' \subseteq \mathcal{P}$ connecting $\mathbf{r}_{a_i,b_i}$ to $\mathbf{l}_{a_{i+1},b_{i+1}}$, $\sigma^*[b_i, b_{i+1}]$ contains no inserted vertices. We have a corresponding graph path $P'_\mathcal{G}$. By Lemma 45, $P'_\mathcal{G}$ exists with $\mathbf{w}(P'_\mathcal{G}) = k$ if and only if $\text{Ded}^1_\delta(\sigma[j, j'], \pi[\beta_{i,j}, \alpha_{i',j'}]) \leq k$. Considering these two types of subpaths, we have that if $\text{Ed}^1_\delta(\pi, \sigma) = k$, then there exists a weighted path $\mathcal{P}_\mathcal{G} \in \mathcal{G}$ with weight k. Conversely, if such a path $\mathcal{P}_\mathcal{G}$ exists, then we can construct σ^* and finally conclude $\text{Ed}^1_\delta(\pi, \sigma) = k$.

To prove the time complexity, first note that Lemmas 40, 41, and 42 in Section 3.4.3 can be extended to the case where consecutive vertices between σ_j and $\sigma_{j'}$ are deleted. This holds because an edit curve is also an insertion curve. The only difference is that to compute the weights of the vertices determined by $\mathcal{A}_{j'}$, the algorithm must remember the weights of the vertices determined by $\mathcal{A}_j$ for all $j \in [j'-k-1, j')$. For a fixed j and for all values of i, it takes $\mathcal{O}(k^2\mathcal{M}(n))$ time to compute all edges $(\sigma^{\text{out}}_{i,j}, \sigma^{\text{in}}_{i,j+1})$, $(\sigma^{\text{out}}_{i,j}, \sigma^{\text{in}}_{i,j+2})$, ..., $(\sigma^{\text{out}}_{i,j}, \sigma^{\text{in}}_{i,j+k+1})$. These edges in the augmented graph for σ-edits take $\mathcal{O}(k^2 m\mathcal{M}(n))$ time to compute, and this dominates the size of $\mathcal{G}$. Hence, it takes $\mathcal{O}(k^2 m\mathcal{M}(n))$ time for the decision version and $\mathcal{O}(mn^2\mathcal{M}(n))$ time for the optimisation version. □

THEOREM 9. *The Fréchet edit distance problem in $\mathbb{R}^2$ can be solved in $\widetilde{\mathcal{O}}(k^2n^3)$ time when edits are allowed on only one of the two curves.*

3.5 Insertions on both curves

In this section, we discuss the case where insertions are allowed on both curves. We use $\text{Ied}_\delta(\pi, \sigma)$ to denote the minimum number of insertions required to obtain insertion curves π' of π and σ' of σ such that $d_{\mathcal{F}}(\pi', \sigma') \leq \delta$.

When insertions are allowed on both curves, we first observe that $\text{Ied}_\delta(\pi, \sigma) \leq m + n$, since one may insert the entire curve σ before π and then insert the entire curve π after σ. This upper bound still holds under the restriction that no insertions are allowed before π_1 and σ_1 or after π_n and σ_m, assuming $d_{\mathcal{F}}(\pi_1, \sigma_1) \leq \delta$ and $d_{\mathcal{F}}(\pi_n, \sigma_m) \leq \delta$. In this case, we obtain $\text{Ied}_\delta(\pi, \sigma) \leq m + n$ by inserting the curve $\sigma[1, m-1]$ between π_1 and π_2 to construct the insertion curve π', and inserting the curve $\pi[2, n]$ between σ_{m-1} and σ_m to construct the insertion curve σ', see Figure 3.11. The Fréchet distance $d_{\mathcal{F}}(\pi', \sigma')$ is at most δ, and at most $m + n$ vertices are inserted.

Observe that the last segment of the inserted subcurve $\pi'[1, 2]$ must be matched to the first segment of the inserted subcurve $\sigma'[m-1, m]$. For both inserted subcurves, the remaining portion (excluding their first or last segment) can be matched to a subcurve of the original

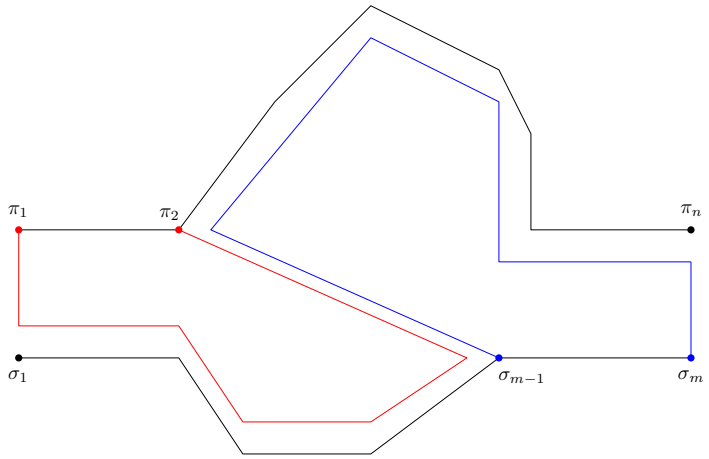

FIGURE 3.11: As long as $\|\pi_1 - \sigma_1\|, \|\pi_n - \sigma_m\| \leq \delta$, we have $\mathrm{Ied}_\delta(\pi, \sigma) \leq n + m$. The red curve is inserted between π_1 and π_2, and the blue curve is inserted between σ_{m-1} and σ_m. The depicted insertion curves contain more vertices than necessary.

curve. This observation is simple yet powerful, because it allows us to construct minimum vertex curves using only information from the original curves.

Before using this observation, we define an augmented graph for π-insertions. Consider the freespace diagram $\mathcal{D}_\delta(\pi, \sigma)$. For each $(i, j) \notin \mathcal{F}_\delta(\pi, \sigma)$, let $\mathbf{t}_{i,j}$ be the bottommost point of $L^f_{i,(j,j+1)}$, and let $b_{i,j}$ be the topmost point of $L^f_{i,(j-1,j)}$ (see Figure 3.12). We will use a similar approach as in Section 3.4.2: we build the freespace graph, combine it with two augmented graphs, and then perform a shortest-path query in the resulting graph.

DEFINITION 48 (Augmented graph for π-insertions). *For each $(i, j) \notin \mathcal{F}_\delta(\pi, \sigma)$, an* augmented graph for π-insertions *determined by $\mathcal{D}_\delta(\pi, \sigma)$ contains*

- *two* dummy vertices $\pi^{\text{in}}_{i,j}$ *and* $\pi^{\text{out}}_{i,j}$,
- *two* 0*-weight edges* $(\pi^{\text{in}}_{i,j}, v(\mathbf{t}_{i,j}))$ *and* $(v(b_{i,j}), \pi^{\text{out}}_{i,j})$, *and*
- *for all $j' \geq j$, an edge* $(\pi^{\text{out}}_{i,j}, \pi^{\text{in}}_{i+1,j'})$ *with weight* $|mv(\pi_i, \pi_{i+1}, \sigma[j, j'])|$.

Algorithm 7 Construct the union $\mathcal{G}$ of the freespace graph (Fact 32), the augmented graphs for both π-insertions and σ-insertions (Definitions 36 and 48), all determined by $\mathcal{D}_\delta(\pi, \sigma)$; see Figure 3.12. For each $(i, j) \notin \mathcal{F}_\delta(\pi, \sigma)$, add two edges $(\sigma^{\text{in}}_{i,j}, \pi^{\text{out}}_{i,j})$ and $(\pi^{\text{in}}_{i,j}, \sigma^{\text{out}}_{i,j})$ with

weight 0, called *alternating edges*. Report $\mathbf{w}(v((n,m)))$, the weight of a shortest weighted path from $(1,1)$ to (n,m), as $\mathrm{Ied}_\delta(\pi,\sigma)$.

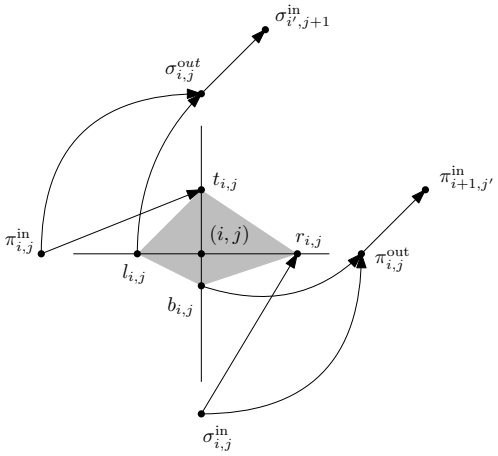

FIGURE 3.12: The dummy vertices for a given (i,j) in the non-free space, together with their connections to each other and to the vertices constructed from the critical points in the freespace diagram, are shown.

We now make several observations to prove that it is sufficient to consider these edges. We first observe that if an inserted subcurve of π^* is fully matched to an inserted subcurve of σ^*, then we can always remove the inserted vertices from (say) σ^* and add them to π^*.

OBSERVATION 49. *Let π^* and σ^* be a pair of insertion curves that realises $\mathrm{Ied}_\delta(\pi,\sigma)$. Let $Q = \sigma^*[j,j+1]$ be an inserted subcurve that matches to $P = \pi^*[\alpha,\beta]$ with at least one inserted vertex, where $\pi^*[\alpha]$ and $\pi^*[\beta]$ are not necessarily vertices. Then there exists a subcurve $P' = \pi'[\alpha,\beta]$ such that $d_{\mathcal{F}}(P', \sigma_j\sigma_{j+1}) \le \delta$ and $|P'| \le |P| + |Q|$.*

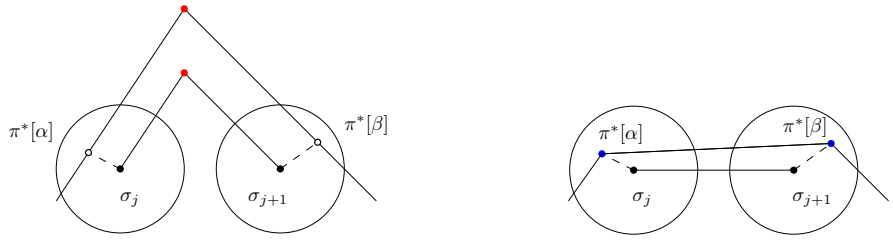

FIGURE 3.13: The red vertices must be inserted vertices, and they can be deleted to pay for adding the two vertices $\pi^*[\alpha]$ and $\pi^*[\beta]$.

PROOF. See Figure 3.13 for an illustration. Since P contains at least one inserted vertex, we can delete all inserted vertices in P and Q, and add the two vertices $\pi^*[\alpha]$ and $\pi^*[\beta]$ (see

Figure 3.14). We set $P' = \pi^*[\alpha]\pi^*[\beta]$, and $|P'| \leq |P| + |Q|$. Moreover, $d_{\mathcal{F}}(P', \sigma_j\sigma_{j+1}) \leq \delta$ because $\pi^*[\alpha] \in \mathbf{b}(\sigma_j)$ and $\pi^*[\beta] \in \mathbf{b}(\sigma_{j+1})$. □

We next define a set $\mathcal{C}$ of *canonical insertion curves*. An insertion curve σ' of σ is said to be *put together* using curves in $\mathcal{C}$ if, for all $j = 1, \ldots, m-1$, the subcurve $\sigma'[j, j+1] \in \mathcal{C}$. The set $\mathcal{C}$ is defined as follows:

$$\mathcal{C} = \left\{ \begin{array}{l|l} mv(\sigma_j, \sigma_{j+1}, \pi[a,b]), & \\ mv(\pi_i, \pi_{i+1}, \sigma[c,d]), & a,b,c,d,i,j \in \mathbb{Z}^+, \\ \sigma_j\sigma_{j+1}, & c \leq d \leq m, a \leq b \leq n \\ \pi_i\pi_{i+1} & i \leq n, j \leq m \end{array} \right\}.$$

In Lemma 50, we will prove that a pair of optimal insertion curves can be put together using the curves in $\mathcal{C}$.

LEMMA 50. *Let π^* and σ^* be insertion curves obtained from π and σ using subcurves in $\mathcal{C}$ such that (i) the number of inserted vertices is minimised, and let this minimum be k, and (ii) the Fréchet distance between them is at most δ. Then* $\text{Ied}_\delta(\pi, \sigma) = k$.

PROOF. If a pair of insertion curves π^* and σ^* can be put together using subcurves in $\mathcal{C}$ such that $d_{\mathcal{F}}(\pi^*, \sigma^*) \leq \delta$ and $|\pi^*| + |\sigma^*| - n - m = k$, then clearly $\text{Ied}_\delta(\pi, \sigma) \leq k$.

It remains to show that if $\text{Ied}_\delta(\pi, \sigma) = k$, then a pair of insertion curves π' and σ' can be put together using the subcurves in $\mathcal{C}$ such that $d_{\mathcal{F}}(\pi', \sigma') \leq \delta$ and $|\pi'| + |\sigma'| - n - m \leq k$. Since $\text{Ied}_\delta(\pi, \sigma) = k$, there exists a pair π^* and σ^* of insertion curves and a matching M^* that realise $\text{Ied}_\delta(\pi, \sigma)$. Our proof will use M^* as a guide and iteratively transform π^* into π' and transform σ^* into σ', so that π' and σ' are put together by subcurves in $\mathcal{C}$. If π^* or σ^* uses the segments of the original curve, then π' or σ' use those segments as well. It remains to consider the subcurves that contain inserted vertices.

We will follow the matching M^* until it first matches an inserted curve starting at π_i (see Figure 3.14). As shown in Lemma 39, we adjust M^* so that π_i matches $\sigma^*[\alpha]$ on the boundary

of $\mathbf{b}(\pi_i)$ and π_{i+1} matches $\sigma^*[\beta]$ on the boundary of $\mathbf{b}(\pi_{i+1})$. Set $\pi' = \pi[1, i]$ and $\sigma' = \sigma[1, \alpha]$. For any pair of vertices σ_h and σ_{h+1} that lie in $\sigma^*[\alpha, \beta]$, we use the method in Observation 49 to remove the inserted vertices in $\sigma^*[h, h+1]$ and its matching subcurve in π^* before adding two vertices to π^*.

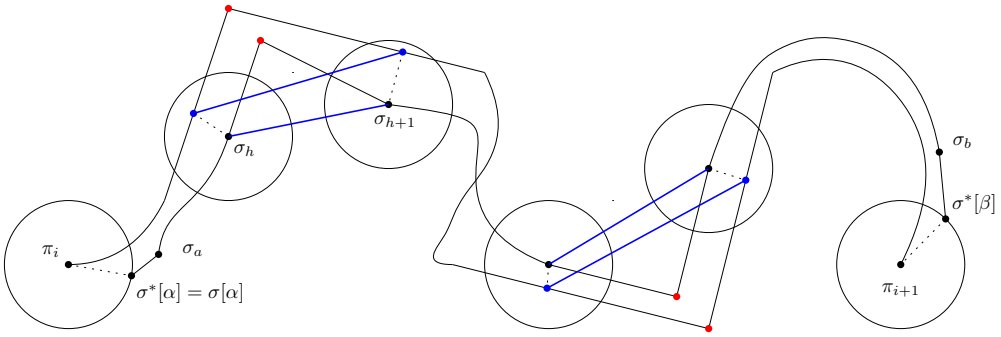

FIGURE 3.14: The newly inserted blue vertices can be paid for by removing the red vertices.

Let $\sigma_a \in \sigma^*$ be the first vertex of the original curve after $\sigma^*[\alpha]$, and let σ_b be the last vertex of the original curve before $\sigma^*[\beta]$. If $\sigma^*[\beta]$ lies on the original curve σ (Case 1), then the entire $\sigma[\alpha, \beta]$ is matched to an insertion curve $\pi^*[i, i+1]$ after transformation. By Fact 33 and Observation 34, $P = mv(\pi_i, \pi_{i+1}, \sigma[a, b])$ is equivalent to $mv(\pi_i, \pi_{i+1}, \sigma[\alpha, \beta])$ and uses the minimum number of inserted vertices. We append $\pi_i \circ P \circ \pi_{i+1}$ to π' and append $\sigma[\alpha, \beta]$ to σ'. The subcurve P is in $\mathcal{C}$.

If $\sigma^*[\beta]$ lies on an inserted subcurve (Case 2), we first append $\pi_i \circ P \circ \pi_{i+1}$ to π and append $\sigma[\alpha, b]$ to σ'. Then, we apply Observation 49 again, this time deleting inserted vertices from $\pi^*[i+1, n]$ and inserting vertices into $\sigma^*[b, b+1]$. After the transformation, $\sigma^*[b+1]$ matches to a point $\pi^*[\gamma]$ (see Figure 3.15), where $\pi^*[\gamma]$ may be a point on the original curve (Case 1) or may not (Case 2).

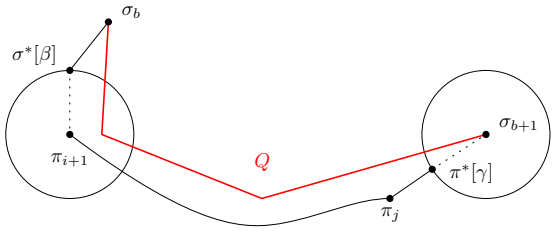

FIGURE 3.15: Use Observation 49 to guarantee that $\pi^*[i+1, j] = \pi[i+1, j]$.

If this is Case 1, we use $Q = mv(\sigma_b, \sigma_{b+1}, \pi[i+1, j])$ to replace $\sigma^*[b, b+1]$, where π_j is the last vertex of π before $\pi[\gamma]$. We append $\pi[i+1, \gamma]$ to π' and append $\sigma_b \circ Q \circ \sigma_{b+1}$ to σ'. In total, we obtain $\pi'[i, \gamma] = P \circ \pi[i+1, \gamma]$ and $\sigma'[\alpha, b+1] = \sigma[\alpha, b] \circ Q$.

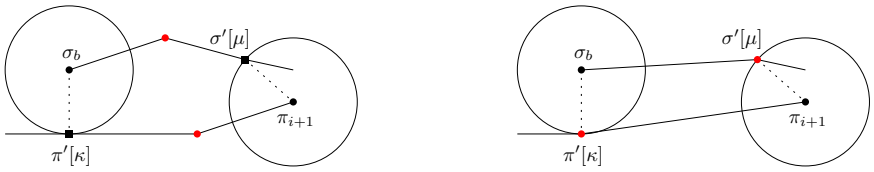

FIGURE 3.16: The subcurves $\pi'[\kappa, i+1]$ and $\sigma'[b, \mu]$ can be adjusted to be segments without having more inserted vertices.

It is clear that the subcurve $\pi'[i, \kappa]$ is similar to $\sigma'[a, b]$, and $\sigma'[\mu, b+1]$ is similar to $\pi'[i+1, \gamma]$, where $\pi'[\kappa]$ matches σ_b and $\sigma'[\mu]$ matches π_{i+1} (see Figure 3.16). It remains to prove that $s_1 = \pi'[\kappa, i+1]$ and $s_2 = \sigma'[b, \mu]$ are similar (i.e., that $d_{\mathcal{F}}(s_1, s_2) \leq \delta$). Observe that although the algorithm in [99] for computing a minimum-vertex curve does not guarantee that s_1 is a segment, we can modify s_1 by removing intermediate vertices and adding $\pi'[\kappa]$ as a vertex. An analogous adjustment can be applied to s_2 to ensure that it is also a segment. Now that s_1 and s_2 are segments, they are similar under linear interpolation, since $\pi'[\kappa] \in \mathbf{b}(\sigma_b)$ and $\sigma'[\mu] \in \mathbf{b}(\pi_{i+1})$.

If Case 2 occurs again, we continue this transformation process and alternate between inserting more vertices in π^* or in σ^*. Analogous arguments can be applied to show that π' and σ' are similar. In every case, we either use segments of the original curves or the subcurves in $\mathcal{C}$ while maintaining $d_{\mathcal{F}}(\pi', \sigma') \leq \delta$. Eventually, M^* matches π_n to σ_m, and the proof is complete. □

Using the above lemma, we have shown that a pair of optimal insertion curves can be put together using the subcurves in $\mathcal{C}$. Recall that we have assigned the weights of all subcurves in $\mathcal{C}$ to the augmented graph for insertions. We next show that it is sufficient to compute a weighted shortest path in $\mathcal{G}$ in order to determine $\text{Ied}_\delta(\pi, \sigma)$.

LEMMA 51. *Let π^* and σ^* be insertion curves obtained from π and σ using subcurves in $\mathcal{C}$ such that (i) the number of inserted vertices is minimal, and let this minimum be k, and*

(ii) the Fréchet distance between them is at most $d_{\mathcal{F}}(\pi^*, \sigma^*) \leq \delta$. Algorithm 7 will output a shortest path in $\mathcal{G}$ of weight $\mathbf{w}(v(t)) = k$.

PROOF. Recall that in Algorithm 7, $\mathcal{G}$ is the union of the freespace graph and the augmented graphs for both σ- and π-insertions. For every integer coordinate (i, j), $\mathcal{G}$ contains two so-called alternating edges, namely $(\pi^{\text{in}}_{i,j}, \sigma^{\text{out}}_{i,j})$ and $(\sigma^{\text{in}}_{i,j}, \pi^{\text{out}}_{i,j})$, each with weight 0.

We first prove that if a path $\mathcal{P}$ exists with $\mathbf{w}(\mathcal{P}) = k$, then there exist two insertion curves π' and σ' that can be put together using subcurves in $\mathcal{C}$ such that $d_{\mathcal{F}}(\pi', \sigma') \leq \delta$ and $|\pi'| + |\sigma'| - n - m \leq k$. To do so, we construct, from every consecutive set of edges in $\mathcal{P}$, a subcurve in $\mathcal{C}$ to put together π' and σ'. Initially, we set $\pi' = \pi_1$ and $\sigma' = \sigma_1$.

We partition $\mathcal{P}$ into a maximal set of subpaths, each subpath P travelling from $v(p)$ to $v(q)$, where $p = (x, y)$ and $q = (x', y')$ are critical points in $\mathcal{D}_\delta(\pi, \sigma)$.

Case 1. If P contains only freespace edges, then simply append $\pi[x, x']$ to π' and $\sigma[y, y']$ to σ'. Clearly, π' and σ' remain similar with no inserted vertices.

Case 2. If P contains edges in the augmented graph, then, due to the structure of $\mathcal{G}$, p is either a left critical point or a bottom critical point. Without loss of generality, let $p = \mathbf{l}_{i,j} = (\alpha_{i,j}, j)$.

Case 2a. If P contains edges in an augmented graph but no alternating edges, then by construction P starts at $v(p)$, proceeds to the dummy vertex $\sigma^{\text{out}}_{i,j}$, then to $\sigma^{\text{in}}_{a,j+1}$ for some integer a, and finally arrives at $v(\mathbf{r}_{a,j+1})$, where $\mathbf{r}_{a,j+1} = (\beta_{a,j+1}, j + 1)$. In this case, P uses edges only from the augmented graph for σ-insertions. Let $Q = mv(\sigma_j, \sigma_{j+1}, \pi[i, a])$. We append $\sigma_j \circ Q \circ \sigma_{j+1}$ to σ' and append $\pi[\alpha_{i,j}, \beta_{a,j+1}]$ to π'. By Fact 31, $d_{\mathcal{F}}(\pi[\alpha_{i,j}, \beta_{a,j+1}], \sigma_j \circ Q \circ \sigma_{j+1}) \leq \delta$. The insertion curves π' and σ' gain exactly $|Q|$ new vertices, which equals the weight of the edge $(\sigma^{\text{out}}_{i,j}, \sigma^{\text{in}}_{a,j+1})$.

Case 2b. If P contains at least one alternating edge, then P contains more than two dummy vertices. In this case, we focus on the dummy vertices. The point q is either a top critical point or a right critical point. Without loss of generality, for some integers $a_1, a_2, ..., a_h$ and $b_1, b_2, ..., b_h$, let $q = \mathbf{r}_{a_h, b_h}$. The graph path P visits the following dummy vertices in this

specific order, starting from $a_1 = i$ and $b_1 = j$.

$$P = p \to \sigma^{\text{out}}_{a_1,b_1} \to \sigma^{\text{in}}_{a_2,b_1+1} \to \pi^{\text{out}}_{a_2,b_1+1} \to \pi^{\text{in}}_{a_2+1,b_2} \to \sigma^{\text{out}}_{a_2+1,b_2} \to \sigma^{\text{in}}_{a_3,b_2+1} \to \dots$$
$$\to \pi^{\text{out}}_{a_{h-1}-1,b_{h-1}} \to \pi^{\text{in}}_{a_{h-1},b_h-1} \to \sigma^{\text{out}}_{a_{h-1},b_h-1} \to \sigma^{\text{in}}_{a_h,b_h} \to q$$

We first append $\pi[\alpha_{a_1,b_1}, a_1]$ to π'. Then, we follow the edges of P. For every edge $(\pi^{\text{out}}_{a_g,b_g}, \pi^{\text{in}}_{a_g+1,b_{g+1}})$, we append $mv(\pi_{a_g}, \pi_{a_g+1}, \sigma[b_g, b_{g+1}])$ to π' and append $\sigma[b_g, b_{g+1}]$ to σ'. For every edge $(\sigma^{\text{out}}_{a_g,b_g}, \sigma^{\text{in}}_{a_{g+1},b_g+1})$, we append $\pi[a_g, a_{g+1}]$ to π' and $mv(\sigma_{b_g}, \sigma_{b_g+1}, \pi[a_g, a_{g+1}])$ to σ'. Finally, we append $\pi[a_h, \beta_{a_h,b_h}]$ to π'. We have appended the curve π'' to π' and σ'' to σ', where π'' and σ'' are defined as follows:

$$\pi'' = \pi[\alpha_{a_1,b_1}, a_1] \circ \pi[a_1, a_2] \circ mv(\pi_{a_2}, \pi_{a_2+1}, \sigma[b_1 + 1, b_2]) \circ \dots \circ \pi[a_{h-1}, a_h] \circ \pi[a_h, \beta_{a_h,b_h}]$$
$$\sigma'' = mv(\sigma_{b_1}, \sigma_{b_1+1}, \pi[a_1, a_2]) \circ \sigma[b_1 + 1, b_2] \circ \dots \circ mv(\sigma_{b_h-1}, \sigma_{b_h}, \pi[a_{h-1}, a_h]).$$

Since both $p = (\alpha_{a_1,b_1}, b_1)$ and $q = (\beta_{a_h,b_h}, b_h)$ are critical points in the freespace, we have $\|\sigma_{b_1} - \pi[\alpha_{a_1,b_1}]\| \le \delta$ and $\|\sigma_{b_h} - \pi[\beta_{a_h,b_h}]\| \le \delta$. Using the same argument as Lemma 50, we can establish that $d_{\mathcal{F}}(\pi'', \sigma'') \le \delta$, and π'' and σ'' have exactly the same number of additional vertices as $\mathbf{w}(P)$. Conversely, if π' and σ' exist, then the path $\mathcal{P}$ can be analogously derived. □

LEMMA 52. *Given two polygonal curves π and σ in $\mathbb{R}^2$ with complexity n and m, respectively, it takes $\mathcal{O}(k(m+n)\mathcal{M}(m+n))$ time to determine whether $\text{Ied}_\delta(\pi, \sigma) \le k$, and it takes $\mathcal{O}(mn\mathcal{M}(m+n))$ time to compute $\text{Ied}_\delta(\pi, \sigma)$.*

PROOF. The correctness of Algorithm 7 is implied by Lemma 50 and 51. To achieve a better time complexity, we use a nearly identical analysis to that in Lemma 43. Instead of computing the entire graph $\mathcal{G}$ in Algorithm 7, we iterate $j = 1, ..., m$ and compute the weights of the vertices in $\mathcal{G}$ defined by j.

Recall that $\mathcal{A}_j$ is the set of indices i such that $\mathbf{l}_{i,j}$ exists. In Lemma 42, we showed that if we know the weights of the vertices defined by $\mathcal{A}_j$, then it takes $\mathcal{O}(k\mathcal{M}(n))$ time to compute the weights of the vertices defined by $\mathcal{A}_{j+1}$. Analogously, let $\mathcal{B}_i$ be the set of indices j such that $b_{i,j}$ exists, and it takes $\mathcal{O}(k\mathcal{M}(m))$ time to compute the weights of the vertices defined by $\mathcal{B}_{i+1}$.

It takes $\mathcal{O}(km\mathcal{M}(n))$ time to compute the necessary edge weights for edges in the augmented graph for σ-insertions, and $\mathcal{O}(kn\mathcal{M}(m))$ time for π-insertions. In total, it takes $\mathcal{O}(k(m\mathcal{M}(n) + n\mathcal{M}(m)))$ time to compute the minimum vertex curves. This time complexity dominates the size of the graph. □

THEOREM 10. *The insert-only Fréchet edit distance problem in $\mathbb{R}^2$ can be solved in $\widetilde{\mathcal{O}}(kn^3)$ time.*

3.6 Conclusion and future work

In this chapter, by embedding edges into the freespace diagram, we obtained faster algorithms for the Fréchet edit distance in three cases: delete-only, edits on one of the two curves, and insertions on both curves. Our approach relies on two ingredients. First, we showed that in the delete-only case it suffices to propagate information along the boundaries of the freespace diagram, rather than computing every cell generated by every segment that a deletion can produce.

Second, we extended the critical insight of [79] that it suffices to compute a finite set of insertion curves. We showed that an even smaller subset of insertion curves is sufficient, and that these insertion curves can be batch-computed more efficiently. We further extended this insight to the case where insertions are allowed on both curves, where it suffices to compute a set of insertion curves determined purely by the original curves.

A natural next step, given these more efficient algorithms, is to ask whether the time complexity can be improved further. In Section 3.3.5 we used the construction of [24] to prove a conditional lower bound for the delete-only case when the Fréchet distance is measured under

the $\mathcal{L}_p$ metric for $p \neq 2$. The case $p = 2$ remains open: is there an algorithm that solves the delete-only case in $\widetilde{\mathcal{O}}(n^2)$ time?

To improve the time complexity of our algorithms when insertions are allowed, a natural question is whether the minimum-link path algorithm itself can be sped up. More generally, one can ask whether there exists a subquadratic-time algorithm that computes the minimum-link path through a set of disks. If the goal is specifically to improve the Fréchet edit distance, it may be enough to answer the decision version: is there an $\widetilde{\mathcal{O}}(kn)$-time algorithm?

3.6.1 Edits on both curves

For the least restricted case, when deletions and insertions are allowed on both curves, the first question is whether the case is even meaningful to study. That is, are there two curves such that $\mathrm{Ed}_\delta(\pi, \sigma) < \min\{\mathrm{Ed}^1_\delta(\pi, \sigma), \mathrm{Ed}^1_\delta(\sigma, \pi)\}$? The answer is yes: allowing edits on both curves can be strictly cheaper (see Figure 3.17).

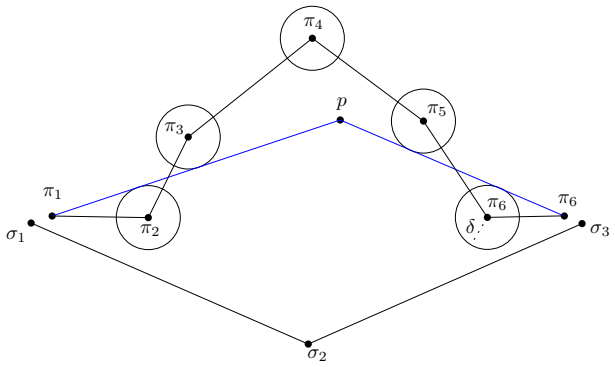

FIGURE 3.17: To realise $\mathrm{Ed}_\delta(\pi, \sigma)$, it suffices to delete σ_2 and π_4 and to insert p between σ_1 and σ_3. If edits are allowed on only one of the two curves, at least one additional edit is required.

It may be helpful to first consider an easier version of the problem.

PROBLEM 4. *Consider as input a segment st, and a curve π such that $\pi_1 \notin \mathbf{b}(s)$ and $\pi_n \notin \mathbf{b}(t)$. Compute a k_1-insertion curve σ' of st and a k_2-deletion curve π' of π with the following properties. The number of edits $k_1 + k_2$ is minimised and the curve σ' is a minimum vertex curve of π'.*

Problem 4 should be solvable when the curves lie in $1d$ (i.e. are intervals), because only a fixed number of turning points need to be considered for σ'. Given a minimum vertex curve σ' in $1d$, we hypothesise that σ' can be modified by shifting each of its vertices either left or right so that every vertex of σ' coincides with the endpoint of some interval (a $1d$ disk). This yields a finite set of candidate vertices for σ'; in $2d$ the situation is substantially harder.

Once Problem 4 is solved, it may be possible to embed the required edits into the augmented graph and compute the edit distance with edits on both curves via a shortest-path computation.

CHAPTER 4

Computing a Subtrajectory Cluster from c-Packed Trajectories

4.1 Introduction

With the proliferation of location-aware devices comes an abundance of trajectory data. One way to process and make sense of many trajectories is to group long and similar subtrajectories. The analysis of long and similar parts of trajectories can provide insights into behaviour and mobility patterns, such as common routes taken and places visited frequently.

Buchin, Buchin, Gudmundsson, Löffler, and Luo [36] initialised the study of subtrajectory cluster problems to detect and extract common movement patterns. The Subtrajectory Cluster (SC) decision problem is defined as follows. Given one or more trajectories, determine if there exists a cluster of $m-1$ non-overlapping subtrajectories and one reference trajectory. The reference trajectory π_r must be at least of length l, and the Fréchet distances between π_r and the other $m-1$ subtrajectories must be at most δ. In the case of animals, long and common movement patterns can indicate movement between grazing spots of sheep or the migration flyway of seabirds. In the case of humans, common movement on a Monday morning can show commuting patterns to find the most heavily congested areas.

Subtrajectory clustering has attracted research from multiple communities. [94] used subtrajectory cluster to analyse the common movement patterns of football players. [34] applied subtrajectory cluster to map reconstruction by clustering common movement patterns of vehicles into road segments. Researchers in the Geographical Information and Data Mining communities also considered the variants and practical performance of subtrajectory cluster algorithms [2, 51, 87, 92, 93, 120, 136]. In addition, the potential of SC is examined in

a wide range of applications, including sports player analysis [140] and human movement analysis [49, 107].

Several theoretical studies of the subtrajectory clustering problem focus on improving the quality of clustering. [2] defined a single objective function, the weighted sum of three quality measures of a clustering. These quality measures include the number of clusters chosen, the quality of the cluster, and the size of the trajectories excluded from the clustering. [30] studied so-called $\triangle$-coverage, aiming to find a set C of curves to cover a polygonal curve such that a curve in C is fixed in size, and $|C|$ is minimised.

However, despite considerable attention from multiple communities, there is no subcubic time algorithm that solves the subtrajectory cluster problem, limiting its usefulness on large data sets. [36] solved the subtrajectory cluster problem in $\mathcal{O}(n^5)$ time when the similarity measure of two trajectories is the Fréchet distance, and [95] further improved the runtime with an $\mathcal{O}(n^3 \log^2 n)$ time algorithm. In addition, [95] showed that there is no $\mathcal{O}(n^{3-\varepsilon})$ algorithm for subtrajectory cluster for any $\varepsilon > 0$ unless the Strong Exponential Time Hypothesis (SETH) fails.

SC is unlikely to have a strongly subquadratic algorithm even if we allow a small approximation factor on the Fréchet distances between subtrajectories, because given two trajectories π_1 and π_2, we can structure the SC problem to find two subtrajectories such that their Fréchet distance is at most $(1 + \varepsilon)\delta$, and the reference trajectory must be as long as the maximum of π_1 and π_2. Solving this instance of SC is equivalent to approximating the Fréchet distance of π_1 and π_2, and Bringmann [23] showed that there is no 1.001-approximation with runtime $\mathcal{O}(n^{2-\varepsilon})$ for the continuous Fréchet distance for any $\varepsilon > 0$, unless SETH fails.

Since an exact subcubic and an approximate subquadratic algorithm are unlikely to exist, we study subtrajectory cluster on a realistic family of trajectories, called c-packed trajectories. A trajectory π is c-packed if, for any ball B of radius r, the length of π lying inside B is at most c times r. The packedness value of a trajectory π is the maximum c for which π is c-packed. Bringmann [23] proved that computing the Fréchet distance has no strong subquadratic algorithm unless SETH fails, and the notion of c-packedness was introduced by Driemel,

Har-Peled, and Wenk [65] to circumvent such conditional lower bound. Since then, the notion of c-packedness has gained considerable attention from the theory community [3, 23, 25, 52, 91], and several real-world data sets have been shown to have low packedness values [65, 90]. In one particular instance, [90] approximated the packedness values of several real-world trajectory data sets. In their experiments, several trajectory data sets have low packedness values, such as the movement patterns of people in Beijing, school buses, European football players, and trawling bats.

In this chapter, given a c-packed trajectory π of complexity n and a desired multiplicative approximation error ε on the Fréchet distance between subtrajectories, we present an $\mathcal{O}((c^2n/\varepsilon^2)\log(c/\varepsilon)\log(n/\varepsilon))$ time algorithm that solves the SC problem. It is worth noting that previous papers considering c-packed curves typically replace a factor n with a polynomial of constant degree in c [62, 88]. We are able to replace a factor of n^2 with c^2/ε^2, bringing the algorithm's running time from cubic to near-linear, assuming $c \in \mathcal{O}(1)$.

Along the way, we develop a tool for simplifying the freespace diagram that may be of independent interest. To efficiently approximate the Fréchet distance, [65] showed that the freespace complexity, i.e., the number of non-empty cells, is $\mathcal{O}(cn/\varepsilon)$ for two simplified c-packed trajectories (see Section 4.2 for an overview of the freespace, or [7] for a formal definition). However, simplifying a trajectory by taking shortcuts between vertices can yield a much shorter trajectory, and the SC problem is sensitive to the length of the trajectories since the reference trajectory has to have a length at least l. To tackle this problem, we developed a tool to construct the freespace diagram in $\mathcal{O}((cn/\varepsilon)\log(cn/\varepsilon))$ time, preserving the length of two trajectories while benefiting from the $\mathcal{O}(cn/\varepsilon)$ freespace complexity. Our tool can be of value for problems in which the length of a trajectory is important, such as subtrajectory cluster [36], partial curve matching [31], and Fréchet distance with speed limit [124].

In Section 4.2, we will formally define the subtrajectory cluster problem and outline the greedy plane sweep algorithms of [36] and [95], which our approach builds on. In Section 4.3, we provide a technical overview of our main results. In Section 4.4, we will discuss how to simplify the freespace diagram to achieve a lower complexity while preserving trajectory lengths.

In Section 4.5, we will consider the restricted case when the reference trajectory must be vertex-to-vertex. In Section 4.6, we will remove this restriction by considering an arbitrary reference trajectory.

4.2 Preliminaries

In this section, we outline previous algorithms for the subtrajectory cluster problem. The subtrajectory cluster problem was first introduced by [36] and later improved by [95]. Instead of looking for a subtrajectory where the Fréchet distances between the reference trajectory and the subtrajectories are exact, we aim to find a solution that approximates the Fréchet distance between subtrajectories in the cluster.

PROBLEM 5 ([95]). *Given a trajectory π of complexity n, a positive integer m, and positive real numbers δ, l, and ε, decide whether there exists a subtrajectory cluster of π such that:*

- *the cluster consists of one reference subtrajectory and $m-1$ other subtrajectories of π,*
- *the reference subtrajectory has Euclidean length at least l,*
- *the Fréchet distance between the reference subtrajectory and any other subtrajectory is at most $(1+\varepsilon)\delta$,*
- *any pair of subtrajectories in the cluster overlap in at most one point.*

[36] solved the exact SC problem using a plane-sweep algorithm on the freespace diagram $\mathcal{D}_\delta(\pi,\pi)$. Let s and t be two points on π, and denote by π_{st} the subtrajectory of π starting at s and ending at t. Let l_s and l_t be the vertical sweep lines $x=s$ and $x=t$ on $\mathcal{D}_\delta(\pi,\pi)$, respectively (see Figure 4.1). An xy-monotone path in $\mathcal{D}_\delta(\pi,\pi)$, or xy-monotone path for short, is a continuous path that is non-decreasing in both x- and y-coordinates. To solve $SC(m,\delta,l)$, the lines l_s and l_t sweep from left to right while ensuring that l_s remains to the left of l_t, and that the reference trajectory π_{st} is at least l long, i.e., $t-s\geq l$. In each interval $[l_s,l_t]$, they compute the maximum number of xy-monotone paths in $\mathcal{D}_\delta(\pi,\pi)$ starting at l_s and ending at l_t.

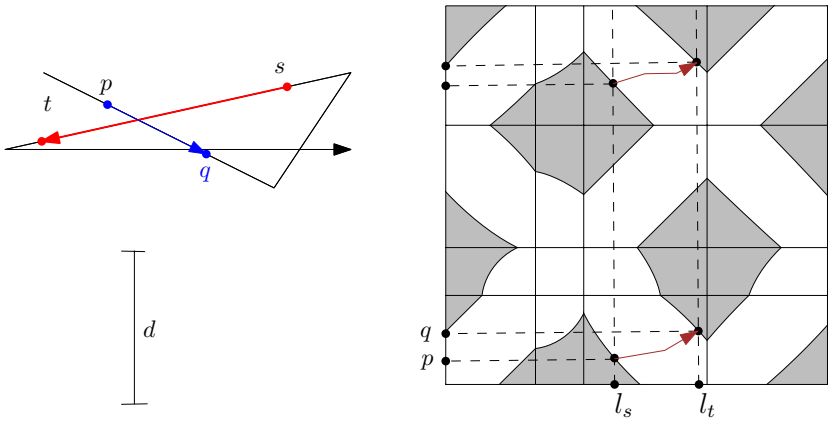

FIGURE 4.1: The freespace diagram $\mathcal{D}_\delta(\pi, \pi)$ and the interval $[l_s, l_t]$ are defined by two points s and t on π. If there exists an xy-monotone path (marked in brown) from (s, p) to (t, q) through the freespace, then the Fréchet distance between the subtrajectories π_{st} and π_{pq} is at most δ.

Let s and t be two points on π, and let (s, p) and (t, q) be two coordinates on $\mathcal{D}_\delta(\pi, \pi)$. As we only consider xy-monotone paths starting at l_s and ending at l_t, we call the xy-monotone path from (s, p) to (t, q) the pq xy-monotone path. First, an xy-monotone path pq must traverse only the freespace. Second, two xy-monotone paths pq and ab must not overlap along the y-interval in more than a single point. Third, the y-coordinates of any pq xy-monotone path cannot overlap the interval $[s, t]$ in more than a single point. We obtain the following subproblem.

SUBPROBLEM 6 ([95]). *Given a trajectory π of complexity n, a positive integer m, a positive real value δ, and a reference subtrajectory of π starting at s and ending at t, let l_s and l_t be two vertical lines in $\mathcal{D}_\delta(\pi, \pi)$ representing the points s and t. Decide whether there exist:*

- *$m - 1$ distinct paths starting at l_s and ending at l_t such that*
- *the y-coordinates of any two xy-monotone paths overlap in at most one point, and*
- *the y-coordinate of any xy-monotone path overlaps the y-interval from s to t in at most one point.*

To find a set $\{p_1q_1, p_2q_2, \ldots, p_{m-1}q_{m-1}\}$ of xy-monotone paths, both algorithms use a greedy approach. First, set p_1 to be the lowest feasible point on l_s, and compute p_1q_1 by searching for the lowest xy-monotone path through the freespace. Inductively, with $p_{i-1}q_{i-1}$ computed, set

p_i to the lowest feasible point on l_s that is on or above q_{i-1}, and proceed similarly. If a search from p_i leads to a dead end, we simply set p_i to the next lowest feasible point on l_s and search again.

The sweep lines stop at all $\mathcal{O}(n^3)$ critical points, and for each critical point there is an interval $[l_s, l_t]$ to consider. [36] solved each instance in $\mathcal{O}(nm) \subseteq \mathcal{O}(n^2)$ time. [95] improved the efficiency by connecting the critical points in a tree-like data structure that allows them to reuse computed xy-monotone paths from previous interval instances. They showed that, in their construction, there are at most $\mathcal{O}(n^3 \log n)$ edges, and each edge takes at most $\mathcal{O}(\log n)$ time to add, remove, or access. This reduces the complexity of the algorithm from $\mathcal{O}(n^5)$ to $\mathcal{O}(n^3 \log^2 n)$ time.

4.3 Technical Overview

Our technical overview is divided into three parts. In Sections 4.3.1, 4.3.2, and 4.3.3, we summarise the main result of Sections 4.4, 4.5, and 4.6 respectively.

4.3.1 Computing the Freespace Diagram

Our algorithm constructs a simplified freespace diagram that preserves trajectory lengths. The size (in terms of Euclidean length) of the simplified freespace diagram is the same as that of the unsimplified freespace diagram. The only difference between the two diagrams is that approximate distances are used in the simplified diagram. In particular, we define a function that uniformly maps a trajectory to its simplification, and we calculate the distances between the mapped simplification points instead of the points on the original trajectory. We prove that the complexity of the simplified freespace diagram is at most $\mathcal{O}(cn/\varepsilon)$, and that the trajectory lengths in the diagram are preserved. Next, we build the simplified freespace diagram. We use an algorithm by Conradi and Driemel [62] to query pairs of nearby segments. Finally, we construct a data structure on the freespace diagram so that we can access the closest non-empty cells below, above, to the left, and to the right in constant time. Putting this all together, we obtain Theorem 11. For a full proof, see Section 4.4.

THEOREM 11. *Given a pair of trajectories, one can construct a simplified freespace diagram in $\mathcal{O}((cn/\varepsilon)\log(cn/\varepsilon))$ time, so that the simplified freespace has complexity $\mathcal{O}(cn/\varepsilon)$, it approximates the Fréchet distance to within a factor of $(1+\varepsilon)$, and it preserves the trajectory lengths of the original trajectory.*

4.3.2 Reference trajectory is vertex-to-vertex

Next, we focus on the special case where the reference trajectory is vertex-to-vertex. Three data structures are used in the vertex-to-vertex subtrajectory cluster algorithm of [95] — a directed graph, a range tree, and a link-cut tree. Originally, the number of leaves per range tree is $\mathcal{O}(n)$, and the directed graph has complexity $\mathcal{O}(n^2)$. We use the c-packedness property to prove that, in our simplified freespace diagram, the number of leaves per range tree is $\mathcal{O}(c/\varepsilon)$, and the directed graph has complexity $\mathcal{O}((cn/\varepsilon)\log(c/\varepsilon))$. The link-cut tree data structure can be used without modification. Putting this all together, we obtain Theorem 12. Recall that m is the desired number of subtrajectories in the cluster. For a full proof, see Section 4.5.

THEOREM 12. *There is an $\mathcal{O}(nm\log(c/\varepsilon)\log(n/\varepsilon))$ time algorithm that solves $SC(T, m, l, (1+\varepsilon)\delta)$ in the case that the reference trajectory is vertex-to-vertex.*

4.3.3 Reference trajectory is arbitrary

Finally, we tackle the general case where the reference trajectory is arbitrary. The main obstacle in the general case is that there are $\Theta(n^3)$ internal critical points that correspond to potential starting and ending positions of the reference trajectory. In fact, [95] showed that, for general (non–c-packed) curves, these internal critical points are essentially unavoidable. They use the $\Theta(n^3)$ internal critical points to show that, under the Strong Exponential Time Hypothesis (SETH), there is no $\mathcal{O}(n^{3-\varepsilon})$ time algorithm for subtrajectory cluster for any $\varepsilon > 0$.

Our main lemma in this section is to bound the number of internal critical points for subtrajectory cluster on c-packed trajectories. The lemma uses the c-packedness property in two different ways. First, the c-packedness property bounds the complexity of the simplified

freespace diagram to linear, which replaces one of the factors of n with c/ε. Second, the c-packedness property is used to prove that, in any horizontal strip, only a constant number of cells have freespace. This replaces another factor of n with c/ε, resulting in $\mathcal{O}(c^2 n/\varepsilon^2)$ internal critical points. Finally, we prove that the interval management data structure can be used in the same way as in [95]. Putting this all together, we obtain Theorem 13. For a full proof, see Section 4.6.

THEOREM 13. *There is an $\mathcal{O}((c^2 n/\varepsilon^2)\log(c/\varepsilon)\log(n/\varepsilon))$ time algorithm that solves $SC(T, m, l, (1+\varepsilon)\delta)$ in the case that the reference trajectory is arbitrary.*

4.4 Computing the Freespace Diagram

In this section, we explain the process of constructing a simplified freespace diagram for two c-packed polygonal curves π and σ. The freespace $\mathcal{F}_\delta(\pi, \sigma)$ describes all pairs of points, one on π and one on σ, whose distance is at most δ [7]. With slight abuse of notation, we parameterise the polygonal curve π such that $\pi[x]$ is a point on π, where $x \in [0, \|\pi\|]$. Formally,

$$\mathcal{F}_\delta(\pi, \sigma) = \{(x, y) \in [0, \|\pi\|] \times [0, \|\sigma\|] \mid \|\pi[x] - \sigma[y]\| \leq \delta\}.$$

To circumvent the quadratic freespace complexity, [65] showed that the freespace complexity of two simplified c-packed curves is $\mathcal{O}(cn/\varepsilon)$. Given a c-packed curve $\pi = p_1 p_2 \ldots p_n$, we simplify π into its $\varepsilon\delta$-simplification $\pi' = \text{simpl}(\pi, \varepsilon\delta) = q_1 q_2 \ldots q_k$ as follows. Let $\mathbf{b}(a, r)$ be the ball centred at a with radius r. First, set $q_1 = p_1$. With q_i defined, traverse π from q_i until a vertex v is outside $\mathbf{b}(q_i, \varepsilon\delta)$ or v is the last vertex of π, and set $q_{i+1} = v$. Continue until all vertices of π are exhausted. [65] showed that the $\varepsilon\delta$-simplification of a c-packed curve is at most $6c$-packed [65, Lemma 4.3], and that the Fréchet distance between π and π' is at most $(1+\varepsilon)\delta$. With a slight adjustment, every segment of a simplified curve can be made at least $\varepsilon\delta$ long.

OBSERVATION 53. *One can simplify a polygonal curve π into its $\varepsilon\delta$-simplification π' such that the Fréchet distance between π and π' is at most $(1+\varepsilon)\delta$, and each segment in π' is at least $\varepsilon\delta$ long.*

Simplifying two c-packed curves can reduce the freespace complexity, but using the plane-sweep algorithm to solve the SC problem on the resulting freespace diagram is unfortunately infeasible. This is because the total length of the simplified trajectories can be much shorter, making it impossible to slide a window of width l on the freespace diagram $\mathcal{D}_{(1+\varepsilon)\delta}(\pi', \sigma')$. To address this issue, we develop a tool that enables the construction of a freespace diagram that maintains the original curve length while also benefiting from the reduced freespace complexity.

4.4.1 Simplifying the Freespace

In this section, we introduce a method that simplifies the freespace. We show that we can construct the *simplified freespace* $\mathcal{F}^s_{(1+\hat{\varepsilon})\delta}(\pi, \sigma)$, where $\hat{\varepsilon}$ is at most 8ε, such that the complexity of the simplified freespace is at most $\mathcal{O}(cn/\hat{\varepsilon})$. In addition, $\mathcal{F}^s_{(1+\hat{\varepsilon})\delta}(\pi, \sigma)$ contains $\mathcal{F}_\delta(\pi, \sigma)$ as a subset, but it is not larger than the freespace of π and σ if we approximate their Fréchet distance; that is,

$$\mathcal{F}^s_{(1+\hat{\varepsilon})\delta}(\pi, \sigma) \subseteq \mathcal{F}_{(1+\hat{\varepsilon})\delta}(\pi, \sigma).$$

We first define a function that uniformly maps parts of the polygonal curve π to segments of π' in Definition 54, which we then use to formally define the simplified freespace in Definition 55. We then formally prove the set inclusions mentioned above in Lemma 56.

DEFINITION 54. *Let π_{uv} be the subcurve of π from point u to v that is simplified into the segment $(u, v) \in \text{simpl}(\pi, \varepsilon\delta)$. Let w be the first intersection point of π_{uv} and the boundary of the ball $\mathbf{b}(u, \varepsilon\delta)$ along π_{uv}, and let u' be the intersection of (u, v) with the boundary of the ball $\mathbf{b}(u, \varepsilon\delta)$. Define the mapping $f_{\pi,\varepsilon\delta} : \pi \to \text{simpl}(\pi, \varepsilon\delta)$ such that $f_{\pi,\varepsilon\delta}$ maps $[u, w)$ to $[u, u')$ uniformly, and maps $[w, v]$ to $[u', v]$ uniformly (see Figure 4.2).*

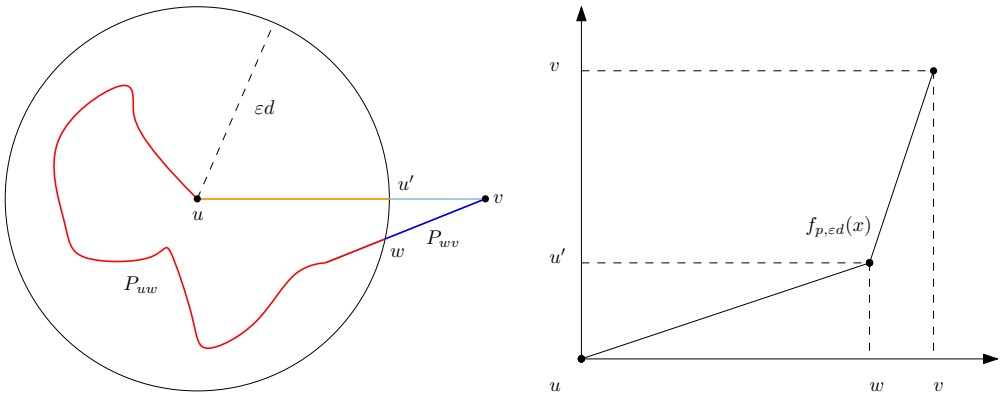

FIGURE 4.2: A figure showcasing the function in Definition 54. The point u' is the intersection of the segment (u, v) with the ball $\mathbf{b}(u, \varepsilon\delta)$, and the point w is the intersection of the subtrajectory π_{uv} with $\mathbf{b}(u, \varepsilon\delta)$. The function $f_{\pi,\varepsilon\delta}$ uniformly maps π_{uw} (red) to (u, u') (orange), not including u' and w. The function $f_{\pi,\varepsilon\delta}$ uniformly maps π_{wv} (blue) to (u', v) (light blue).

DEFINITION 55. *Define the* simplified freespace *of π and σ with respect to the Fréchet distance $\delta > 0$ and a parameter $\varepsilon > 0$ as*

$$\mathcal{F}^s_{(1+\varepsilon)\delta}(\pi, \sigma) = \{(x, y) \in [0, \|\pi\|] \times [0, \|\sigma\|] \mid \|f_{\pi,\varepsilon\delta}(\pi[x]) - f_{\sigma,\varepsilon\delta}(\sigma[y])\| \leq (1 + \varepsilon)\delta\}.$$

Similarly, let $\mathcal{D}^s_{(1+\varepsilon)\delta}(\pi, \sigma)$ be the simplified freespace diagram.

LEMMA 56. *For $\varepsilon > 0$, $\mathcal{F}_\delta(\pi, \sigma) \subseteq \mathcal{F}^s_{(1+4\varepsilon)\delta}(\pi, \sigma) \subseteq \mathcal{F}_{(1+8\varepsilon)\delta}(\pi, \sigma)$.*

PROOF. With slight abuse of notation, let $x = \pi[x]$ and $y = \sigma[y]$, for $x \in [0, \|\pi\|]$ and $y \in [0, \|\sigma\|]$. Let $x' = f_{\pi,\varepsilon\delta}(x)$ and $y' = f_{\sigma,\varepsilon\delta}(y)$. Observe that $\|x - x'\| \leq 2\varepsilon\delta$ for all $x \in \pi$ because, if x is within the ball $\mathbf{b}(u, \varepsilon\delta)$, then x is at most $2\varepsilon\delta$ away from x'. If x is outside $\mathbf{b}(u, \varepsilon\delta)$, it is at most $\varepsilon\delta$ away from x' due to the simplification.

To show $\mathcal{F}_\delta(\pi, \sigma) \subseteq \mathcal{F}^s_{(1+4\varepsilon)\delta}(\pi, \sigma)$: if a point $(x, y) \in \mathcal{F}_\delta(\pi, \sigma)$, then $\|x - y\| \leq \delta$. By the triangle inequality,

$$\|x' - y'\| \leq \|x' - x\| + \|y' - y\| + \|x - y\| \leq 2\varepsilon\delta + 2\varepsilon\delta + \delta = (1 + 4\varepsilon)\delta,$$

hence (x', y') must also be in $\mathcal{F}^s_{(1+4\varepsilon)\delta}(\pi, \sigma)$.

To show $\mathcal{F}^s_{(1+4\varepsilon)\delta}(\pi,\sigma) \subseteq \mathcal{F}_{(1+8\varepsilon)\delta}(\pi,\sigma)$: if a point $(x',y') \in \mathcal{F}^s_{(1+4\varepsilon)\delta}(\pi,\sigma)$, then

$$\|x-y\| \leq \|x'-x\| + \|y'-y\| + \|x'-y'\| \leq 2\varepsilon\delta + 2\varepsilon\delta + (1+4\varepsilon)\delta = (1+8\varepsilon)\delta,$$

so $(x,y) \in \mathcal{F}_{(1+8\varepsilon)\delta}(\pi,\sigma)$. □

Similar to how we defined the (u,v) cell, let the (π_{uv},σ_{ab}) cells be the cells in the freespace diagram defined by the subcurves π_{uv} and σ_{ab}. We show that we can compute the intersection of the simplified freespace with the (π_{uv},σ_{ab}) cells in constant time.

LEMMA 57. *Given vertices u,v on π and a,b on σ, one can construct the cells in $\mathcal{D}^s_{(1+\varepsilon)\delta}(\pi,\sigma)$ defined by π_{uv} and σ_{ab} in constant time.*

PROOF. The following definition follows from the definition of $f_{\pi,\varepsilon\delta}$. Let u' be the first intersection of $\mathbf{b}(u,\varepsilon\delta)$ and (u,v) along $\mathrm{simpl}(\pi,\varepsilon\delta)$, and let a' be the first intersection of $\mathbf{b}(a,\varepsilon\delta)$ and (a,b) along $\mathrm{simpl}(\sigma,\varepsilon\delta)$. Let w be the first intersection of $\mathbf{b}(u,\varepsilon\delta)$ and π_{uv}, and let c be the first intersection of $\mathbf{b}(a,\varepsilon\delta)$ and σ_{ab}.

Consider a partition of the (π_{uv},σ_{ab}) cells into four cells generated from π_{uw}, π_{wv}, σ_{ac}, and σ_{cb}. Define W to be the rectangle $[0,\|\pi_{uw}\|] \times [0,\|\sigma_{ac}\|]$. We will show that the intersection of the simplified freespace and the (π_{uw},σ_{ac}) cells is an ellipse clipped at W, using the arguments in the proof of [101, Lemma 30.2.1].

With slight abuse of notation, we redefine π_{uw} as the affine mapping from $[0,\|\pi_{uw}\|]$ to points on π_{uw}. The function $f_{\pi,\varepsilon\delta} \circ \pi_{uw}$ is an affine function, as the composition of two affine functions $f_{\pi,\varepsilon\delta}$ and π_{uw} is also affine, and similarly, $f_{\sigma,\varepsilon\delta} \circ \sigma_{ab}$ is affine. Therefore, $h : (x,y) \mapsto f_{\pi,\varepsilon\delta}(\pi_{uw}(x)) - f_{\sigma,\varepsilon\delta}(\sigma_{ab}(y))$ is an affine function. Assuming general position of (u,v) and (a,b), h is also one-to-one. All the desired configurations of (x,y) satisfying $\|h(x,y)\| \leq \delta$ are mapped to the disk B of radius δ centred at the origin.

Consider the intersection of B and the image of h, i.e., $h(\mathbb{R}^2) \cap B$. Then the simplified freespace of the (π_{uw},σ_{ac}) cells is the set $h^{-1}(h(\mathbb{R}^2) \cap B) \cap W$. The inverse of an affine function is also an affine function, and thus so is h^{-1}. The affine image of a disk is an ellipse,

and therefore so is $h^{-1}(h(\mathbb{R}^2) \cap B)$. Consequently, the (π_{uw}, σ_{ac}) cell is an ellipse clipped into a rectangle, and it takes constant time to compute.

The above arguments extend to the other three cells. Therefore, constructing the cells defined by π_{uw} and σ_{ab} in $\mathcal{D}^s_{(1+\varepsilon)\delta}(\pi, \sigma)$ takes constant time. □

The complexity of the simplified freespace $\mathcal{F}^s_{(1+\hat{\varepsilon})\delta}(\pi, \sigma)$ is $\mathcal{O}(cn/\hat{\varepsilon})$ if π and σ are c-packed. Assuming that π_{uv} and σ_{ab} are simplified into the segments $(u, v) \in \pi'$ and $(a, b) \in \sigma'$, respectively, the simplified freespace intersects the (π_{uv}, σ_{ab}) cells if and only if the distance between (u, v) and (a, b) is at most $(1 + \hat{\varepsilon})\delta$. The rest follows by modifying the proof of [65, Lemma 4.4].

COROLLARY 58. *Let π and σ be two c-packed curves with complexity n, and let $\hat{\varepsilon}$ be a constant times a parameter $\varepsilon > 0$. The complexity of the simplified freespace $\mathcal{F}^s_{(1+\hat{\varepsilon})\delta}(\pi, \sigma)$ is $\mathcal{O}(cn/\varepsilon)$.*

4.4.2 Compute the Non-empty Cells

To take advantage of the near-linear complexity of the simplified freespace, we use an algorithm by Conradi and Driemel [62] to efficiently compute the non-empty cells without inspecting all pairs of segments.

FACT 59 ([62, Lemma 59]). *Given two c-packed curves π and σ in $\mathbb{R}^2$, a parameter $\delta \geq 0$, and let π' and σ' be their $\varepsilon\delta$-simplifications. In $\mathcal{O}((cn/\varepsilon)\log(cn/\varepsilon))$ time, one can find all pairs of segments $(u, v) \in \pi'$ and $(a, b) \in \sigma'$ such that the distance between (u, v) and (a, b) is at most δ.*

To construct the simplified freespace diagram efficiently, we first observe the following.

OBSERVATION 60. *If the segments $(u, v) \in \pi'$ and $(a, b) \in \sigma'$ are more than $(1 + 2\varepsilon)\delta$ apart, then π_{uv} and σ_{ab} are more than δ apart.*

The above observation enables us to determine whether the (π_{uv}, σ_{ab}) cells are empty by determining whether (u, v) and (a, b) are near.

4.4.3 Constructing the Simplified Freespace Diagram

Given two c-packed polygonal curves π and σ, we will use the results from previous subsections to construct the simplified freespace diagram using the steps below. In Lemma 57, we showed that if π_{uv} and σ_{ab} are simplified into the segments $(u, v) \in \pi'$ and $(a, b) \in \sigma'$, respectively, we can compute the (π_{uv}, σ_{ab}) cells in constant time. Such an aggregation of (π_{uv}, σ_{ab}) cells is an *aggregated non-empty cell*, and we treat them as a single cell for simplicity.

(1) Simplify π and σ into their $\varepsilon\delta$-simplifications π' and σ'.
(2) Find all pairs of nearby segments from π' and σ' that are at most $(1 + \hat{\varepsilon})\delta$ apart using Fact 59.
(3) For each pair of nearby segments $(u, v) \in \pi'$ and $(a, b) \in \sigma'$, compute the (π_{uv}, σ_{ab}) cells using Lemma 57.
(4) Sort all non-empty cells horizontally and vertically.
(5) Connect non-empty cells in a graph fashion such that a non-empty cell is connected to the first non-empty cells above, below, to the left, and to the right.

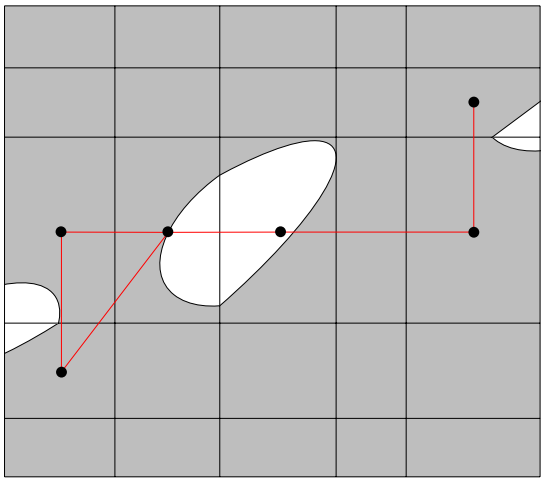

FIGURE 4.3: The non-empty cells are connected horizontally and vertically to skip empty cells.

Given two polygonal curves π and σ of complexity n, simplifying them (step 1) takes $\mathcal{O}(n)$ time. By Fact 59, step 2 takes $\mathcal{O}((cn/\varepsilon)\log(cn/\varepsilon))$ time. Computing a cell in $\mathcal{D}^s_{(1+\hat{\varepsilon})\delta}(\pi, \sigma)$ takes $\mathcal{O}(1)$ time by Lemma 57. The diagram $\mathcal{D}^s_{(1+\hat{\varepsilon})\delta}(\pi, \sigma)$ has at most $\mathcal{O}(cn/\varepsilon)$

non-empty cells, which takes $\mathcal{O}(cn/\varepsilon)$ time to compute in step 3; sorting them in step 4 takes $\mathcal{O}((cn/\varepsilon)\log(cn/\varepsilon))$ time. Connecting each cell to at most four other cells takes $\mathcal{O}(cn/\varepsilon)$ time in step 5. Putting this together, we obtain Lemma 61, and we summarise our result in Theorem 11.

LEMMA 61. *Let π and σ be two c-packed curves of complexity n. Let $\varepsilon > 0$ and $\delta > 0$ be two parameters, and let $\hat{\varepsilon} \leq 8\varepsilon$. One can construct and connect $\mathcal{O}(cn/\varepsilon)$ aggregated non-empty cells of the simplified freespace diagram $\mathcal{D}^s_{(1+\hat{\varepsilon})\delta}(\pi,\sigma)$ in $\mathcal{O}((cn/\varepsilon)\log(cn/\varepsilon))$ time such that*

$$\mathcal{F}_\delta(\pi,\sigma) \subseteq \mathcal{F}^s_{(1+\hat{\varepsilon})\delta}(\pi,\sigma) \subseteq \mathcal{F}_{(1+\hat{\varepsilon})\delta}(\pi,\sigma).$$

Given an aggregated non-empty cell C, one can access the first aggregated non-empty cells below, above, to the left, and to the right of C in $\mathcal{O}(1)$ time.

THEOREM 11. *Given a pair of trajectories, one can construct a simplified freespace diagram in $\mathcal{O}((cn/\varepsilon)\log(cn/\varepsilon))$ time, so that the simplified freespace has complexity $\mathcal{O}(cn/\varepsilon)$, it approximates the Fréchet distance to within a factor of $(1+\varepsilon)$, and it preserves the trajectory lengths of the original trajectory.*

4.5 Reference Trajectory is Vertex-to-Vertex

Throughout the rest of the chapter, we assume that the freespace diagram is the simplified freespace diagram $\mathcal{D}^s_{(1+\varepsilon)\delta}(\pi,\pi)$ in Lemma 61. Next, we use the algorithm of [95] to determine whether there is a solution to $SC(T, m, l, (1+\varepsilon)\delta)$ where π is a c-packed trajectory, and the reference subtrajectory π_{st} is vertex-to-vertex; that is, both s and t must be endpoints of some segment of π.

Three data structures are used in the vertex-to-vertex subtrajectory cluster algorithm of [95] — a directed graph, a range tree, and a link-cut tree. In Section 4.5.1, we outline the data structures. In Section 4.5.2, we show that the number of leaves per range tree is $\mathcal{O}(c/\varepsilon)$, and that the directed graph has complexity $\mathcal{O}((cn/\varepsilon)\log(c/\varepsilon))$. In Section 4.5.3, we show that the link-cut tree data structure can be used without modification.

4.5.1 The Directed Graph

In this section, we discuss the construction of a directed graph to store candidate xy-monotone paths, as proposed by [95]. They defined a specific type of xy-monotone path that exists only within a row or column, and showed that any general xy-monotone path π can be decomposed into a series of these so-called *basic xy-monotone paths* π', such that the y-span of π is a subset of the y-span of π'. These basic xy-monotone paths are stored in a directed graph, which can be queried efficiently to find feasible xy-monotone paths.

DEFINITION 62 ([95]). *A basic xy-monotone path is an xy-monotone path that is contained entirely within a single row or column of the freespace diagram, starting at a critical point on a vertical cell boundary and ending at a critical point on a horizontal cell boundary, or vice versa.*

[95, Lemma 16] showed that there is an xy-monotone path from a critical point a to a critical point b on the freespace diagram if and only if there is a sequence of basic xy-monotone paths between a and b. Their idea is to decompose an xy-monotone path into a path $p_1 p_2 \dots p_k$ such that $p_1 = a$, $p_k = b$, and the path from p_i to p_{i+1} is a basic xy-monotone path. One can first transform an xy-monotone path σ into a set of almost-basic xy-monotone paths $q_1 \dots q_k$ inductively: if q_i lies on a vertical (resp. horizontal) cell boundary, then q_{i+1} is the next intersection of σ with a horizontal (resp. vertical) boundary. One can then transform the path $q_1 \dots q_k$ into a series of basic xy-monotone paths as follows. If q_i lies on a vertical (resp. horizontal) cell boundary, then p_i is the critical point below (resp. left of) q_i.

FACT 63 ([95, Lemma 16]). *Given a pair of critical points a and b in the freespace diagram, there is an ab xy-monotone path if and only if there is a sequence of basic xy-monotone paths $p_1 \dots p_k$ such that p_i is a critical point for $1 \leq i \leq k$, $p_1 = a$, and $p_k = b$.*

With basic xy-monotone paths defined, we want to construct a graph to store all possible basic xy-monotone paths. We do so row-by-row and column-by-column. For an arbitrary row, where H is the top boundary, let p_i be the bottom-most critical point of the ith vertical cell boundary, where $0 \leq i \leq n + 1$. A brute-force approach is to connect every p_i to every

critical point q on H such that there is a p_iq basic xy-monotone path. However, the number of edges is cubic in this case.

Define q_i to be the rightmost critical point on the top boundary such that there is a p_iq_i basic xy-monotone path. A key observation is that if there is a p_iq_i xy-monotone path, then there is a p_iq xy-monotone path for every critical point q on H that lies to the right of p_i and to the left of q_i. Indeed, let r be the intersection of the p_iq_i xy-monotone path with the left vertical cell boundary of the cell containing r. Since the interior of a non-empty cell is convex, there is a p_ir xy-monotone path.

The above observation enables Gudmundsson and Wong to define a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to efficiently store all possible basic xy-monotone paths [95]. For an arbitrary row, they first construct a range tree RT storing the critical points on H with respect to their increasing x-coordinates. They then connect p_i to a node v in RT whenever there is a basic xy-monotone path from p_i to every critical point in the leaves of v (see Figure 4.4). Each column of the freespace diagram is processed analogously. As we will use the range tree extensively in the following section, we define the canonical subset to differentiate it from the canonical squares used in the previous section. Given a node v in a range tree, *the canonical subset* of v is the set of points stored in the leaves of v [15].

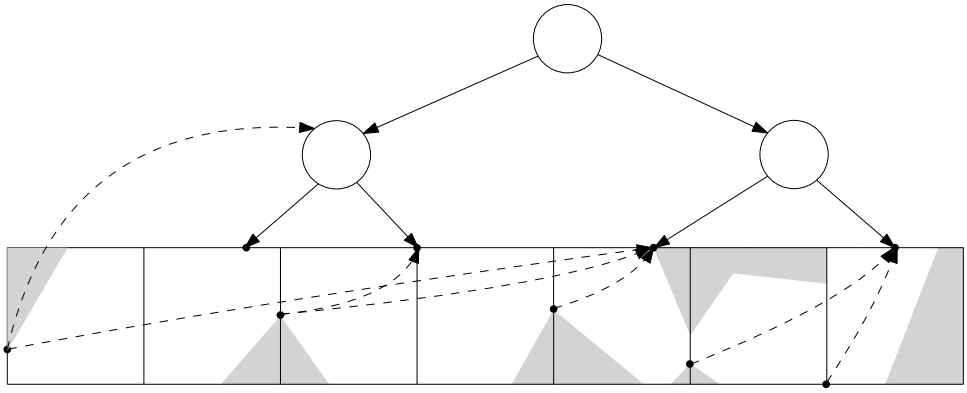

FIGURE 4.4: Part of the graph $\mathcal{G}$. We build a directed range tree from the critical points on the top boundary. A bottom critical point p on a vertical cell boundary is connected to a node v of the range tree whenever there is a basic xy-monotone path from p to every critical point in the canonical subset of v, illustrated with dashed lines.

[95] described an algorithm that finds q_i for each p_i. However, we cannot use their algorithm for two reasons. First, their algorithm only works in the very restrictive case where all non-empty cells have critical points on their top boundary, which is not true in the freespace diagram of two general curves. Second, their algorithm only works when all cells in a row are non-empty.

We now show how to find q_i for each p_i. Let r_i be the top critical point on a vertical cell boundary. For some p_i on the ith vertical cell boundary, we draw a horizontal line to the right from p_i. If this horizontal line is blocked first immediately before the jth vertical cell boundary and $y(p_i) \leq y(p_j)$ (resp. $y(p_i) \geq y(r_j)$), we say that p_i is blocked by p_j (resp. by r_j).

There are two things that can happen (see Figure 4.5). First, the line is blocked by some point b on the boundary between freespace and non-free space in the cell immediately to the left of r_j or p_j. The point b is either higher than r_j or lower than p_j. If b is below some p_j, then $q_i = q_j$ since there is a $p_i p_j q_j$ xy-monotone path. If b is above some r_j, then q_i is simply the rightmost critical point on H that lies to the left of r_j and to the right of p_i.

Second, the line from p_i is blocked at a point b by non-free space that completely separates two adjacent cells, or it is blocked by the right boundary of the freespace diagram. Suppose b lies between the jth and $(j-1)$th vertical cells; then q_i is simply the rightmost critical point on H that lies to the left of the jth vertical cell boundary and to the right of p_i.

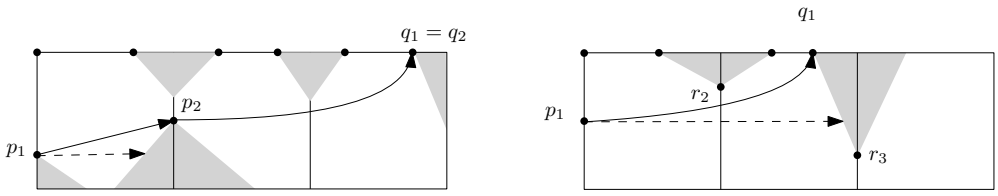

FIGURE 4.5: In the left figure, p_1 is blocked by p_2. In the right figure, p_1 is blocked by r_3.

Knowing the above, given p_i, if we can find the first p_j or r_j that blocks p_i, we can determine q_i. We use a binary search tree to store the top and bottom critical points of the vertical cell boundaries based on their increasing x-coordinates, where critical points on the same vertical cell boundary are stored in the same leaf. In addition, each node v stores the maximum and

minimum y-coordinates of the critical points in the canonical subset of v. One can query this binary search tree to find the first p_j or r_j that blocks p_i in $\mathcal{O}(\log n)$ time. In fact, we can do the same thing for any arbitrary point p in the freespace, as long as we know which cell p resides in.

OBSERVATION 64. *Given a set of n_k consecutive non-empty cells in the same row with vertical boundary indices $\{1, \ldots, n_k + 1\}$, one can preprocess them in $\mathcal{O}(n_k \log n_k)$ time such that one can find the first p_j or r_j that blocks p in $\mathcal{O}(\log n)$ time.*

With the above observation, and the dynamic programming algorithm described in [95, Lemma 20], we can preprocess the freespace diagram such that, given a free point p, we can find the rightmost critical point on the top boundary of the same row in $\mathcal{O}(\log n)$ time. This property is useful in Section 4.6, where the reference trajectory is no longer vertex-to-vertex and an xy-monotone path can start from the interior of a non-empty cell. We combine the above insights and observations in Lemma 65 below.

LEMMA 65. *Given a row of n_k non-empty cells in the freespace diagram and a point p in the freespace, one can preprocess these cells in $\mathcal{O}(n_k \log n_k)$ time, and find the rightmost critical point q on the top boundary such that there is a pq xy-monotone path in $\mathcal{O}(\log n_k)$ time.*

PROOF. Given a row of non-empty cells, we apply the algorithm below to each set of consecutive adjacent non-empty cells such that the boundary between two cells intersects the freespace. Let $x(p)$ and $y(p)$ be the x- and y-coordinates of a point p, respectively. Let the indices of the leftmost and rightmost vertical boundaries be l and r, respectively.

Preprocess the row using Observation 64 in $\mathcal{O}(n_k \log n_k)$ time. Iterate i from r to l, and for each p_i, use Observation 64 to find the leftmost p_j or r_j such that $y(p_i) \leq y(p_j)$ or $y(p_i) \geq y(r_j)$, respectively, in $\mathcal{O}(\log n_k)$ time. If p_j is leftmost, set $q_i = q_j$. If r_j is leftmost, use binary search to find the rightmost critical point q such that $x(p_i) \leq x(q) \leq x(r_j)$ in $\mathcal{O}(\log n_k)$ time. In total, this takes $\mathcal{O}(n_k \log n_k)$ time.

Then, given a free point p between the ith and $(i+1)$th cell boundaries, one can find the first p_j or r_j that blocks p, and repeat the above process to find the rightmost critical point q such that there is a pq xy-monotone path in $\mathcal{O}(\log n_k)$ time.

The correctness of this algorithm relies on two key facts. First, if p_j is the leftmost critical point such that p_i is blocked by p_j, then there is a p_ip_j xy-monotone path. Second, if r_j is the leftmost critical point that blocks p_i, then there is a p_iq xy-monotone path as long as q lies between p_i and r_j. Indeed, since the intersection of the freespace with a cell is convex, if p_i is not blocked by any critical point of a cell C, then p_i cannot be blocked by the interior of C. □

4.5.2 Using a Directed Graph to Store Candidate Monotone Paths

We next prove that the directed graph has a smaller size in the simplified freespace diagram. Specifically, we show that the range tree has at most $\mathcal{O}(c/\varepsilon)$ leaves, and to do so, it suffices to show that there exist at most $\mathcal{O}(c/\varepsilon)$ critical points on each horizontal or vertical boundary of the simplified freespace diagram.

LEMMA 66. *In the simplified freespace diagram $\mathcal{D}^s_{(1+\hat{\varepsilon})\delta}(\pi,\pi)$, let H be a horizontal (resp. vertical) strip that is at least $\varepsilon\delta$ wide in its y-span (resp. x-span). The intersection of H and the simplified freespace $\mathcal{F}^s_{(1+\hat{\varepsilon})\delta}(\pi,\pi)$ occurs in at most $\mathcal{O}(c/\varepsilon)$ aggregated cells.*

PROOF. Let T' be the $\varepsilon\delta$-simplification of π, and let π_{uv} simplify into the segment $(u,v) \in T'$. Let $u' \subseteq (u,v)$ be a small subsegment that is at least $\varepsilon\delta$ long. Let $S_{u'} = u' \oplus \mathbf{b}(0,(1+\hat{\varepsilon})\delta)$.

Using a similar construction and the arguments of [65, Lemma 4.4], one can prove that at most $\mathcal{O}(c/\varepsilon)$ segments in T' intersect $S_{u'}$. Based on the construction of the simplified freespace $\mathcal{F}^s_{(1+\hat{\varepsilon})\delta}(\pi,\pi)$, a point $(x,y) \in \mathcal{F}^s_{(1+\hat{\varepsilon})\delta}(\pi,\pi)$ is in the freespace if and only if

$$\|f_{\pi,\hat{\varepsilon}\delta}(\pi[x]) - f_{\pi,\hat{\varepsilon}\delta}(\pi[y])\| \leq (1+\hat{\varepsilon})\delta.$$

As such, at most $\mathcal{O}(c/\varepsilon)$ aggregated cells have simplified freespace intersecting H. □

Next, we bound the construction time and space complexity of the directed graph in [95].

LEMMA 67. *Given a c-packed trajectory π of complexity n, constructing $\mathcal{G}$ for the simplified freespace diagram $\mathcal{D}^s_{(1+\hat{\varepsilon})\delta}(\pi,\pi)$ takes $\mathcal{O}((cn/\varepsilon)\log(n/\varepsilon))$ time. The graph $\mathcal{G}$ has $\mathcal{O}(cn/\varepsilon)$ nodes and $\mathcal{O}((cn/\varepsilon)\log(c/\varepsilon))$ edges.*

PROOF. Let n_k be the number of non-empty aggregated cells in the jth row of $\mathcal{D}^s_{(1+\hat{\varepsilon})\delta}(\pi,\pi)$. Construction of the range tree for the top (resp. right) boundary of a row (resp. column) takes $\mathcal{O}(n_k \log n_k)$ time [15]. For all p_i, finding q_i takes $\mathcal{O}(n_k \log n_k)$ time, and recall that there are $\mathcal{O}(cn/\varepsilon)$ critical points in $\mathcal{D}^s_{(1+\hat{\varepsilon})\delta}(\pi,\pi)$. The total construction time is as follows.

$$\sum_{j=0}^{n+1} n_k \log n_k \;\leq\; \log\Big(\frac{cn}{\varepsilon}\Big) \sum_{j=0}^{n+1} n_k \;=\; \log\Big(\frac{cn}{\varepsilon}\Big)\, \mathcal{O}\Big(\frac{cn}{\varepsilon}\Big) \;\in\; \mathcal{O}\Big(\Big(\frac{cn}{\varepsilon}\Big) \log\Big(\frac{n}{\varepsilon}\Big)\Big)\,.$$

By Corollary 58, the simplified freespace diagram has $\mathcal{O}(cn/\varepsilon)$ non-empty aggregated cells, and therefore $\mathcal{G}$ has $\mathcal{O}(cn/\varepsilon)$ nodes. In a range tree, given a continuous interval $[q_k, q_i]$, one can find $\mathcal{O}(\log n)$ nodes such that these nodes include $[q_k, q_i]$ in their canonical subset, where n is the total number of items in the leaves [15]. There are at most $\mathcal{O}(c/\varepsilon)$ nodes on a horizontal or vertical boundary by Lemma 66, and each critical point p_i on a vertical (resp. horizontal) cell boundary connects to $\mathcal{O}(\log(c/\varepsilon))$ nodes. Therefore, the total number of edges is $\mathcal{O}((cn/\varepsilon)\log(c/\varepsilon))$. □

4.5.3 Storing and Reusing Pre-computed Paths

A link-cut tree [132] maintains a forest that allows link and cut operations on subtrees in $\mathcal{O}(\log n)$ amortised time. In addition, a link-cut tree allows finding the root of a node in $\mathcal{O}(\log n)$ amortised time. The algorithm of [95] uses a link-cut tree to store and re-use xy-monotone paths. When a sweepline, either l_s or l_t, stops at a new critical point p, they need only add p to the existing link-cut tree maintained from previous instances, rather than recomputing the xy-monotone paths.

With the graph $\mathcal{G}$ defined, we can analyse the total running time of the algorithm of [95] on the simplified freespace diagram. The key observation regarding the running time is that, in their algorithm, if an edge leads to a dead end, it is marked and will not be used in future searches. Furthermore, inserting or removing an edge takes $\mathcal{O}(\log n)$ amortised time in a link-cut tree.

THEOREM 12. *There is an $\mathcal{O}(nm\log(c/\varepsilon)\log(n/\varepsilon))$ time algorithm that solves $SC(T, m, l, (1+\varepsilon)\delta)$ in the case that the reference trajectory is vertex-to-vertex.*

PROOF. Construction of the simplified freespace diagram takes $\mathcal{O}((cn/\varepsilon)\log(cn/\varepsilon))$ time by Theorem 11. Construction of $\mathcal{G}$ takes $\mathcal{O}((cn/\varepsilon)\log(n/\varepsilon))$ time by Lemma 67. In addition to the critical points in $\mathcal{G}$, [95] showed that we need to consider $\mathcal{O}(mn)$ greedy critical points. In total, the graph $\mathcal{G}$ has at most $\mathcal{O}(nm\log(c/\varepsilon))$ edges, because an edge is added to and removed from the link-cut tree at most once, and adding or removing an edge from the link-cut tree takes $\mathcal{O}(\log(n/\varepsilon))$ time since the maximum number of nodes in the link-cut tree is upper-bounded by the number of nodes in $\mathcal{G}$. Therefore, maintaining the link-cut tree takes $\mathcal{O}(nm\log(c/\varepsilon)\log(n/\varepsilon))$ time. □

4.6 Reference Trajectory is Arbitrary

Our results in this section rely heavily on the work of [95]. For brevity, we only highlight the important parts of their algorithm and the analysis of our improvements.

When the reference trajectory is arbitrary, an xy-monotone path can start and finish at arbitrary positions in non-empty cells. Therefore, in addition to the critical points in the freespace diagram and the greedy critical points, [95] defined three new types of *internal critical points* [95, Definition 25]. An internal critical point must lie in the interior of a non-empty cell and lie on the boundary of the freespace. They made the following distinctions (see Figure 4.6).

(1) **End-of-cell critical point:** the leftmost and rightmost points in the freespace of a non-empty cell.

(2) **Propagated critical point:** a point on the boundary of the freespace that shares a y-coordinate with a critical point.

(3) **l-apart critical points:** two points on the boundary of the freespace that are a horizontal distance of l apart.

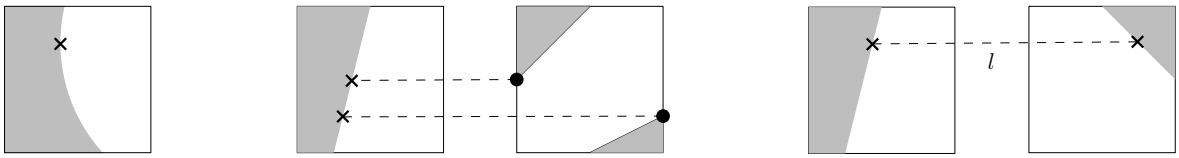

FIGURE 4.6: The three types of internal critical points are illustrated using a cross in the left, middle, and right figures, respectively. From left to right, they are the end-of-cell critical points (left), the propagated critical points (middle), and the l-apart critical points (right).

There could be an infinite number of l-apart critical points in a pair of non-empty cells. One can detect that two cells generate infinitely many l-apart critical points by comparing the quadratic equations of their interiors (see Figure 4.7). To cope with this case, one can move the endpoint of one of the segments by a minuscule amount, say $\delta = \varepsilon^2 d$. Therefore, for the rest of the chapter, we assume that there are at most a constant number of l-apart critical points per pair of cells.

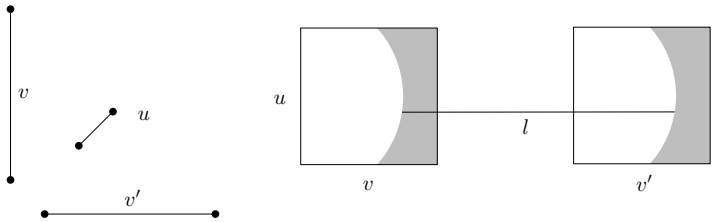

FIGURE 4.7: The placement of segments u, v, and v' and their respective (u, v) and (u, v') cells. There are infinitely many l-apart critical points when the boundaries of the freespace in two cells can be expressed by the same equation and are exactly l apart.

We first bound the number of internal critical points and the time it takes to compute them. One can compute the end-of-cell and l-apart critical points in linear time with respect to the number of non-empty cells, since there are at most a constant number of them per pair of cells. In Lemma 66, we showed that in a narrow horizontal strip, only a small number of

cells intersect the freespace. An output-sensitive query algorithm is therefore efficient for finding the non-empty cells that a critical point p propagates to. Accordingly, we can use an interval tree [15] to store the y-spans of all non-empty cells in a row, and query the intervals intersecting $y(p)$ in logarithmic time. We formalise the above arguments in Lemma 68 below.

LEMMA 68. *Assuming that there is a constant number of l-apart critical points per pair of cells, it takes $\mathcal{O}(cn\log(n/\varepsilon) + c^2n/\varepsilon^2)$ time to compute $\mathcal{O}(c^2n/\varepsilon^2)$ internal critical points in the simplified freespace diagram $\mathcal{D}^s_{(1+\hat{\varepsilon})\delta}(\pi,\pi)$.*

PROOF. There are $\mathcal{O}(cn/\varepsilon)$ non-empty aggregated cells in $\mathcal{D}^s_{(1+\hat{\varepsilon})\delta}(\pi,\pi)$, or non-empty cells for short, and $\mathcal{O}(cn/\varepsilon)$ end-of-cell critical points in $\mathcal{D}^s_{(1+\hat{\varepsilon})\delta}(\pi,\pi)$. Each critical point propagates $\mathcal{O}(c/\varepsilon)$ times by Lemma 66, and therefore there are $\mathcal{O}(c^2n/\varepsilon^2)$ propagated critical points. We can charge a cell with a constant number of l-apart critical points. Therefore, there are at most $\mathcal{O}(cn/\varepsilon)$ l-apart critical points. In total, there are $\mathcal{O}(c^2n/\varepsilon^2)$ internal critical points.

One can compute the end-of-cell critical points by iterating through the freespace diagram in $\mathcal{O}(cn/\varepsilon)$ time. To compute the l-apart critical points, we can start from the first non-empty cell C in a row and find the first cell that is l apart from C, and solve a constant number of quadratic equations. We can then slide this l-apart line and do the same for all pairs of cells that are l apart in all rows, in $\mathcal{O}(cn/\varepsilon)$ time in total.

To compute the propagated critical points, we construct an interval tree [15] for each row in $\mathcal{D}^s_{(1+\hat{\varepsilon})\delta}(\pi,\pi)$ to store the maximum and minimum y-coordinates of the freespace in the non-empty cells. Let n_i be the number of non-empty cells in the ith row. We can sum the construction time of the interval trees:

$$\sum_{i=1}^{n} n_i \log n_i \;\leq\; \frac{cn}{\varepsilon}\log\Big(\frac{cn}{\varepsilon}\Big) \;\in\; \mathcal{O}\Big(\Big(\frac{cn}{\varepsilon}\Big)\log\Big(\frac{cn}{\varepsilon}\Big)\Big).$$

Given a critical point p in the ith row, one can query the interval tree in $\mathcal{O}(\log n_i + c/\varepsilon) \in \mathcal{O}(\log n + c/\varepsilon)$ time to compute the propagated critical points from p using Lemma 66

and [15]. With $\mathcal{O}(cn/\varepsilon)$ critical points, computing the propagated critical points takes $\mathcal{O}(cn \log n/\varepsilon + c^2 n/\varepsilon^2)$ time.

□

With the additional internal critical points, the number of reference trajectories and the number of greedy critical points increase. We can use the algorithm from the previous section and obtain the following result.

LEMMA 69. *There is an $\mathcal{O}((c^2mn/\varepsilon^2)\log(c/\varepsilon)\log(n/\varepsilon))$–time algorithm that solves $SC(T, m, l, (1+\varepsilon)\delta)$ in the case where the reference trajectory is arbitrary.*

PROOF. Construction of the simplified freespace diagram takes $\mathcal{O}(cn/\varepsilon + cn/\varepsilon^3)$ time by Theorem 11. Construction of $\mathcal{G}$ takes $\mathcal{O}((cn/\varepsilon)\log(n/\varepsilon))$ time by Lemma 67. The total number of critical points is upper-bounded by the number of propagated critical points, which is $\mathcal{O}(c^2n/\varepsilon^2)$. The total number of greedy critical points is $\mathcal{O}((c^2n/\varepsilon^2)\cdot m)$. Sorting the critical points takes $\mathcal{O}((c^2mn/\varepsilon^2)\log(n/\varepsilon))$ time. For every critical point p, computing the rightmost point q such that there is a pq xy-monotone path takes $\mathcal{O}((c^2nm/\varepsilon^2)\log n)$ time by Lemma 65. The graph $\mathcal{G}$ has at most $\mathcal{O}((c^2mn/\varepsilon^2)\log(c/\varepsilon))$ edges after adding the greedy critical points and the internal critical points. [95] showed that an edge is added to and removed from the link-cut tree at most once, and adding or removing an edge from the link-cut tree takes $\mathcal{O}(\log(n/\varepsilon))$ time, since the maximum number of nodes in the link-cut tree is upper-bounded by the number of nodes in $\mathcal{G}$. Therefore, maintaining the link-cut tree takes $\mathcal{O}((c^2mn/\varepsilon^2)\log(c/\varepsilon)\log(n/\varepsilon))$ time. □

4.6.1 Improve Further with an Interval Management Data Structure

The bottleneck in Lemma 69 is operating on the outgoing edges of the $\mathcal{O}(c^2mn/\varepsilon^2)$ greedy critical points, which are generated from the $\mathcal{O}(c^2n/\varepsilon^2)$ propagated critical points. To avoid computing the greedy critical points, [95] used a dynamic monotonic interval data structure [83] to store overlapping monotonic intervals that represent the y-spans of xy-monotone paths between l_s and l_t. Instead of searching for a set of xy-monotone paths

between each window greedily, they showed that one can update and query the interval data structure to retrieve $m-1$ non-overlapping intervals, all in $\mathcal{O}(\log n)$ amortised time.

THEOREM 13. *There is an $\mathcal{O}((c^2n/\varepsilon^2)\log(c/\varepsilon)\log(n/\varepsilon))$ time algorithm that solves $SC(T, m, l, (1+\varepsilon)\delta)$ in the case that the reference trajectory is arbitrary.*

PROOF. Constructing the simplified freespace diagram takes $\mathcal{O}((cn/\varepsilon)\log(cn/\varepsilon))$ time by Theorem 11. Computing and sorting the internal critical points takes $\mathcal{O}((c^2n/\varepsilon^2)\log(n/\varepsilon))$ time by Lemma 68. There are $\mathcal{O}((c^2n/\varepsilon^2)\log(c/\varepsilon))$ edges in total by Lemma 66, and each edge takes $\mathcal{O}(\log(n/\varepsilon))$ time to insert or remove, since there are at most $\mathcal{O}(c^2n/\varepsilon^2)$ nodes in the link-cut tree. In total, we spend $\mathcal{O}((c^2n/\varepsilon^2)\log(c/\varepsilon)\log(n/\varepsilon))$ time to maintain the edges in $\mathcal{G}$.

Each internal critical point is treated as an event, and maintaining the interval data structure takes $\mathcal{O}(\log n)$ amortised time per event point (see [95, Theorem 2]), and thus $\mathcal{O}((c^2n/\varepsilon^2)\log n)$ in total. The overall complexity is dominated by maintaining the edges. □

4.7 Conclusion and future work

We presented an algorithm that solves the subtrajectory cluster problem on c-packed trajectories π with an approximation factor on the Fréchet distance, achieving an $\mathcal{O}((c^2n/\varepsilon^2)\log(c/\varepsilon)\log(n/\varepsilon))$ time complexity. Our algorithm builds upon the near-optimal algorithm proposed by [95], but with significant improvements. By carefully analysing the structural properties of c-packed trajectories, we have shown that important parameters, such as the number of propagated critical points, are significantly lower than the theoretical $\mathcal{O}(n)$ upper bound for realistic trajectories. As a result, our algorithm improves upon the near-optimal algorithm by replacing a factor of n^2 with c^2/ε^2, leading to more efficient subtrajectory clustering on realistic trajectories.

As for future work, it would be interesting to see if we can generalise our result to less restrictive types of curves such as λ-low-density curves [16]. [65] proved that a c-packed

curve is $\mathcal{O}(c)$-low-density, but an $\mathcal{O}(1)$-low-density curve can in the worst case be $\mathcal{O}(n)$-packed. [65] showed that two $\mathcal{O}(1)$-low-density curves produce a freespace diagram with a subquadratic number of cells. However, the key Lemma 66 that enabled our time-complexity improvement does not trivially generalise to low-density curves. It would be interesting to see whether a similar observation can be made for λ-low-density curves.

CHAPTER 5

Spanner for the $0/1/\infty$ Weighted Region Problem

5.1 Introduction

The Weighted Region Problem (WRP) is a generalisation of the shortest path problem, considering a planar subdivision E where each face has a non-negative weight associated with it. A path σ in E can be partitioned into a set of subpaths $\{\sigma_1, \ldots, \sigma_k\}$ based on its intersection with faces in E, where a subpath σ_i starts at the point s_i and ends at the point t_i. Both s_i and t_i must lie on the boundary of the same face F_i. The weight of the subpath σ_i is the Euclidean length of σ_i times the weight assigned to F_i. The total weight of a path is the sum of the weights of its subpaths. The goal of the WRP is to find the weighted shortest path from a source point s to a target point t. When the weights are in the set $\{0, 1, \infty\}$, this problem is referred to as the $0/1/\infty$ Weighted Region Problem [84].

Researchers have conjectured that the WRP is difficult to solve [84], and recent studies confirm this conjecture — the Weighted Region Problem is unsolvable in the algebraic computation model over the rational numbers. De Carufel et al. [47] demonstrated that the WRP cannot be solved exactly even with only three different weights. De Berg et al. [17] confirmed its unsolvability with just two different weights. Mitchell and Papadimitriou [125] illustrated that in two dimensions, a weighted shortest path can intersect at least $\Omega(n^2)$ boundaries even when the regions are convex.

Due to the difficulty of solving the WRP exactly, approximation algorithms have been considered. A common approach is to discretise the problem space, either by assuming the space is a tessellation of convex polygons with exactly one associated weight [21], or by

placing Steiner (sample) points on the boundaries of the regions [4, 5, 58, 119, 135]. In these approaches, the number of sample points depends not only on the complexity of the regions but also on geometric parameters such as the maximum integer coordinate of any vertex and the ratio r_w of the maximum weight over the minimum weight. As r_w increases, so does the number of required sample points. As a result, the weights are required to be strictly positive.

5.1.1 Related work

Our work is closely related to the data structure and algorithm by Gewali et al. [84] to solve the $0/1/\infty$ weighted region problem. Their algorithm takes a polygonal domain with N vertices as input and constructs a *critical graph* $\mathcal{G}^* = (\mathcal{V}^*, \mathcal{E}^*)$ (a type of visibility graph) with $\mathcal{O}(N^2)$ edges. Dijkstra's shortest path algorithm can be used on $\mathcal{G}^*$ to compute a weighted shortest path between any pair of vertices in $\mathcal{O}(N^2)$ time.

In $0/1/\infty$ weighted regions, a weighted shortest path P^* avoids obstacles and traverses the 0-regions freely, while minimising its length in the plane (1-region). Consider two (closed) regions A and B, each either a 0-region or an obstacle. The key observation in [84] is that an edge in P^* connecting A and B must be *locally optimal* (see Fact 82). For example, an edge (a, b) connecting two convex 0-regions A and B must be perpendicular to the tangent touching $a \in A$ and the tangent touching $b \in B$. Gewali et al. [84] showed that $\mathcal{G}^*$ contains all such locally optimal edges, which implies that $\mathcal{G}^*$ must contain the optimal path between any pair of vertices in $\mathcal{G}^*$.

5.1.2 Our Contribution

In this chapter, we build on the work by Gewali et al. [84] with a focus on the 0-regions as they are not handled well by existing approximation schemes (using sample points or tessellation). In Section 5.2, we consider the $0/1$ weighted region problem where the 0-regions are convex but not necessarily polygonal.

PROBLEM 7. *In the planar subdivision induced by the plane with weight* 1 *and a set* $\mathcal{Z}$ *of non-overlapping convex zero-cost regions (0-regions) with weight* 0*, given an approximation*

error $0 < \varepsilon < 1$*, find a* $(1+\varepsilon)$*-approximate weighted shortest path from an arbitrary point* s *to an arbitrary point* t*.*

The high-level idea is that, in order to obtain $(1+\varepsilon)$-approximate shortest paths, we place $\mathcal{O}(1/\varepsilon)$ sample points on the boundary of each 0-region; the number of sample points is independent of other parameters. Using these sample points, we construct a Θ-graph and $\mathcal{O}(1/\varepsilon)$ trapezoidal maps, which are part of our data structure $\mathcal{B}$. The trapezoidal maps ensure the existence of good paths between 0-regions that are close[1] to each other, while the Θ-graph ensures the same for 0-regions that are far from each other. To the best of our knowledge, our algorithm is the first near-linear time $(1+\varepsilon)$-approximation algorithm that finds an approximate weighted shortest path in a $0/1$ weighted region. Our algorithm is near-optimal, as a weighted shortest path can have $\Omega(n+N)$ complexity.

THEOREM 14. *Consider a planar subdivision induced by a plane with weight 1, containing a set* $\mathcal{Z}$ *of non-overlapping convex 0-regions with weight 0. Let* $|\mathcal{Z}| = n$ *and* N *denote the total number of vertices in* $\mathcal{Z}$*. For any approximation factor* $0 < \varepsilon < 1$*, a data structure* $\mathcal{B}$ *can be constructed over* $\mathcal{Z}$ *in* $\mathcal{O}(N + (n/\varepsilon^2)(\log(n/\varepsilon) + \log N))$ *expected time, with a total size of* $\mathcal{O}(N + n/\varepsilon^2)$*. When queried with points* s *and* t*,* $\mathcal{B}$ *can return a weighted path* P *from* s *to* t *in* $\mathcal{O}(N + n/\varepsilon^2 + (n/\varepsilon)\log(n/\varepsilon) + (\log N)/\varepsilon)$ *time, satisfying* $\mathbf{w}(P) \leq (1+\varepsilon) \cdot \mathbf{w}(P^*)$*, where* P^* *is the optimal weighted shortest path from* s *to* t*.*

To use our algorithm on an application, we prove the above theorem in a more general setting, where the 0-regions are non-polygonal. In Section 5.3, we use our algorithm to approximate the partial weak Fréchet similarity of two polygonal curves. This problem was first studied by Buchin et al. [31], and they presented a cubic time algorithm. De Carufel et al. [48] later transformed the problem into a weighted shortest path problem amidst $0/1$-regions. Using Theorem 14, our algorithm is the first near-quadratic time $(1+\varepsilon)$-approximation algorithm for computing the partial weak Fréchet similarity between a pair of polygonal curves.

Buchin et al. [32] showed that there is no strongly subquadratic time algorithm for approximating the weak Fréchet distance within a factor less than 3 unless the strong exponential-time

[1] A precise definition is provided in Lemma 75 and 77, Section 5.2.

hypothesis fails. Approximating the partial weak Fréchet similarity is at least as hard as approximating the weak Fréchet distance. As a result, it is unlikely that a subquadratic time algorithm exists, and our algorithm is near-optimal in time complexity.

THEOREM 15. *One can approximate the partial weak Fréchet similarity of two curves with respect to the L_2 metric within a factor of $(\sqrt{2}+\varepsilon)$ in $\mathcal{O}((n^2/\varepsilon^2)\log(n/\varepsilon))$ expected time.*

In Section 5.4, we generalise our data structure to also allow convex obstacles that cannot be traversed, i.e., obstacles of weight ∞. By introducing additional sample points, we show that if we need to take a detour from a sample point a to a sample point b, there exists a set D (a detour) of edges in $\mathcal{B}$ such that the total length of D approximates the distance $\mathbf{d}(a,b)$ within a factor of $1+\varepsilon$. In the special case that the 0-regions and obstacles are polygonal, $\mathcal{B}$ is a $(1+\varepsilon)$-spanner of the input vertices. To the best of our knowledge, our algorithm is the first near-linear time $(1+\varepsilon)$-approximation algorithm for the weighted shortest path in a $0/1/\infty$ weighted region.

THEOREM 16. *Consider a planar subdivision induced by a plane with a weight of 1, consisting of two sets of convex and non-overlapping regions: 0-regions $\mathcal{Z}$ with a weight of 0, and obstacles $\mathcal{O}$ with a weight of ∞. Let $n=|\mathcal{Z}|+|\mathcal{O}|$ and let N denote the total number of vertices in $\mathcal{Z}\cup\mathcal{O}$. For any approximation factor $0<\varepsilon<1$, a data structure $\mathcal{B}$ can be constructed over $\mathcal{Z}\cup\mathcal{O}$ in $\mathcal{O}(N+(n/\varepsilon^3)(\log(n/\varepsilon)+\log N))$ expected time, with a total size of $\mathcal{O}(N+n/\varepsilon^3)$. When queried with arbitrary points s and t, $\mathcal{B}$ returns a weighted path P from s to t in $\mathcal{O}(N+n/\varepsilon^3+(n/\varepsilon^2)\log(n/\varepsilon)+(\log N)/\varepsilon)$ time, ensuring that $\mathbf{w}(P)\leq(1+\varepsilon)\cdot\mathbf{w}(P^*)$, where P^* is the optimal weighted shortest path from s to t.*

5.2 Shortest path amidst 0-regions

The exact version of Problem 7 has a brute-force solution. Given two 0-regions A and B, where s and t are considered 0-regions with no interior, let $\mathbf{d}(A,B)$ be the Euclidean distance between A and B. Let $\mathcal{G}_c=(\mathcal{V},\mathcal{E}_c)$ be a complete graph, where $\mathcal{V}=\mathcal{Z}\cup\{s,t\}$. For each edge $(A,B)\in\mathcal{E}_c$ for all pairs of $A\in\mathcal{V}_c$ and $B\in\mathcal{V}_c$, set the weight $\mathbf{w}(A,B)=\mathbf{d}(A,B)$.

Then, finding the optimal path P^* from s to t is equivalent to finding a weighted shortest path P^* in $\mathcal{G}_c$, which can be solved using Dijkstra's shortest path algorithm. However, the total number of edges required is at least $\Omega(n^2)$.

To compute an approximate solution, the goal is to construct an undirected weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with a near-linear number of edges, such that there exists a path P in $\mathcal{G}$ with $\mathbf{w}(P) \leq (1 + \varepsilon) \cdot \mathbf{w}(P^*)$.

To this end, we will use two data structures: trapezoidal maps and Θ-graphs. Both data structures are used to determine which pairs of 0-regions are connected. The trapezoidal maps will ensure that there exist good paths between 0-regions that are close to each other, while the Θ-graph ensures the same for 0-regions that are far from each other.

5.2.1 Construction of the data structure

Recall that the 0-regions are convex. In order to define our data structure, we first define a set of directions. Let $\theta < \pi/6$ be a fixed positive real number. Let $r(k\theta)$ be the direction with a counter-clockwise angle of $k\theta$ with the positive x-axis, where k is a non-negative integer. Let $r(p, k\theta)$ be the ray originating from the point p with a counter-clockwise angle of $k\theta$ with the positive x-axis. For simplicity, we write $r(k) = r(k\theta)$ and $r(p, k) = r(p, k\theta)$. To simplify the discussion, we will assume that $(\pi/2)/\theta \in \mathbb{Z}$. This assumption guarantees that if $r(k)$ exists, then the directions $r(k\theta + \pi/2)$, $r(k\theta + \pi)$, and $r(k\theta - \pi/2)$ also exists.

For a 0-region A, we define a set $\mathcal{SP}(A)$ of sample points on the boundary of A. Let $sp(A, k\theta) = sp(A, k)$ be a sample point on the boundary ∂A of A such that $sp(A, k)$ is extreme in the direction $r(k)$ (see Figure 5.1). When the geometric region A is clear from context, we write $sp(k)$ instead of $sp(A, k)$.

Each 0-region A is considered an open set, i.e., if $p \in \partial A$, then $p \notin A$. Let $\partial A(a, a')$ define the subset of ∂A traversed from point a to point a' in counter-clockwise order, where $a, a' \in \partial A$. We say a line l *overlaps* A if l and A intersect at more than one point. We say two regions, A and B, are non-overlapping if their interiors do not intersect.

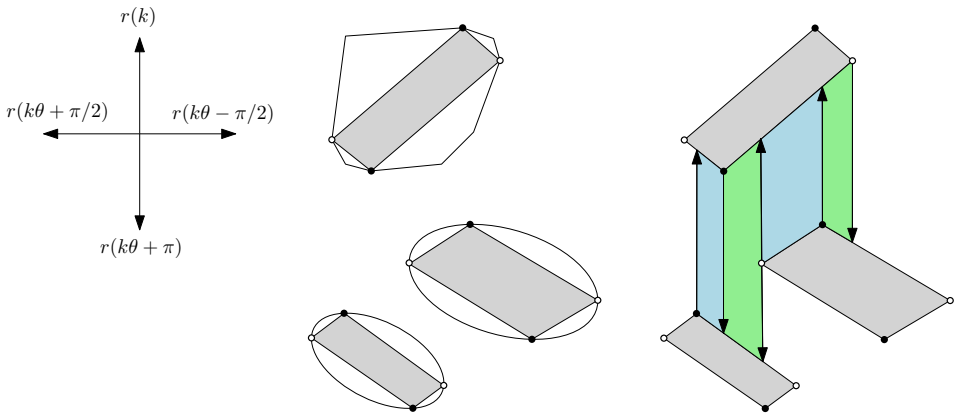

FIGURE 5.1: The sample points that are extreme in the directions of $r(k)$ and $r(k\theta+\pi)$ are marked with black dots. The sample points that are extreme in the directions of $r(k\theta\pm\pi/2)$ are marked with circles. Using these sample points, we generate a simplified polygon and construct $\mathcal{M}(k)$; the blue and green regions are examples of faces in $\mathcal{M}(k)$.

A point $p\in\partial A$ may be an extreme point for more than one direction, in which case we call p a vertex of A. There may be more than one extreme point on A for a single direction. If p is the extreme point on A for consecutive directions $\{r(k),r(k+1),...,r(k+m)\}$, we say $p=sp(k)$, $p=sp(k+1)$, ..., and $p=sp(k+m)$ simultaneously. If there is more than one extreme point for a single direction $r(k)$, these extreme points must lie on one or more consecutive segments on ∂A, and only the furthest points a and b are $sp(k)$. The sample points on the boundary of a convex region can be computed by traversing the boundary.

OBSERVATION 70. *Given n convex regions with N vertices in total, there are $\mathcal{O}(n/\theta)$ sample points, and it takes $\mathcal{O}(N+n/\theta)$ time to compute them.*

Using the sample points on a 0-region A, we can generate a simplified 0-region (a convex polygon) $\texttt{simpl}(A)$ by connecting adjacent sample points of every 0-region (see Figure 5.1). Using the set $\texttt{simpl}(\mathcal{Z})$ of simplified 0-regions, we will generate a set of trapezoidal maps, and we say $\texttt{simpl}(A)$ and $\texttt{simpl}(B)$ are adjacent if they are both adjacent to the same face in a trapezoidal map. We construct the query data structure $\mathcal{B}$ using Algorithm 1.

Algorithm 1 Construct $\mathcal{B}$ with 0-regions

This algorithm takes as input a set $\mathcal{Z}$ of non-overlapping and convex 0-regions, and constructs a data structure $\mathcal{B}$. The undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is initially empty.

(1) Compute the sample points $\mathcal{SP}(\mathcal{Z})$, and add $\mathcal{SP}(\mathcal{Z})$ to $\mathcal{V}$.
(2) Pick an arbitrary sample point as the anchor $ak(A)$ for every $A \in \mathcal{Z}$. For each $p \in \mathcal{SP}(A)$, add $e = (p, ak(A))$ to $\mathcal{E}$, and set $\mathbf{w}(e) = 0$.
(3) For each direction $r(k)$, generate a trapezoidal map $\mathcal{M}(k)$ using $\texttt{simpl}(\mathcal{Z})$, and do the following for each $\mathcal{M}(k)$ (see [15, Theorem 6.3 and 6.8] for trapezoidal map construction).
   - For each face $F \in \mathcal{M}(k)$ adjacent to A and B, $A \neq B$, add $e = (ak(A), ak(B))$ to $\mathcal{E}$, and set $\mathbf{w}(e) = \mathbf{d}(A, B)$.
(4) With $\mathcal{V}$ as the input, generate a Θ-graph $\mathcal{G}_\Theta = (\mathcal{V}, \mathcal{E}_\Theta)$.
   - For each edge $(p, q) \in \mathcal{E}_\Theta$, if p and q do not belong to the same 0-region, add $e = (p, q)$ to $\mathcal{E}$, and set $\mathbf{w}(e) = \|pq\|$.
(5) Return $\mathcal{B} = \{\mathcal{M}(k) \mid \forall k \in [0, 2\pi/\theta), k \in \mathbb{Z}\} \cup \{\mathcal{G}, \mathcal{G}_\Theta\}$ as the data structure.

5.2.1.1 Analysis

We bound the size of the data structure $\mathcal{B}$ and its construction time. In Step 1, by Observation 70, it takes $\mathcal{O}(N + n/\theta)$ time to compute all $\mathcal{O}(n/\theta)$ sample points. Step 2 takes $\mathcal{O}(n/\theta)$ time to connect every sample point to its respective anchor. In Step 3, it takes $\mathcal{O}((n/\theta)\log(n/\theta))$ expected time [15] to build the trapezoidal map $\mathcal{M}(k)$ over $\texttt{simpl}(\mathcal{Z})$ for each of the $\mathcal{O}(1/\theta)$ directions, resulting in a total of $\mathcal{O}((n/\theta^2)\log(n/\theta))$ expected time. In every $\mathcal{M}(k)$, we construct at most three edges in $\mathcal{E}$ per sample point $a = sp(A, k\theta + \pi/2)$, if $r(a, k)$ and $r(a, k\theta + \pi)$ hit different 0-regions. Therefore $|\mathcal{M}(k)| \in O(n/\theta)$, and in total $\sum_k \mathcal{M}(k) = O(n/\theta^2)$. Once $\mathcal{M}(k)$ is constructed for all k, using the algorithm by Edelsbrunner [69], it takes $\mathcal{O}(\log N)$ time to compute the shortest distance between two convex 0-regions for each of the $\mathcal{O}(n/\theta^2)$ faces. In Step 4, it takes $\mathcal{O}((n/\theta^2)\log(n/\theta))$ time to construct a Θ-graph using $\mathcal{O}(n/\theta)$ sample points, and $\mathcal{O}(1/\theta)$ cones [126]. Step 4a takes $\mathcal{O}(n/\theta^2)$ time, since $|\mathcal{E}_\Theta| = O(n/\theta^2)$.

Later, we will show that for an approximation factor ε, we have that $\theta \in \mathcal{O}(\varepsilon)$. The resulting complexities are summarised below.

LEMMA 71. *Given an approximation factor $0 < \varepsilon < 1$, and n non-overlapping convex 0-regions with total complexity N, one can build the data structure $\mathcal{B}$ in $\mathcal{O}(N + (n/\varepsilon^2)(\log(n/\varepsilon) + \log N))$ time, and the total size of $\mathcal{B}$ is $\mathcal{O}(N + n/\varepsilon^2)$.*

To the best of our knowledge, constructing a trapezoidal map using arcs has not been studied before. Therefore, in Algorithm 1, we used the well-studied result on the trapezoidal map construction using segments.

For each direction $r(k)$, we built a trapezoidal map $\mathcal{M}(k)$ using the simplified 0-regions $\texttt{simpl}(\mathcal{Z})$, see Step 3 of Algorithm 1. Consider the union of $\mathcal{Z}$ and $\texttt{simpl}(\mathcal{Z})$. We argue that constructing a trapezoidal map using the simplified regions captures the structure of the trapezoidal map if we had used the subboundaries of the original regions instead. To do this, we show that if a ray $r = r(p, k)$ originating from a sample point p on 0-region A hits a subboundary $\partial B(u, v)$ first, then r must hit the segment uv next.

First, the 0-regions are non-overlapping, so A cannot intersect the region enclosed by $\partial B(u, v)$ and uv. Second, without loss of generality, let $r(p, k)$ coincide with the y-axis. By construction, if $r(k)$ exists, then $r(k\theta + \pi/2)$ and $r(k\theta - \pi/2)$ exist, which implies that there exist two sample points: one is the leftmost point of A, and one is the rightmost. Therefore, the horizontal span of $\partial B(u, v)$ and uv are equal. A ray shooting in the direction $r(k)$ cannot hit $\partial B(u, v)$ first but misses uv.

OBSERVATION 72. *If a ray $r = r(p, k)$ hits a subboundary $\partial B(u, v)$ first, then r must hit uv next.*

With the data structure defined, we analyse the quality of the path we obtain from $\mathcal{B}$.

5.2.2 Trapezoidal map

Before arguing that there exists a good path using the edges constructed, we start with an observation about the pair of points realising the shortest distance between two 0-regions. For a convex region A and a point $p \in \partial A$, there exists at least one *supporting line* $l = l_t(A, p)$

going through p such that A lies entirely in one of the two halfplanes determined by l [137]. Let $p \in \partial A$, and $q \in \partial B$. We can observe that if pq realises $\mathbf{d}(A, B)$, then pq must be perpendicular to a pair of supporting lines $l_t(A, p)$ and $l_t(B, q)$.

OBSERVATION 73. *Let A and B be two convex regions. Let pq be the line segment realising $\mathbf{d}(A, B)$, where $p \in A$ and $q \in B$. The segment pq must be perpendicular to a pair of supporting lines $l_t(A, p)$ and $l_t(B, q)$.*

We will show that if pq realises the distance between two 0-regions, we can transform pq into another segment pq' such that pq' is parallel to some direction $r(k)$, and $\|pq'\|$ approximates $\|pq\|$. To do this, we first need a fact.

LEMMA 74. *Given a segment pq, let α (resp. β) be the acute angle between pq and $r(p, k+1)$ (resp. $r(p, k)$), where $\alpha + \beta = \theta < \pi/6$. Let q' be the intersection of $r(p, k+1)$ and $r(q, k\theta + \pi/2)$. We have that $\|pq'\| = (\cos(\beta)/\cos(\theta)) \cdot \|pq\|$, and $\|qq'\| = (\sin(\alpha)/\cos(\theta)) \cdot \|pq\|$.*

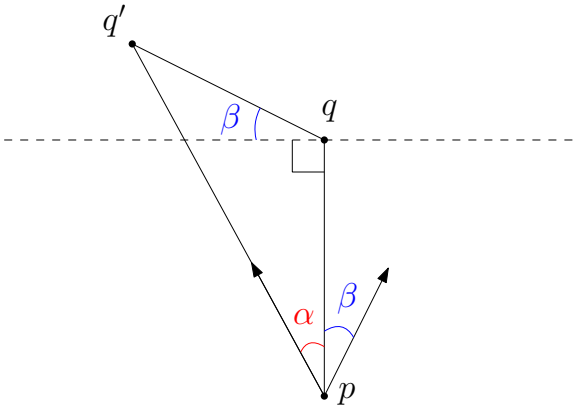

FIGURE 5.2: The construction in Lemma 74. The segment $\|pq\|$ lies between two directions, and α and β are the respective angles, where $\alpha + \beta = \theta < \pi/6$.

PROOF. See Figure 5.2 for the construction. By the law of sines, we have

$$\frac{\|pq'\|}{\sin(\measuredangle pqq')} = \frac{\|pq\|}{\sin(\measuredangle pq'q)} \implies \|pq'\| = \sin(\measuredangle pqq') \cdot \frac{\|pq\|}{\sin(\measuredangle pq'q)}.$$

Filling in our angles, we get

$$\|pq'\| = \frac{\sin(\frac{\pi}{2} + \beta)\|pq\|}{\sin(\frac{\pi}{2} - \alpha - \beta)} = \frac{\cos(\beta)}{\cos(\theta)} \cdot \|pq\|.$$

Similarly, by the law of sines, we have

$$\frac{\|qq'\|}{\sin(\measuredangle qpq')} = \frac{\|pq\|}{\sin(\measuredangle pq'q)} \implies \|qq'\| = \sin(\measuredangle qpq') \cdot \frac{\|pq\|}{\sin(\measuredangle pq'q)}.$$

Filling in our angles, we get

$$\|qq'\| = \frac{\sin(\alpha)\|pq\|}{\sin(\frac{\pi}{2} - \alpha - \beta)} = \frac{\sin(\alpha)}{\cos(\theta)} \cdot \|pq\|.$$

The proof is complete. □

Let pq realise $\mathbf{d}(A, B)$. We consider the scenario when $\|pq\|$ is relatively small compared to the horizontal span of (say) B. Using the above lemma, we will show that we have constructed a set of edges in $\mathcal{E}$ connecting two sample points $a \in A$ and $b \in B$, such that the total weight of these edges approximates $\|pq\|$. Recall that P^* is a weighted shortest path between two 0-regions, and see the definition of α and β in Lemma 74.

LEMMA 75. *Let $pq \subseteq P^*$, and let pq realise $\mathbf{d}(A, B)$, where $p \in A$ and $q \in B$. Let $p \in \partial A(a', a)$, and $q \in \partial B(b, b')$, where points a and a' (resp. b and b') are adjacent sample points on 0-region A (resp. B). If $\max\{\|pa'\|, \|qb\|\} \geq (\sin(\alpha)/\cos(\theta)) \cdot \|pq\|$ or $\max\{\|pa\|, \|qb'\|\} \geq (\sin(\beta)/\cos(\theta)) \cdot \|pq\|$, then there exists a path $P \subseteq \mathcal{E}$ from A to B such that $\mathbf{w}(P) \leq (\cos(\beta)/\cos(\theta)) \cdot \|pq\|$.*

PROOF. Without loss of generality, assume that $\|qb\| \geq (\sin(\alpha)/\cos(\theta)) \cdot \|pq\|$. Lemma 74 implies that there exists a point $q' \in \partial B(b, q)$ such that pq' is in some direction $r(k)$, and $\|pq'\| \leq (\cos(\beta)/\cos(\theta)) \cdot \|pq\|$.

Observe that pq cannot overlap any 0-region; otherwise, P^* is not optimal. If pq' does not overlap any 0-region (see Figure 5.3, left), we fix the orientation of pq', and move p along $\partial A(p, a)$ and q along $\partial B(b, q)$, until pq' touches a sample point.

If pq' touches a (resp. b), then $r(a, k)$ (resp. $r(b, k\theta + \pi)$) hits B (resp. A). If pq' touches a sample point $v \notin A \cup B$, v must be extreme in the direction $r(k\theta - \pi/2)$. As a result, $r(v, k)$ hits B, and $r(v, k\theta+\pi)$ hits A. In either case, A and B are adjacent in some face of $\mathcal{M}(k)$, and

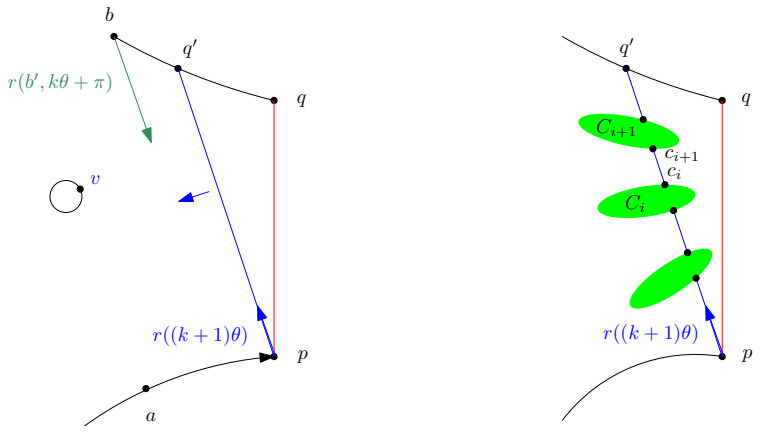

FIGURE 5.3: In the left figure, if pq' does not overlap a 0-region, we slide pq' until it touches a sample point. In the right figure, we slide each inter-region segment (cd as an example) the same way.

the edge $e = (ak(A), ak(B))$ is in $\mathcal{E}$ by construction. As a result, $\mathbf{w}(e) = \mathbf{d}(A, B) = \|pq\|$. This is also trivially true when p or q' is already a sample point.

Otherwise, the segment pq' overlaps a set E' of 0-regions, and there exists a path P from A to B through E' (see Figure 5.3, right). Since the 0-regions do not overlap, the boundaries of the 0-regions in E' partition pq' into a set of intra-region and inter-region segments. Let cd be one among the set S of inter-region segments, where c is on a 0-region C, and d is on a 0-region D. Using the same argument as above, one can slide cd until it touches a sample point, and edge $(ak(C), ak(D)) \in \mathcal{E}$ exists by construction.

In total, traveling from A to B via the 0-regions E' must be less costly than $\|pq'\|$, since $\mathbf{w}(ak(C), ak(D)) = \mathbf{d}(C, D) \leq \|cd\|$, and the intra-region segments have weight 0. Summing up the cost of P, we have that

$$\mathbf{w}(P) = \sum_{cd \in S} \mathbf{w}(ak(C), ak(D)) < \sum_{cd \in S} \|cd\| < \|pq'\| = \frac{\cos(\beta)}{\cos(\theta)} \cdot \|pq\|. \qquad \square$$

5.2.3 Θ-Graph

In Step 4 of Algorithm 1, we constructed a Θ-graph $\mathcal{G}_\Theta = (\mathcal{V}_\Theta, \mathcal{E}_\Theta)$ using the defined sample points. The vertices $\mathcal{V}_\Theta$ are simply all sample points. The edges in $\mathcal{E}_\Theta$ are constructed using

the standard Θ-graph construction [15]. Recall that in Algorithm 1, for every edge $(p, q) \in \mathcal{E}_\Theta$, with $p \in A$, $q \in B$, and $A \neq B$, we add an edge (p, q) to $\mathcal{E}$, and set $\mathbf{w}(p, q) = \|pq\|$.

The Θ-graph gives us enough information to construct a set of "good" edges in $\mathcal{E}$ when the distances between p (resp. q) and its adjacent sample points are small compared to $\|pq\|$. In this case, we argue that there exists a pair of sample points $a \in A$ and $b \in B$, such that $\|ab\| \leq (1/\cos(\theta)) \cdot \|pq\|$. Similar to Lemma 74, we first prove a geometric property.

LEMMA 76. *Given a segment pq, let α (resp. β) be the acute angle between pq and $r(p, k+1)$ (resp. $r(p, k)$), where $\alpha + \beta = \theta < \pi/6$. Let q' be the intersection of $r(p, k+1)$ and $r(q, k\theta + \pi/2)$, and let p' be the intersection of $r(q, k\theta + \pi)$ and $r(p, (k+1)\theta + \pi/2)$. Let c be the intersection of pq' and qp'. We have that $\|cp'\| + \|cq'\| = (\sin(\alpha) + \sin(\beta))/(\cos(\theta)\sin(\theta)) \cdot \|pq\|$, and $\|p'q'\| = (1/\cos(\theta)) \cdot \|pq\|$.*

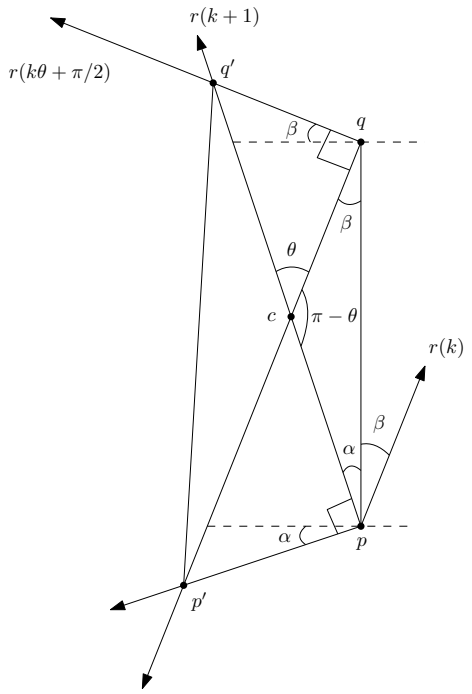

FIGURE 5.4: The construction in Lemma 76.

PROOF. See Figure 5.4 for the construction. Without loss of generality, assume $\|pq\| = 1$. Using the law of sines in the triangle $\triangle pqc$, we have the following.

$$\frac{\|cq\|}{\sin(\measuredangle qpc)} = \frac{\|pq\|}{\sin(\measuredangle qcp)} \implies \|cq\| = \frac{\sin(\measuredangle qpc)}{\sin(\measuredangle qcp)} \cdot \|pq\| = \frac{\sin(\alpha)}{\sin(\pi - \theta)} = \frac{\sin(\alpha)}{\sin(\theta)} \qquad (1)$$

Observe that $\measuredangle cqq' = \pi/2$, since $\measuredangle cqq'$ is the result of rotating $r(q, k\theta + \pi/2)$ counter-clockwise by $\pi/2$. Therefore, $\triangle cqq'$ is a right triangle, and we have the following.

$$\begin{aligned}
\|cq'\| &= \frac{\|cq\|}{\sin(\measuredangle cq'q)} \\
&= \frac{\|cq\|}{\sin(\frac{\pi}{2} - \theta)} && \triangleright\ \measuredangle cq'q = \frac{\pi}{2} - \theta \text{ by construction} \\
&= \frac{1}{\cos(\theta)} \cdot \frac{\sin(\alpha)}{\sin(\theta)} && \triangleright\ \text{Using (1)}
\end{aligned}$$

Using an analogous computation, we obtain

$$\|cp'\| = \frac{1}{\cos(\theta)} \cdot \frac{\sin(\beta)}{\sin(\theta)}.$$

Combining $\|cp'\|$ and $\|cq'\|$, we have that

$$\|cp'\| + \|cq'\| = \frac{\sin(\alpha) + \sin(\beta)}{\cos(\theta)\sin(\theta)}. \tag{2}$$

Next, since $\triangle p'qq'$ is a right triangle, we can bound $\|p'q'\|$ as follows.

$$\begin{aligned}
\|p'q'\|^2 &= \|qq'\|^2 + \|qp'\|^2 \\
&= (\frac{\sin(\alpha)}{\cos(\theta)})^2 + (\|cp'\| + \|cq\|)^2 && \triangleright\ \text{Using Lemma 74} \\
&= (\frac{\sin(\alpha)}{\cos(\theta)})^2 + (\frac{\sin(\beta)}{\cos(\theta)\sin(\theta)} + \frac{\sin(\alpha)}{\sin(\theta)})^2 && \triangleright\ \text{Using (2)} \\
&= (\frac{\sin(\alpha)}{\cos(\theta)})^2 + (\frac{\sin(\theta - \alpha)}{\cos(\theta)\sin(\theta)} + \frac{\sin(\alpha)}{\sin(\theta)})^2 \\
&= (\frac{\sin(\alpha)}{\cos(\theta)})^2 + (\frac{\sin(\theta)\cos(\alpha) - \sin(\alpha)\cos(\theta)}{\cos(\theta)\sin(\theta)} + \frac{\sin(\alpha)}{\sin(\theta)})^2 \\
&= (\frac{\sin(\alpha)}{\cos(\theta)})^2 + (\frac{\cos(\alpha)}{\cos(\theta)} - \frac{\sin(\alpha)}{\sin(\theta)} + \frac{\sin(\alpha)}{\sin(\theta)})^2 \\
&= (\frac{1}{\cos(\theta)})^2
\end{aligned}$$

Since $0 < \theta < \pi/6$, all terms are greater than 0, and $\|p'q'\| = (1/\cos(\theta)) \cdot \|pq\|$. □

In the case that both p and q are close to their adjacent sample points, we argue that there exists a pair (a, b) of sample points such that $\|ab\|$ approximates $\|pq\|$.

LEMMA 77. *Let $pq \subseteq P^*$, and let pq realise $\mathbf{d}(A, B)$, where $p \in A$ and $q \in B$. Let $p \in \partial A(a', a)$, and $q \in \partial B(b, b')$, where points a and a' (resp. b and b') are adjacent sample points on A (resp. B). If $\max\{\|pa'\|, \|qb\|\} < (\sin(\alpha)/\cos(\theta)) \cdot \|pq\|$ and $\max\{\|pa\|, \|qb'\|\} < (\sin(\beta)/\cos(\theta)) \cdot \|pq\|$, then $\mathbf{d}(a, b) < (1/\cos(\theta)) \cdot \|pq\|$.*

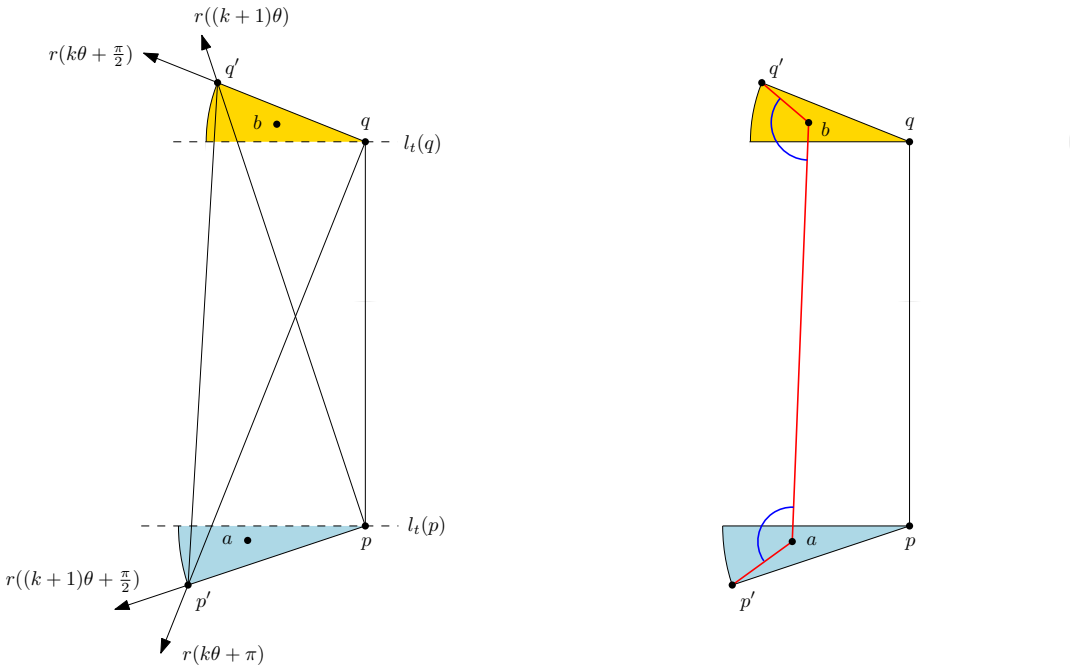

FIGURE 5.5: With b fixed, moving a to p' strictly increases $\|ab\|$.

PROOF. We argue that $\|ab\| < \|p'q'\|$ (see Figure 5.5), where q' (resp. p') is the intersection of $r(p, (k+1)\theta)$ (resp. $r(q, k\theta+\pi)$) and B (resp. A). Let p'' (resp. q'') be the intersection of $r(q, k\theta + \pi)$ (resp. $r(p, k + 1)$) and $l_t(p)$ (resp. $l_t(q)$). Without loss of generality, assume pq is parallel to the y-axis, and $y(q) > y(p)$, and consider the sample point a next to p counter-clockwise.

The region A is convex; therefore a must be below the supporting line $l_t(p)$. Due to how the sample points are constructed, the sample point a must be above $r(p, (k + 1)\theta + \pi/2)$. Since

$\|pa\| < (\sin(\beta)/\cos(\theta)) \cdot \|pq\|$, a must reside in the disk $D = D(p, \|pp'\|)$ centred at p with radius $\|pp'\|$. Combining the above restrictions, we have that a must reside in the smaller circular sector S of D enclosed by $l_t(p)$ and pp'. An analogous argument shows that b must reside in the smaller circular sector of $D(q, \|qq'\|)$ enclosed by $l_t(q)$ and qq'.

Consider the line l through ab, and observe that pushing a along l towards the boundary of S strictly increases $\|ab\|$. Next, now that a lies on the boundary of S, we push a towards p' along the boundary of S. This also increases $\|ab\|$, since $\measuredangle bap' \geq \pi/2$. Now that $a = p'$, an analogous argument applies to b, since $\measuredangle p'bq' \geq \pi/2$. In conclusion, $\|ab\| \leq \|p'q'\|$.

By Lemma 76, $\|ab\| \leq \|p'q'\| \leq (1/\cos(\theta)) \cdot \|pq\|$, and the proof is complete. □

5.2.4 The quality of the path

For now, assume that s and t lie in some 0-region. An optimal s-t path P^* consists of a set of segments, where the endpoints of each segment lie on the boundaries of the 0-regions. A segment pq either lies within a 0-region or connects two different 0-regions. Since it costs nothing to follow an edge inside a 0-region, the weight of P^* is the total weight of those edges connecting different 0-regions.

Let pq realise the distance between 0-regions A and B, where p lies on ∂A between sample points a and a', and q lies on ∂B between sample points b and b'. In Lemma 75, we have shown that if $\max\{\|pa'\|, \|qb\|\} \geq (\sin(\alpha)/\cos(\theta)) \cdot \|pq\|$ or $\max\{\|pa\|, \|qb'\|\} \geq (\sin(\beta)/\cos(\theta)) \cdot \|pq\|$, there exists a path $P \subseteq \mathcal{E}$ from a sample point on A to a sample point on B of length at most $(\cos(\beta)/\cos(\theta)) \cdot \|pq\|$.

In Lemma 77, we have shown that if $\max\{\|pa'\|, \|qb\|\} < (\sin(\alpha)/\cos(\theta)) \cdot \|pq\|$ and $\max\{\|pa\|, \|qb'\|\} < (\sin(\beta)/\cos(\theta)) \cdot \|pq\|$, there exist sample points $a \in A$ and $b \in B$, such that $\mathbf{d}(a, b) < (1/\cos(\theta)) \cdot \|pq\|$. To obtain a path between a and b in this case, we rely on the Θ-graph. The tightest bounds on the length of this path are due to Bose et al. [22], who showed that the spanning ratio of a Θ-graph is at most $r_\theta = 1 + 2\sin(\theta/2)/(\cos(\theta/2) - \sin(\theta/2))$.

In summary, if $pq \subseteq P^*$, there exists a path P from A to B such that

$$\mathbf{w}(P) \leq \max\{\frac{\cos(\beta)}{\cos(\theta)}, \frac{1}{\cos(\theta)} \cdot r_\theta\} \cdot \mathbf{w}(P^*) \leq \frac{1}{\cos(\theta)} \cdot r_\theta \cdot \mathbf{w}(P^*).$$

Given an approximation factor $0 < \varepsilon < 1$, we compute a proper value for $0 < \theta < \pi/6$ as follows.

$$\frac{1}{\cos(\theta)} \cdot (1 + \frac{2\sin(\frac{\theta}{2})}{\cos(\frac{\theta}{2}) - \sin(\frac{\theta}{2})}) = \frac{1}{1 - \sin(\theta)} \leq 1 + \varepsilon \implies \theta \leq \sin^{-1}(\frac{\varepsilon}{1+\varepsilon}) \in \mathcal{O}(\varepsilon)$$

Using θ, we can construct the data structure $\mathcal{B}$. Combining Lemma 75 and Lemma 77, we have the following.

LEMMA 78. *Data structure $\mathcal{B}$ contains a path $P \subseteq \mathcal{E}$ from sample point a to sample point b such that $\mathbf{w}(P) \leq (1+\varepsilon) \cdot \mathbf{w}(P^*)$, where P^* is the optimal path from a to b.*

5.2.5 Finding a shortest path amidst 0-regions

Now that we know good paths exist between all pairs of sample points, it remains to include our arbitrary query points s and t. Given the data structure $\mathcal{B} = \{\mathcal{M}(k) \mid \forall k \in [0, 2\pi/\theta), k \in \mathbb{Z}\} \cup \{\mathcal{G}, \mathcal{G}_\Theta\}$, a point s, and a point t, we query the approximate shortest path from s to t using Algorithm 2.

Once s and t are added to $\mathcal{B}$, they are treated as 0-regions with no interiors. The point s is its extreme point in every direction. Therefore, the distance between s and its adjacent sample points is 0. As a result, Lemma 75 and Lemma 77 both apply. Thus $\mathcal{B}$ with s and t added contains a $(1+\varepsilon)$-approximation of the shortest path from s to t, which Dijkstra's shortest path algorithm will find.

5.2.5.1 Analysis

Finally, we look at the query time. Recall that $\varepsilon \in \mathcal{O}(\theta)$. In Step 1, we consider $\mathcal{O}(1/\varepsilon)$ trapezoidal maps. For each trapezoidal map, it takes $\mathcal{O}(\log(n/\varepsilon))$ time to perform a point-location query [15], and it takes $\mathcal{O}(\log N)$ time to compute $\mathbf{d}(s, A)$ [69]. In Step 2, it takes

Algorithm 2 Query s-t weighted shortest path amidst 0-regions

This algorithm takes as input a point s and a point t, and a data structure $\mathcal{B} = \{\mathcal{M}(k) \mid \forall k \in [0, 2\pi/\theta), k \in \mathbb{Z}\} \cup \{\mathcal{G}, \mathcal{G}_\Theta\}$ storing a set of 0-regions. It outputs a $(1+\varepsilon)$-approximate weighted shortest path from s to t. In Step 1 and Step 2, this algorithm shows how to add s to $\mathcal{B}$, and the same operations are used to add t.

(1) For each trapezoidal map $\mathcal{M}(k)$, do the following for point s.
   (a) Query the face F containing s.
   (b) Let F be adjacent to 0-regions A and B. Add $e = (s, ak(A))$ to $\mathcal{E}$, and set $\mathbf{w}(e) = \mathbf{d}(s, A)$. Perform the same operation for s and B.

(2) Add s to $\mathcal{G}_\Theta$. Specifically, using s as the apex, we construct a set of disjoint cones with θ-angle. For each point p closest to s in each cone, add $e = (s, p)$ to $\mathcal{E}$, and set $\mathbf{w}(e) = \|sp\|$. For every existing vertex $p \in \mathcal{G}_\Theta$, and every existing edge $(p, q) \in \mathcal{E}_\Theta$, if s is closer to p than q is, add $e = (s, p)$ to $\mathcal{E}$, and set $\mathbf{w}(e) = \|sp\|$. (Note that Θ-graph uses projected distance instead of Euclidean distance.)

(3) Use Dijkstra's shortest path algorithm to compute a path P' from s to t in $\mathcal{G}$. Transform P' into a path P amidst the 0-regions and return P.

$\mathcal{O}(n/\varepsilon^2 + (n/\varepsilon)\log(n/\varepsilon))$ time to find the closest point of s in each cone, and at the same time check if s is closest to any point p. In Step 3, it takes $\mathcal{O}(|\mathcal{E}| + |\mathcal{V}| \log |\mathcal{V}|) = \mathcal{O}(n/\varepsilon^2)$ time to run Dijkstra's shortest path algorithm to find the shortest path P' from s to t in $\mathcal{V}$ [72]. It takes $\mathcal{O}(n/\varepsilon^2 + N)$ time to transform P' into a path P in the environment. The query time is $\mathcal{O}(N + n/\varepsilon^2 + (n/\varepsilon)\log(n/\varepsilon) + (\log N)/\varepsilon)$.

THEOREM 14. *Consider a planar subdivision induced by a plane with weight 1, containing a set $\mathcal{Z}$ of non-overlapping convex 0-regions with weight 0. Let $|\mathcal{Z}| = n$ and N denote the total number of vertices in $\mathcal{Z}$. For any approximation factor $0 < \varepsilon < 1$, a data structure $\mathcal{B}$ can be constructed over $\mathcal{Z}$ in $\mathcal{O}(N + (n/\varepsilon^2)(\log(n/\varepsilon) + \log N))$ expected time, with a total size of $\mathcal{O}(N + n/\varepsilon^2)$. When queried with points s and t, $\mathcal{B}$ can return a weighted path P from s to t in $\mathcal{O}(N + n/\varepsilon^2 + (n/\varepsilon)\log(n/\varepsilon) + (\log N)/\varepsilon)$ time, satisfying $\mathbf{w}(P) \leq (1+\varepsilon) \cdot \mathbf{w}(P^*)$, where P^* is the optimal weighted shortest path from s to t.*

5.3 Partial weak Fréchet similarity

This section highlights one application of our data structure: approximating the partial weak Fréchet similarity. This problem was previously considered by Buchin et al. [31] and De Carufel et al. [48]. We start with an introduction to the Fréchet distance and the weak Fréchet

distance, eventually leading us to the formal definition of the partial weak Fréchet similarity problem.

Derived from the standard (strong) Fréchet distance problem, the notion of weak Fréchet distance removes the requirement that the person and the dog must move forward at all times. This means that the reparameterisation f that describes the movement still needs to be continuous but not necessarily monotone. To determine if the weak Fréchet distance $\delta_{wF}(P, Q)$ is at most d, we need to find only a (potentially not xy-monotone) path through the freespace from s to t in $\mathcal{D}_\delta(P, Q)$. Buchin et al. [32] showed that there is no strongly subquadratic time algorithm for approximating the weak Fréchet distance within a factor less than 3 unless the strong exponential-time hypothesis fails.

Buchin et al. [31] proposed the *partial Fréchet similarity* (partial similarity in short) to deal with the Fréchet distance's sensitivity to outliers. Instead of determining whether a leash of length d is enough to complete the walk, partial similarity determines how much can be completed given a leash of length d. The partial similarity is the total length of the portion of two curves that are matched under the Fréchet distance d. Let $\|xy\|$ be the distance between point x and point y under the L_p norm. Let $\|v\|$ be the L_2 norm of the vector v. Under the L_p metric, given the desired Fréchet distance d, the partial similarity $S_{f,g}(P, Q)$ of curves P and Q with respect to the reparameterisations f and g is formally defined as follows [31].

$$S_{f,g}(P, Q) = \inf_{\|P(f(t))-Q(g(t))\|\leq d} (\|P'(f(t))\| + \|Q'(g(t))\|)\ dt$$

Naturally, we want to compute a pair of reparameterisations f and g that maximise the partial similarity. To do this, Buchin et al. [31] proposed a cubic time algorithm under L_1. They showed that it is sufficient to find an xy-monotone and *rectilinear* path P from s to t such that P intersects as much freespace as possible, where s (resp. t) is the bottom-left (resp. top-right) corner of the freespace diagram.

Under the weak Fréchet distance, the monotonicity requirement is removed. But since a path P can traverse back and forth in the freespace diagram, it is no longer meaningful for P to intersect as much freespace as possible. We instead are interested in computing a path that

intersects as little forbidden space as possible to minimise the portions of the two curves that are not matched within distance d. Therefore, solving the partial weak Fréchet distance problem under the L_2 metric is equivalent to finding a weighted rectilinear shortest path amidst a set of $\mathcal{O}(n^2)$ non-overlapping and convex 0-regions embedded in the plane (the forbidden space) with weight 1. By Fact 1, a 0-region is the freespace within a cell, which has constant complexity.

Amidst the 0-regions and measured in L_p metric, let OPT_{L_p} denote the weight of the weighted shortest path from s to t, and let ALG_{L_p} denote the weight of the weighted path computed by Theorem 14. Since $OPT_{L_1} \leq \sqrt{2} \cdot OPT_{L_2}$, we have

$$OPT_{L_2} \leq OPT_{L_1} \leq \sqrt{2} \cdot OPT_{L_2} \leq \sqrt{2} \cdot ALG_{L_2} \leq \sqrt{2} \cdot (1+\varepsilon) OPT_{L_2},$$

which leads to the following theorem.

THEOREM 15. *One can approximate the partial weak Fréchet similarity of two curves with respect to the L_2 metric within a factor of $(\sqrt{2}+\varepsilon)$ in $\mathcal{O}((n^2/\varepsilon^2)\log(n/\varepsilon))$ expected time.*

5.4 Shortest path amidst 0-regions and obstacles

In this section, we generalise our data structure from Section 5.2 to allow convex obstacles that cannot be traversed, i.e., obstacles with weight ∞. Our problem is finding an approximate shortest path amidst 0-regions and obstacles. More concretely, we consider the following problem.

PROBLEM 8. *In the planar-subdivision induced by the plane with weight 1, and a set of non-overlapping convex regions consisting of obstacles with weight ∞, and 0-regions with weight 0, given an approximation error $0 < \varepsilon < 1$, find a $(1+\varepsilon)$-approximate weighted shortest path from point s to point t.*

In this section, we redefine $\mathbf{d}(A, B)$ as the minimum distance between two geometric regions A and B in a $0/1/\infty$ weighted setting. The Θ-graph can be constructed in an environment with obstacles. Clarkson [59] described such a construction over points and polygonal

obstacles, and proved that a path that $(1+\varepsilon)$-approximates $\mathbf{d}(a,b)$ exists in the Θ-graph, where a and b are vertices. We will use this Θ-graph in the rest of the chapter.

Like in the previous section, we will first describe the construction of the data structure $\mathcal{B}$ and analyse the time and space complexity. We then show that $\mathcal{G} \in \mathcal{B}$ contains a good path between every pair of sample points. We use this to argue the approximation ratio for arbitrary s and t.

5.4.1 Construction of the data structure

In order to deal with obstacles, we need to define two new types of sample points. For clarity, we refer to the sample points defined previously as the *original sample points*. In Section 5.2, a trapezoidal map $\mathcal{M}(k)$ was only used to determine if two 0-regions should be connected, and we did not explicitly compute the intersection of a vertical segment and the boundary of a region. With the introduction of obstacles, we do need such intersections. Consider a sample point a. When constructing $\mathcal{M}(k)$, we shoot two vertical rays from $a \in A$, one upwards and one downwards. Let p be the first intersection of $r(a,k)$ with the boundary of some region that is not A. We call p a *propagated sample point*.

The other type of sample points we need is the *tangent sample points*. Given two disjoint obstacles A and B, and a common tangent l, if l touches A at point a and B at b, we add a and b as tangent sample points. When we say a point a is a sample point, a can be any type of sample point. If l coincide with one or consecutive segments of A (resp. B), then we pick the nearest point to B (resp. A) as the tangent sample point.

Recall that $\texttt{simpl}(A)$ is the simplified region by connecting every pair of adjacent sample points of A. $\texttt{simpl}(\mathcal{Z})$ is the set of simplified 0-regions, and $\texttt{simpl}(\mathcal{O})$ is the set of simplified obstacles. We formally define the construction of our data structure. Given a set $\mathcal{O}$ of convex obstacles and a set $\mathcal{Z}$ of convex 0-regions, we build our data structure using Algorithm 3.

Algorithm 3 Construct $\mathcal{B}$ with 0-regions and obstacles

This algorithm takes as input a set of non-overlapping and convex regions, including 0-regions $\mathcal{Z}$ and obstacles $\mathcal{O}$, and constructs a data structure $\mathcal{B}$. The undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is initially empty.

(1) Compute the original sample points $\mathcal{SP}(\mathcal{Z}) \cup \mathcal{SP}(\mathcal{O})$, and add them to $\mathcal{V}$.
(2) For each direction $r(k)$, generate a trapezoidal map $\mathcal{M}(k)$ using $\texttt{simpl}(\mathcal{Z}) \cup \texttt{simpl}(\mathcal{O})$.
(3) For every $\mathcal{M}(k)$, do the following for every face F adjacent to A and B, $A \neq B$.
  (a) Compute the propagated sample points, and add them to $\mathcal{V}$.
  (b) If A and B are both 0-regions, add $e = (ak(A), ak(B))$ to $\mathcal{E}$, and set $\mathbf{w}(e) = \mathbf{d}(A, B)$.
  (c) If at least one of A and B is an obstacle, let ab and $a'b'$ be the vertical segments defining F, where $a, a' \in A$, and $b, b' \in B$. Add edges $e_1 = (a, b)$ and $e_2 = (a', b')$ to $\mathcal{E}$. Set $\mathbf{w}(e_1) = \|ab\|$ and $\mathbf{w}(e_2) = \|a'b'\|$.
  (d) If A and B are both obstacles, we compute their common tangents. For each common tangent that touches A at a and B at b, add a and b to $\mathcal{V}$.
(4) Redefine $\texttt{simpl}(A)$ as the polygon generated by connecting adjacent sample points of A, original, propagated, and tangent sample points included. With $\mathcal{V}$ and $\texttt{simpl}(\mathcal{O})$ as the input, generate a Θ-graph $\mathcal{G}_\Theta = (\mathcal{V}, \mathcal{E}_\Theta)$.
  - For each edge $(p, q) \in \mathcal{E}_\Theta$, if p and q belong to different regions A and B, add $e = (p, q)$ to $\mathcal{E}$ and set $\mathbf{w}(e) = \|pq\|$.
(5) For every pair of adjacent sample points a, a' on an obstacle, add $e = (a, a')$ to edges, and set $\mathbf{w}(e) = \|aa'\|$.
(6) Pick an arbitrary sample point as the anchor $ak(A)$ for every $A \in \mathcal{Z}$. For each sample point a of a 0-region A, add $e = (a, ak(A))$ to $\mathcal{E}$, and set $\mathbf{w}(e) = 0$.
(7) Return $\mathcal{B} = \{\mathcal{M}(k) \mid \forall k \in [0, 2\pi/\theta), k \in \mathbb{Z}\} \cup \{\mathcal{G}, \mathcal{G}_\Theta\}$ as the data structure.

5.4.1.1 Analysis

Recall that we are given an approximation error ε and a set of non-overlapping convex regions consisting of obstacles $\mathcal{O}$ and 0-regions $\mathcal{Z}$. There are n regions, and the total complexity of $\mathcal{O} \cup \mathcal{Z}$ is $\mathcal{O}(N)$. In Step 1, it takes $\mathcal{O}(N + n/\varepsilon)$ time to compute the sample points. In Step 2, it takes $\mathcal{O}((n/\varepsilon^2)\log(n/\varepsilon))$ expected time to construct the trapezoidal maps [15].

Step 3 considers $\mathcal{O}(n/\varepsilon^2)$ trapezoidal map faces. In Step 3a, given a vertical segment c intersecting a non-vertical segment $ab \in A$, it takes $\mathcal{O}(\log N)$ time to use a binary search on $\partial A(a, b)$ to find the intersection of c and $\partial A(a, b)$. In Step 3b, using the algorithm by Edelsbrunner [69], it takes $\mathcal{O}(\log N)$ time to compute the distance between two 0-regions. In Step 3c, it takes constant time to compute the length of the vertical segments. In Step 3d, using

the algorithm by Kirkpatrick and Snoeyink [116], and Guibas et al. [98], it takes $\mathcal{O}(\log N)$ time to compute at most four common tangents. In total, Step 3 takes $\mathcal{O}((n/\varepsilon^2)\log(N))$ time, and we add at most $\mathcal{O}(n/\varepsilon^2)$ tangent and propagated sample points.

In Step 4, using Clarkson's algorithm [59], it takes $\mathcal{O}((n/\varepsilon^3)\log(n/\varepsilon))$ time to generate a Θ-graph with obstacles using $\mathcal{O}(n/\varepsilon^2)$ sample points and $\mathcal{O}(1/\varepsilon)$ cones. In Step 5, it takes $\mathcal{O}(n/\varepsilon^2 + N)$ time to generate an edge and set the weight for every pair of adjacent sample points. Step 6 takes $\mathcal{O}(n/\varepsilon^2)$ time to connect every sample point in a 0-region to its anchor.

We generate $\mathcal{O}(n/\varepsilon)$ original sample points. We generate a constant number of propagated and tangent sample points for each of the $\mathcal{O}(n/\varepsilon^2)$ faces. As a result, each propagated or tangent sample point contributes a constant number of edges in $\mathcal{E}$. A Θ-graph constructed with $\mathcal{O}(n/\varepsilon^2)$ sample points, and $\mathcal{O}(1/\varepsilon)$ cones has at most $\mathcal{O}(n/\varepsilon^3)$ edges. Therefore $|\mathcal{G}| = \mathcal{O}(n/\varepsilon^3)$. We summarise the complexities below.

LEMMA 79. *Given an approximation error $0 < \varepsilon < 1$, and n non-overlapping convex regions including 0-regions and obstacles with total complexity N, one can build the data structure $\mathcal{B}$ in $\mathcal{O}(N + (n/\varepsilon^3)(\log(n/\varepsilon) + \log N))$ expected time, and the total size of $\mathcal{B}$ is $\mathcal{O}(N + n/\varepsilon^3)$.*

The structure of the rest of the section is as follows. By Lemma 80, the distance between two adjacent sample points on the boundary of an obstacle approximates the straight line segment. Therefore, we show that we can "snap" the vertices of an optimal path to our sample points. For every segment $pq \subseteq P^*$, we then argue that either the trapezoidal map or the Θ-graph contains a path approximating $\|pq\|$ to within a factor of $1 + \varepsilon$.

5.4.2 Walking on the boundary is not expensive

We first argue that if a and a' are adjacent sample points of A, then $\|aa'\|$ approximates $\|\partial A(a, a')\|$ within a factor of $\sec(\theta/2)$. Therefore, if we find a path P amidst the simplified obstacles, we only need to pay a small factor to transform P into a path amidst the original obstacles. Then, we prove that if pq is part of the optimal path, we can replace pq with a path $P \subseteq \mathcal{E}$, such that $\mathbf{w}(P)$ approximates $\|pq\|$.

LEMMA 80. *Let a and b be adjacent sample points on ∂A, where a appears after b in a counter-clockwise walk. We have that $\|\partial A(a,b)\| \leq \sec(\theta/2) \cdot \|ab\|$.*

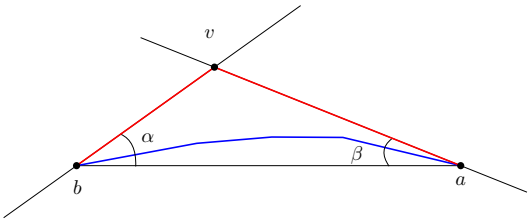

FIGURE 5.6: The length of $\partial A(a,b)$ (blue) is upper-bounded by $\|av\| + \|bv\|$ (red). The two lines are the tangent lines of A through a and b, respectively.

PROOF. See Figure 5.6. Let v be the intersection of $l_t(a)$ and $l_t(b)$. By convexity of A, $\|\partial A(a,b)\| \leq \|av\| + \|vb\|$. By the construction of the sample points, $\measuredangle vab + \measuredangle vba \leq \theta$. Let $\alpha = \measuredangle vab$, and let $\beta = \measuredangle vba$. Then, using the law of sines, we have the following.

$$\|\partial A(a,b)\| \leq \|av\| + \|bv\| \leq \frac{\sin(\beta) \cdot \|ab\|}{\sin(\theta)} + \frac{\sin(\alpha) \cdot \|ab\|}{\sin(\theta)} = \frac{\sin(\alpha) + \sin(\beta)}{\sin(\theta)} \cdot \|ab\|$$

The above expression is maximised when $\alpha = \beta = \theta/2$, resulting in the following.

$$\|\partial A(a,b)\| \leq \frac{2\sin(\frac{\theta}{2})}{\sin(\theta)} = \sec(\frac{\theta}{2}) \cdot \|ab\| \qquad \square$$

The above lemma implies the following. Let $\mathbf{d_S}(a,b)$ be the distance between point a and point b in the environment with simplified obstacles and simplified 0-regions, and let P be the path achieving this distance. If we partition P using the sample points, in the worst case, each segment connects adjacent sample points on obstacles. This implies the following corollary.

COROLLARY 81. *Let a and b be two sample points. We have that $\mathbf{d}(a,b) \leq \sec(\theta/2) \cdot \mathbf{d_S}(a,b)$.*

5.4.3 Snapping a segment of the optimal path to the sample points

Gewali et al. [84] defined three types of *locally optimal* edges joining two simple polygonal regions, and they proved that the shortest path from s to t must be comprised of these locally

optimal edges [84, Lemma 2.5] (ignoring edges in 0-regions). For convex obstacles and 0-regions, we need to consider only four types of segments.

FACT 82. *If segment pq is in the optimal weighted path P^* amidst convex and non-overlapping 0-regions and obstacles, there must exist two supporting lines $l_t(p)$ and $l_t(q)$ such that pq belongs to one of the following cases (ignoring segments in 0-regions).*

1) *pq connects two 0-regions such that $pq \perp l_t(q)$ and $pq \perp l_t(p)$.*
2) *pq connects the point p on a 0-region A and the point q on an obstacle B such that $pq \subset l_t(q)$, and $pq \perp l_t(p)$.*
3) *pq lies on one of the common tangents of two different obstacles.*
4) *p and q are two points on the same obstacle, and $pq = \partial A(p, q)$ or $pq = \partial A(q, p)$.*

In Sections 5.4.3.1, 5.4.3.2, and 5.4.3.3, we handle each type of edge in Case 1, 2, and 3, respectively. For an edge pq in each of these cases, we argue that a good path in our data structure approximates pq. Section 5.4.4 summarises the approximation ratio using the Case 4 edges.

More specifically, in the following subsections, we argue that for each type of segment $pq \subseteq P^*$, there exists a path P constructed using our data structure such that the length of P approximates $\|pq\|$. If pq is parallel to a direction $r(k)$, based on the construction of the sample points and Fact 82, p and q are both sample points, and the argument is straightforward. Therefore, we will focus on the scenario where pq is not parallel to any predefined direction. We assume without loss of generality that pq lies between the direction $r(k)$ and $r(k+1)$, and α (resp. β) is the measure of the acute angle between $r(p, k)$ (resp. $r(p, k+1)$) and pq. Furthermore, by Corollary 81, we consider only simplified obstacles.

5.4.3.1 Case 1: pq connects two 0-regions

We observe that, unfortunately, Lemma 75 does not trivially apply. When we rotate pq to pq', if pq' overlaps obstacles, a path generated using the skewed set of obstacles can be much

longer than $\|pq'\|$, since the path would have to take a detour around the obstacles. The following lemmas resolve this issue.

LEMMA 83. *Let A and B be two 0-regions. Let $pq \subseteq P^*$, where $p \in \partial A$, and $q \in \partial B$. If $\|qb\| \geq (\sin(\alpha)/\cos(\theta)) \cdot \|pq\|$, where b is a sample point adjacent to q, then there exists a sample point $a \in \partial A$ such that $\mathbf{d_S}(a, b) \leq (\cos(\beta)/\cos(\theta)) \cdot \|pq\|$.*

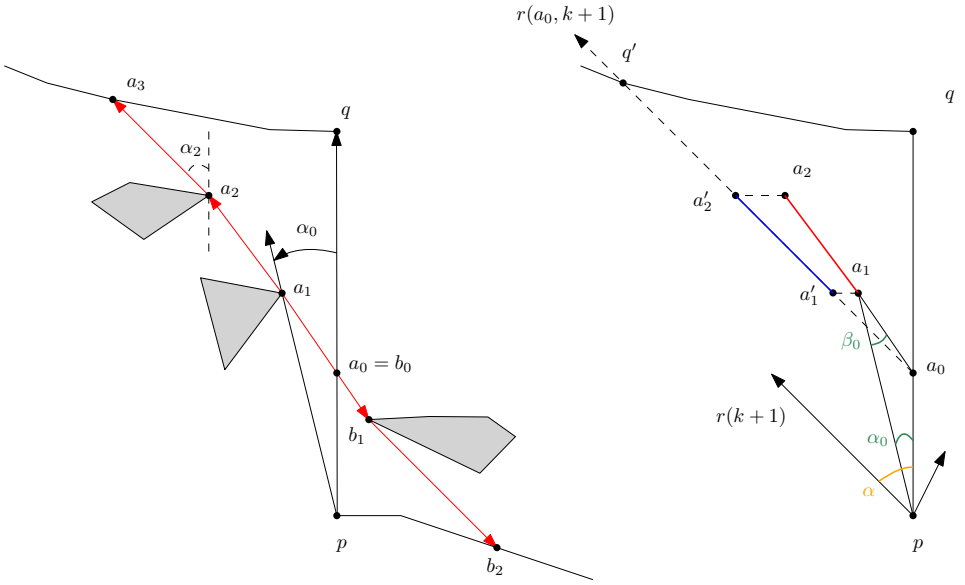

FIGURE 5.7: The x-axis points upwards. In the left figure, the red path is generated with the sweepline process described in Lemma 84. In the right figure, by triangle inequality, $\|a_1a_2\| \leq \|a_1'a_2'\|$, and $\beta_0 + \alpha_0 \leq \alpha$.

PROOF. In Lemma 75, we showed that if $\|qb\| \geq (\sin(\alpha)/\cos(\theta))\|pq\|$, we can transform pq into pq'. If pq' intersects no obstacles, there exists a path P from A to B such that $\|P\|$ approximates $\|pq\|$, which also implies this lemma. Otherwise, pq' intersects at least one obstacle. Without loss of generality, we assume that pq is aligned with the x-axis, where $x(q) > x(p)$.

First, consider a horizontal ray $r = r(p, 0)$, and rotate r counter-clockwise until the new ray, r_0, hits a sample point a_1 on the boundary of some simplified obstacle (see Figure 5.7, left). Let the counter-clockwise rotation from r to r_0 be α_0. With α_{i-1} and a_i defined, let a_{i+1} be the first sample point that is hit by rotating the ray $r_i = r(a_i, \sum_{j=0}^{i-1} \alpha_j)$ counter-clockwise. We continue this process until either r_i is parallel with direction $r(k+1)$, or r_i hits a sample

point on ∂B. Let this final ray be r_l, and r_l hits B at a sample point a_l. Indeed, if r_l is in the direction $r(k+1)$, then $r(a_{l-1}, k)$ hits B, and a_l is a propagated sample point.

Similarly, let $r' = r(a_1, \pi + \alpha_0)$. We perform the same process starting from r'. Let $b_1, \ldots, b_{l'}$ be the sample points generated, where $b_{l'} \in \partial A$, and let $\beta_0, \ldots, \beta_{l'-1}$ be the set of angles generated.

Let the intersection of $a_1 b_1$ and pq be $a_0 = b_0$. We will show the following.

$$\sum_{i=0}^{i=l-1} \|a_i a_{i+1}\| + \sum_{i=0}^{i=l'-1} \|b_i b_{i+1}\| \leq \frac{\cos(\beta)}{\cos(\theta)} \cdot \|pq\| \tag{1}$$

Observe that $\alpha_0 + \beta_0 \leq \alpha$, since the sweep-ray process stops once the ray is parallel with direction $r(k+1)$. Let q' be the point on $\partial B(b, q)$ such that $a_0 q'$ is parallel with $r(k+1)$. Let a_i' be the point on $a_0 q'$, such that $x(a_i') = x(a_i)$ (see Figure 5.7, right). By triangle inequality, $\|a_i a_{i+1}\| \leq \|a_i' a_{i+1}'\|$. By Lemma 74, $\|a_0 q\| < (\cos(\beta)/\cos(\theta)) \cdot \|a_0 q'\|$. Combining the above, we have the following.

$$\sum_{i=0}^{i=l-1} \|a_i a_{i+1}\| < \sum_{i=0}^{i=l-1} \|a_i' a_{i+1}'\| < \frac{\cos(\beta)}{\cos(\theta)} \|a_0 q\|$$

Using an analogous argument on the sample points $\{b_0, \ldots, b_{l'}\}$, and the fact that $\|pq\| = \|pa_0\| + \|b_0 q\|$, we proved Equality (1). We have shown that there exist two sample points a_l and $b_{l'}$, such that $\mathbf{d_S}(a_l, b_{l'}) \leq (\cos(\beta)/\cos(\theta)) \cdot \|pq\|$, completing the proof. □

LEMMA 84. *Let A and B be two 0-regions. Let $pq \subseteq P^*$, where $p \in \partial A$, and $q \in \partial B$. If $\|qb\| < (\sin(\alpha)/\cos(\theta)) \cdot \|pq\|$ and $\|pa\| < (\sin(\beta)/\cos(\theta)) \cdot \|pq\|$, where a (resp. b) is a sample point adjacent to p (resp. q), then $\mathbf{d_S}(a, b) \leq (\sin(\alpha) + \sin(\beta))/(\cos(\theta)\sin(\theta)) \cdot \|pq\|$.*

PROOF. If ab overlaps no obstacles, $\mathbf{d_S}(a, b) \leq \|ab\|$. Therefore, we focus on the case where ab overlaps at least one obstacle. Let a' (resp. b') be the intersection of $r_a = r(a, (k+1)\theta)$ (resp. $r_b = r(b, k\theta)$) and pq. See Figure 5.8, left for the construction.

Observe that if ab overlaps an obstacle C, then C cannot overlap aa' or bb'. Assume the opposite that C overlaps aa'. Since the regions are non-overlapping, and $pq \subseteq P^*$, the sample

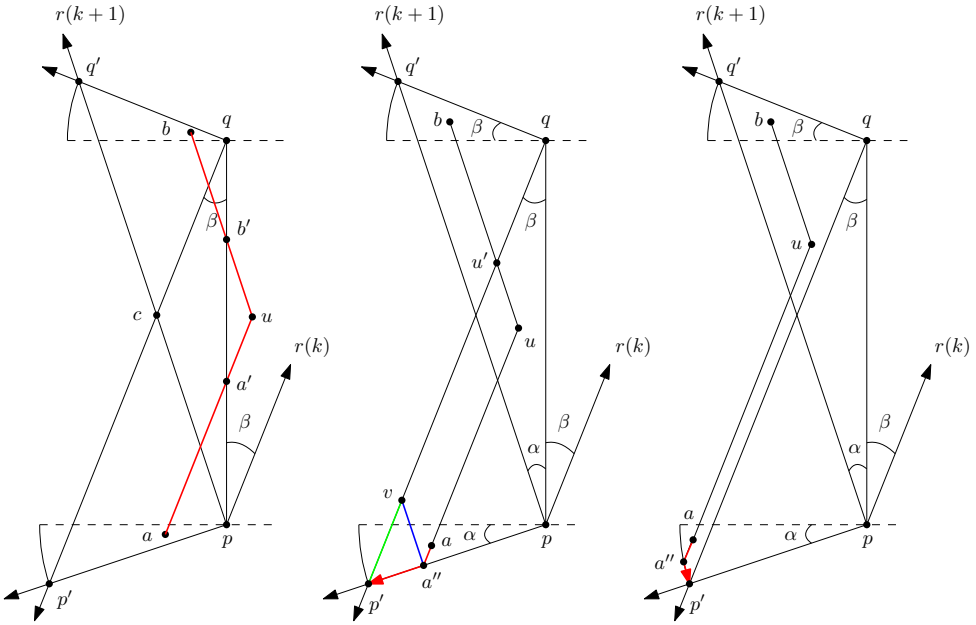

FIGURE 5.8: In the left figure, by triangle inequality, $\|aa'\| + \|a'b'\| + \|b'b\|$ is upper-bounded by $\|au\| + \|ub\|$. In the middle and right figure, fixing b, $\|au\| + \|bu\|$ strictly increases when a first moves to the boundary of the circular sector, and then moves along the boundary to p' (paths marked in red).

point $v = sp(C, k\theta - \pi/2)$ would have to lie in $\triangle apa'$. Therefore, the ray $r(v, (k+1)\theta + \pi)$ hits A. As a result, a propagated sample point is closer to p than a, contradicting the assumption that a is the closest sample point to p. Analogously, C cannot overlap bb'.

Consider the intersection u of $r(a, k)$ and $r(b, (k+1)\theta + \pi)$. We argue that $\mathbf{d_S}(a, b) \leq \|au\| + \|bu\|$. Since no simplified obstacles can intersect r_a or r_b, we can build a convex path P from a to b within $\triangle aub$ circumventing all obstacles. By Lemma 80, $\|P\| \leq \|au\| + \|bu\|$.

Let q' be the intersection of $r(p, (k+1)\theta)$ and $r(q, k\theta+\pi/2)$, and let point p' be the intersection of $r(q, k\theta + \pi)$ and $r(p, (k+1)\theta + \pi/2)$. Using analogous argument in Lemma 76, we argue that $\|ab\|$ is maximised when $a = p'$ and $b = q'$. Since a must be above $r(p, (k+1)\theta + \pi/2)$, below $l_t(p)$ and within a distance $(\sin(\beta)/\cos(\theta)) \cdot \|pq\|$ from p, a must lie in the smaller circular sector S of the disk $D(p, \|pp'\|)$ bounded by $l_t(p)$ and pp'.

With b fixed, consider pushing a along $r(a, k\theta + \pi)$ until a lies on the boundary ∂S of S. a is either above or below $p'q$; see Figure 5.8, middle and right, for an illustration of the respective

case. We will show that $\|au\| + \|bu\|$ strictly increases throughout this process. Observe that by construction, pushing a towards ∂S strictly increases $\|au\|$ while $\|bu\|$ remains unchanged.

Let a'' be the intersection of $r(a, k\theta + \pi)$ and ∂S. If a'' lies on $D(p, \|pp'\|)$, pushing a'' towards p' increases both $\|a''u\|$ and $\|bu\|$. If a'' lies on pp', let v be the intersection of $r(a'', k+1)$ and qp'. Let u' be the intersection of bu and qp'. We argue that $\|a''u\| + \|bu\|$ is maximised when $a'' = p'$ and $u = u'$. Specifically, the length we gain ($\|p'u'\| - \|a''u\|$) is more than the length we lose ($\|uu'\|$).

We argue that $\|p'u'\| - \|a''u\| \geq \|uu'\|$. Since $\Diamond a''vu'u$ is a parallelogram, $\|a''u\| = \|vu'\|$ and $\|uu'\| = \|a''v\|$. Therefore, we have that $\|p'u'\| - \|a''u\| - \|uu'\| = \|p'v\| - \|a''v\|$. By construction, $\triangle vp'a''$ is a right triangle with $p'v$ as the hypotenuse. As a result, $\|p'v\| \geq \|a''v\|$, and $\|p'u'\| - \|a''u\| \geq \|uu'\|$. Therefore, with b fixed, $\|au\| + \|bu\|$ is maximised when $a = p'$. Using an analogous argument by fixing a at p', we have that $\|ab\|$ is maximised when $a = p'$ and $b = q'$. Combining the above arguments and Lemma 76, we have that

$$\mathbf{d_S}(a, b) \leq \|au\| + \|bu\| \leq \|p'c\| + \|q'c\| = \frac{\sin(\alpha) + \sin(\beta)}{\cos(\theta)\sin(\theta)} \cdot \|pq\|. \qquad \square$$

We combine Lemma 83 and Lemma 84 to obtain a bound on the path length where p and q both lie on 0-regions. Note that when two points p and a lie on the boundary of the same 0-region, $\mathbf{d_S}(a, p) = 0$.

LEMMA 85. *Let A and B be two convex 0-regions. Let $pq \subseteq P^*$, where $p \in \partial A$ and $q \in \partial B$. There exists a pair of sample points $a \in A$ and $b \in B$, such that $\mathbf{d_S}(p, a) + \mathbf{d_S}(a, b) + \mathbf{d_S}(b, q) \leq \max\{(\cos(\beta)/\cos(\theta)), (\sin(\alpha) + \sin(\beta))/(\cos(\theta)\sin(\theta))\} \cdot \|pq\|$.*

5.4.3.2 Case 2: pq connects two obstacles

Using Fact 82, we know that if pq connects two obstacles, then pq must coincide with a common tangent of A and B. When two obstacles are close, p (and q) may be very far from its adjacent sample points. Hence, we need the tangent sample points when two obstacles are close (connected via a trapezoidal map). We now show that if p and q lie on obstacles, an approximate path exists in $\mathcal{G} \in \mathcal{B}$.

LEMMA 86. *Let A and B be two convex obstacles. Let $pq \subseteq P^*$, where $p \in \partial A$ and $q \in \partial B$. There exists a pair of sample points $a \in A$ and $b \in B$, such that* $\mathbf{d_S}(p,a)+\mathbf{d_S}(a,b)+\mathbf{d_S}(b,q) \leq (1/\cos(\theta)) \cdot \|pq\|$.

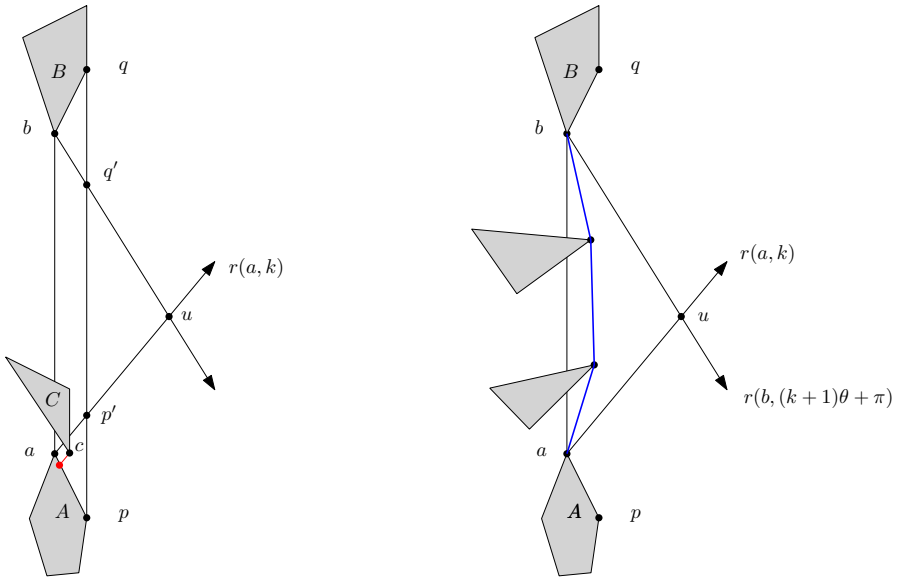

FIGURE 5.9: In the left figure, if an obstacle C overlaps ab and $r(a,k)$, then there must exist a point $c \in C$ lying in $\triangle app'$. There exists a propagated sample point (red) that is closer to p than a is. Otherwise, in the right figure, there exists a convex path P (shown in blue) from a to b, and $\|P\|$ is at most $\|au\| + \|ub\|$.

PROOF. We first observe that if A and B are connected using the trapezoidal map, then both tangent sample points p and q were added by construction. Clearly, $\mathbf{d_S}(p,q) = \|pq\| \leq (1/\cos(\theta)) \cdot \|pq\|$. Therefore, we consider the scenario when A and B are not connected using the trapezoidal map. Without loss of generality, let pq lie on the y-axis with $y(q) > y(p)$, and let pq lie between $r(p,k)$ and $r(p,k+1)$.

If both A and B lie on the same side of the line through pq, we assume that without loss of generality, they lie on the left side (see Figure 5.9). Let a be the closest sample point of p, and let b be the closest sample point of q, such that $y(b) < y(q)$ and $y(a) > y(p)$. Observe that a must be above $r(p,k+1)$, and similarly, b must be below $r(q, k\theta + \pi)$.

Since a is the closest sample point to p and b is the closest sample point of q, we have that $\mathbf{d_S}(p,a) = \|pa\|$ and $\mathbf{d_S}(b,q) = \|bq\|$. If ab overlaps no obstacles, $\mathbf{d_S}(a,b) \leq \|ab\|$.

Since we do not need to take a detour from a to b, it is sufficient to find an upperbound on $\|pa\| + \|ab\| + \|bq\|$. Therefore, we focus on the worse case where ab overlaps at least one obstacle. Let u be the intersection of $r(a, k)$ and $r(b, (k+1)\theta + \pi)$. It remains true that if an obstacle C overlaps ab and $r(a, k)$ simultaneously, there must exist a propagated sample point on A that is closer to p than a is. Therefore the arguments in Lemma 84 apply, and $\mathbf{d_S}(a, b) \leq \|au\| + \|ub\|$.

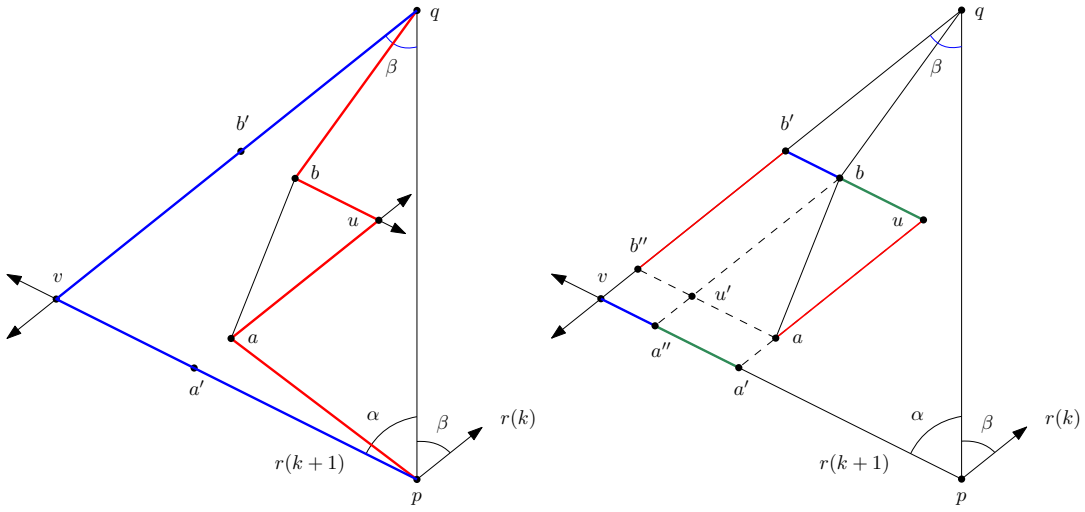

FIGURE 5.10: In the left figure, $\mathbf{d_S}(a, b)$ is maximised when a and b lie on pv and qv, respectively, and $\|pa\| + \mathbf{d_S}(a, b) + \|qb\|$ (red) is at most $\|pv\| + \|qv\|$ (blue). In the right figure, segments with the same colour have equal lengths.

Let v be the intersection of $r(p, k+1)$ and $r(q, k\theta + \pi)$ (see Figure 5.10, left). We argue that $\|pa\| + \|au\| + \|ub\| + \|bq\|$ is maximised when a lies on $r(p, k+1)$ and b lies on $r(q, k\theta + \pi)$. Then, when both a and b lie on their respective rays, we show that the total length $\|pa\| + \|au\| + \|ub\| + \|bq\|$ is at most $\|pv\| + \|qv\|$. To do this, we partition the latter and show that each partition (or two partitions combined) pays for a segment in the former.

By constructing the sample points, a and b must reside in $\triangle pqv$. The segment ab must lie between the directions $r(k)$ and $r(k+1)$, as otherwise, A and B are connected via the trapezoidal map. As a result, $y(b) > y(a)$, and u is to the right of ab. See Figure 5.10, right for the following definitions. Let u' be the intersection of $r(b, k\theta + \pi)$ and $r(a, k+1)$. Observe that $\|bu'\| = \|au\|$ and $\|au'\| = \|bu\|$. Let b' (resp. b'') be the intersection of $r(b, k+1)$

(resp. $r(a, k+1)$) and qv. Both b' and b'' are well-defined, since $r(b, k+1)$, $r(a, k+1)$, and $r(p, k+1)$ are parallel. The points a' and a'' are defined similarly.

Using the triangle inequality, we have that $\|qb\| \leq \|qb'\| + \|bb'\|$. We observe that $\|b'b''\| = \|au\|$ and $\|bb'\| = \|va''\|$. Combining the above facts, if we focus on $\|qb\| + \|bu\|$, we have

$$\begin{aligned} \|qb\| + \|bu\| &\leq \|qb'\| + \|b'b\| + \|bu\| && \triangleright \text{ Triangle inequality} \\ &= \|qb'\| + \|a'v\|. && \triangleright \Diamond b'ua'v \text{ is a parallelogram} \end{aligned} \tag{1}$$

Similarly,

$$\|pa\| + \|au\| \leq \|pa'\| + \|a'a\| + \|au\| \leq \|pa'\| + \|b'v\|. \tag{2}$$

Combining (1) and (2), and the fact that $\mathbf{d_S}(a, b) \leq \|au\| + \|ub\|$, we have the following.

$$\begin{aligned} \mathbf{d_S}(p, a) + \mathbf{d_S}(a, b) + \mathbf{d_S}(b, q) &\leq \|pa\| + \|au\| + \|ub\| + \|bq\| \\ &\leq \|pv\| + \|qv\| \\ &\leq \sec(\frac{\theta}{2}) \cdot \|pq\| && \triangleright \text{ Lemma 80} \end{aligned}$$

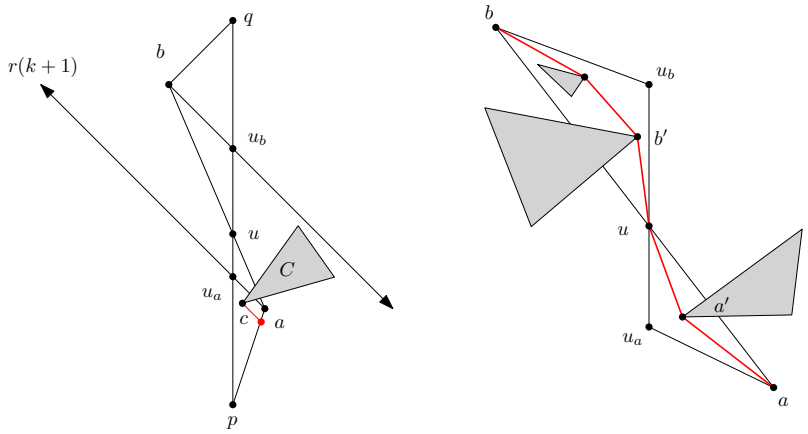

FIGURE 5.11: In the left figure, if an obstacle C overlaps ab and $r(a, k+1)$, then a propagated sample point (red) must be closer to p than a is. In the right figure, two convex paths (red) can be generated in $\triangle auu_a$ and $\triangle buu_b$.

Using analogous arguments, if A and B lie on the opposite sides of pq (see Figure 5.11, left), and an obstacle C overlaps ab, part of C must lie in either triangle $\triangle pau$ or $\triangle qbu$ (not both).

Without loss of generality, assume that C overlaps $\triangle pau$. If C additionally overlaps $r(a, k+1)$, a propagated sample point d is closer to q than b, contradicting the assumption that b is the closest sample point. An analogous argument applies when C overlaps $\triangle qbu$. Therefore if C overlaps ab, then C must reside in the either $\triangle a = \triangle auu_a$ or $\triangle b = \triangle buu_b$, where u_b (resp. u_a) is the intersection of $r(b, (k+1)\theta + \pi)$ (resp. $r(a, k+1)$) and ab.

We can construct two convex paths P_a and P_b (see Figure 5.11, right); P_a resides in $\triangle a$ and connects a to u, and P_b resides in $\triangle b$ and connects b to u. By Lemma 80, we have that

$$\begin{aligned} awdawdasd\mathbf{d_S}(a,b) &\leq \|P_a\| + \|P_b\| \\ &\leq \|au_a\| + \|u_au_b\| + \|u_bb\|. \end{aligned} \tag{1}$$

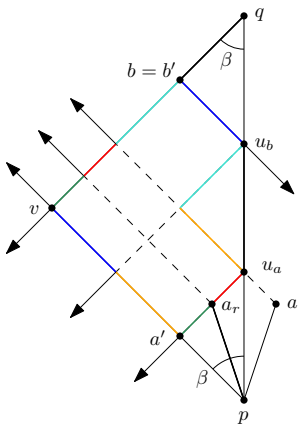

FIGURE 5.12: Segments with matching colour have equal length. Every segment in $\{pa_r, au_a, u_au_b, u_bb\}$ can be paid for by either segment(s) on pv or segment(s) on qv.

We now upperbound (1). Let v be the intersection of $r(q, k\theta + \pi)$ and $r(p, k\theta + 2\beta)$ (see Figure 5.12). Let point a_r be the reflection of a along pq. Observe that since a is above $r(p, k\theta)$, a_r must be above $r(p, k\theta + 2\beta)$, and $\|pa_r\| = \|pa\|$. Let a' (resp. b') be the intersection of $r(a_r, k\theta + \pi)$ (resp. $r(b, k\theta + 2\beta)$ and pv (resp. qv).

Using an analogous charging argument, we have that

$$\|au_a\| + \|u_au_b\| + \|u_bb\| \leq \|pv\| + \|qv\| \leq \frac{1}{\cos(\beta)} \cdot \|pq\|.$$

Combining with the bound on $\mathbf{d_S}(a,b)$ in the case that a and b lie on the same side of pq, we have that

$$\mathbf{d_S}(p,a)+\mathbf{d_S}(a,b)+\mathbf{d_S}(b,q) = \|pa\|+\mathbf{d_S}(a,b)+\|bq\|$$
$$\leq \max\{\sec(\frac{\theta}{2}), \frac{1}{\cos(\beta)}\}\cdot\|pq\| \leq \frac{1}{\cos(\theta)}\cdot\|pq\| \quad \square$$

5.4.3.3 Case 3: pq connects a 0-region and an obstacle

In this section, we prove that if pq connects an obstacle A and a 0-region B, there is a path P that approximates pq.

LEMMA 87. *Let A be a convex obstacle and let B be a convex 0-region. Let $pq \subseteq P^*$, where $p \in \partial A$ and $q \in \partial B$. There exists a pair of sample points $a \in A$ and $b \in B$, such that $\mathbf{d_S}(p,a)+\mathbf{d_S}(a,b)+\mathbf{d_S}(b,q) \leq (\cos(\beta)/\cos(\theta))\cdot\|pq\|$.*

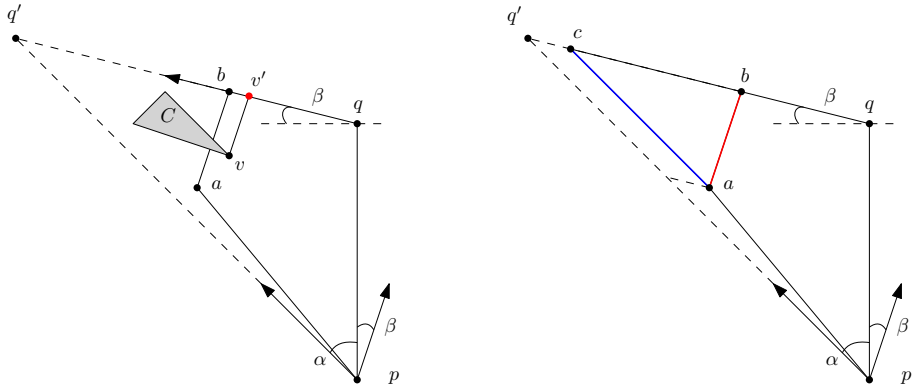

FIGURE 5.13: In the left figure, ab is parallel to the direction $r(k)$. An obstacle C overlapping ab generates a propagated sample point v' (red) closer to q than b. In the right figure, $\|pa\|+\|ab\|$ is at most $\|pq'\|$ since ab (red) is at most as long as ac (blue), and $\|pa\|+\|ac\| \leq \|pq'\|$.

PROOF. Let pq be aligned with the y-axis, such that $y(q) > y(p)$. If pq is in some direction $r(k)$, then both p and q are sample points, and $(p,q) \in \mathcal{E}$. Therefore, we consider the case where pq is between direction $r(k)$ and $r(k+1)$. Let β (resp. α) be the acute angle between $r(p,k)$ (resp. $r(p,k+1)$) and pq (see Figure 5.13).

Let $a \in \partial A$ be the first sample point in a counter-clockwise order after p. Let $b \in \partial B$ be the first sample point in a clockwise order after q. Let q' be the intersection of $r(p,k+1)$

and $r(q, k\theta + \pi/2)$. By convexity and the placement of our sample points, a must be above $r(p, k+1)$, and b must lie below $r(q, k\theta + \pi/2)$.

If ab is parallel with $r(k)$, then ab cannot overlap any obstacle C. Otherwise, there exists a sample point $v \in \partial C$ in the direction of $r(k\theta - \pi/2)$ (see Figure 5.13, left). Since v must lie on or to the left of pq, $r(v, k)$ generates a propagated sample point v' on $\partial B(b, q)$, and v' is closer to q than b is, contradicting the assumption that b is the closest sample point. Therefore, $\mathbf{d_S}(p, a) + \mathbf{d_S}(a, b) \leq \|pa\| + \|ab\|$.

To bound $\|pa\|+\|ab\|$, we observe that $\|pa\|+\|ab\|$ is maximised when b lies on $r(q, k\theta+\pi/2)$ (see Figure 5.13, right). Let c be the intersection of $r(a, k+1)$ and $r(q, k\theta + \pi/2)$. Observe that ab is perpendicular to pq'. We have that $\|ab\| \leq \|ac\|$, since ab is a leg of the right triangle $\triangle abc$, and ac is the hypotenuse. Using the arguments in Lemma 85, $\|pa\| + \|ac\| \leq \|pq'\| = (\cos(\beta)/\cos(\theta)) \cdot \|pq\|$. We thus have a path from p to q via a and b of length

$$\|pa\| + \|ab\| + \mathbf{d_S}(b, q) = \|pa\| + \|ab\| \leq \|pa\| + \|ac\| \leq \|pq'\| \leq \frac{\cos(\beta)}{\cos(\theta)} \cdot \|pq\|.$$

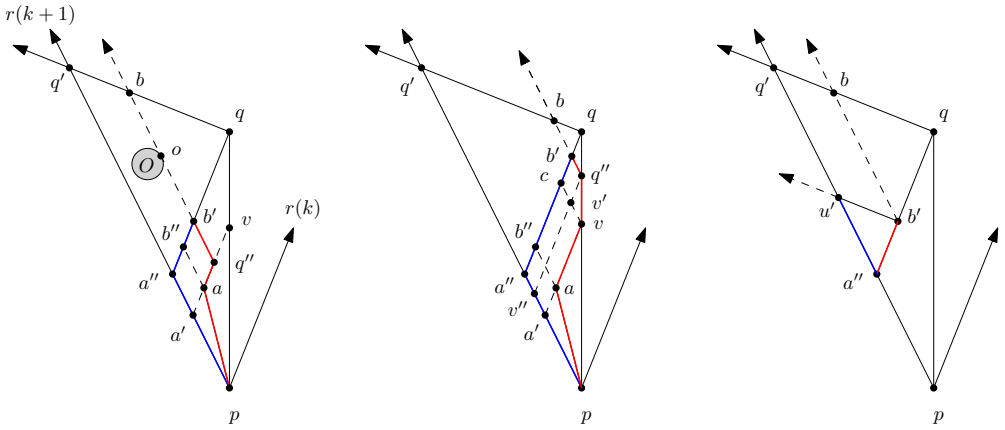

FIGURE 5.14: In the left and middle figure, the length of both red paths are upper-bounded by the length of the blue path. In the right figure, $\|a''u'\| > \|a''b'\|$ by construction.

If ab is not parallel with $r(k+1)$, the ray $r(a, k)$ must intersect pq at some point v (a does not propagate to b). For the same reasons as before, an obstacle overlaps neither $r(a, k)$ nor pq. Let $q'' = q$. Consider sliding q'' towards a along the concatenated polygonal chain $qv \circ va$ until the ray $r(q'', k+1)$ touches b (see Figure 5.14). Observe that $r(q'', k+1)$ cannot intersect

an obstacle during the slide. Indeed, if $r(q'', k+1)$ touches an obstacle O at point o, o must be a sample point with respect to the direction $r((k+1)\theta - \pi/2)$, and o propagates to b with the ray $r(o, k+1)$. Let a' (resp. a'') be the point on pu such that $a'v$ (resp. $a''q$) is parallel to $r(k)$. Let b' be the intersection of $r(q'', k+1)$ and $a''q$. Let b'' be the point on $a''q$ such that ab'' is parallel to $r(k+1)$.

When we stop the sliding, either 1) q'' is on av, or 2) q'' is on qv (see Figure 5.14, left and middle, respectively). In order to avoid obstacles, we have to take a detour from a to b. In both cases, we argue that the length of the detour is at most $\|pa''\| + \|a''b'\| + \|b'b\|$. In case 1, the maximum length of the detour we must take is $d_1 = \|pa\| + \|aq''\| + \|q''b\|$. We use a charging argument: $\|b'b''\|$ pays for $\|aq''\|$, and $\|a'a''\|$ pays for $\|b'q''\|$. $\|aa'\| = \|b''a''\|$, and by triangle inequality, $\|pa\| \leq \|pa'\| + \|a'a\|$. Therefore, $d_1 \leq \|pa''\| + \|a''b'\| + \|b'b\|$.

In case 2, the maximum length of the detour is $d_2 = \|pa\| + \|av\| + \|vq''\| + \|q''b\|$. Let v'' be the point on pu such that $v''q''$ is parallel to $r(k)$. Let v' be the intersection of $r(v, k+1)$ and $v''q''$. Let c be the point on $b'b''$ such that vc is parallel to $r(k+1)$. Using analogous argument, $\|a''v''\|$ pays for $\|b'q''\|$, and $\|b''v''\|$ pays for $\|av\|$. By triangle inequality, $\|a''b''\| + \|pa'\|$ pays for $\|ap\|$, and $\|b'v''\| + \|a'v''\|$ pays for $\|vq''\|$. For both cases, we have that

$$\|pa\| + \mathbf{d_S}(a, b) \leq \|pa''\| + \|a''b'\| + \|b'b\|. \tag{1}$$

Next, we argue that $\|pa''\| + \|a''b'\| + \|b'b\|$ is at most $\|pu\|$ (see Figure 5.14, right). Let u' be the point on pu such that $b'u'$ is parallel to qu. By construction, pu is parallel to $r(k\theta + \pi/2)$, and $a''b'$ is parallel to $r(k\theta)$. The triangle $\triangle b'u'a''$ is therefore a right triangle with $a''u'$ as the hypotenuse. $\|bb'\| = \|uu'\|$, and therefore we have that

$$\|pa''\| + \|a''b'\| + \|b'b\| < \|pa''\| + \|a''u'\| + \|u'u\| = \|pu\|. \tag{2}$$

Combining (1), (2), Lemma 74, and the fact that bq lies in a 0-region, we complete the proof with the following.

$$\mathbf{d_S}(p, a) + \mathbf{d_S}(a, b) + \mathbf{d_S}(b, q) = \|pa\| + \mathbf{d_S}(a, b) \leq \|pu\| \leq \frac{\cos(\beta)}{\cos(\theta)} \cdot \|pq\| \qquad \square$$

5.4.4 The quality of the path

In Lemmas 85, 86, and 87, we have shown that for every segment pq in Case 1-3 in Fact 82, either there exists a path $P \subseteq \mathcal{E}$ such that $\mathbf{w}(P)$ approximates $\|pq\|$, or there exist two sample points a and b such that $\mathbf{d_S}(p,a) + \mathbf{d_S}(a,b) + \mathbf{d_S}(b,q)$ approximates $\|pq\|$. Taking the maximum ratio in the three lemmas, we have the following.

$$\begin{aligned}\mathbf{d_S}(p,a) + \mathbf{d_S}(a,b) + \mathbf{d_S}(b,q) &\leq \max\{\frac{\cos(\beta)}{\cos(\theta)}, \frac{1}{\cos(\theta)}, \frac{\sin(\alpha)+\sin(\beta)}{\cos(\theta)\sin(\theta)}\} \cdot \|pq\| \\ &\leq \frac{2\sin(\frac{\theta}{2})}{\cos(\theta)\sin(\theta)} \cdot \|pq\|\end{aligned}$$

For a Case 4 segment $\partial A(p,q)$, where both p and q lie on the obstacle A, assume without loss of generality that p occurs before q in P^*, and let $pq = \partial A(p,q)$.

If $\partial A(p,q)$ contains no sample point, then assume that the optimal path uses segment $p'p$ to reach A, and qq' to leave A. We argue that there exists an approximate path P that approximates $\|p'p\| + \|\partial A(p,q)\| + \|qq'\|$. Let a (resp. b) be the closest sample point to p (resp. q), such that $\partial A(p,q) \subseteq \partial A(a,b)$. In Lemmas 85, 86, and 87, we have paid for a path $P_p \subseteq \mathcal{E}$ from p' to p through a and a path $P_q \subseteq \mathcal{E}$ from q to q' through b. Since there is no sample point on $\partial A(p,q)$, instead of going from a to p and q to b, we take the path ab directly. The unused cost of $\mathbf{d_S}(a,p)$ and $\mathbf{d_S}(q,b)$ pays for $\|ab\|$.

Let $p_\perp$ (resp. $q_\perp$) be the orthogonal projection of p (resp. q) on ab. Clearly, $\|\partial A(a,p)\| \geq \|ap_\perp\|$ and $\|\partial A(q,b)\| \geq \|bq_\perp\|$. By Lemma 80, $\|\partial A(p,q)\| \leq \sec(\theta/2) \cdot \|p_\perp q_\perp\|$. Therefore, we connect P_p and P_q using ab to generate a path P, and we have that

$$\|P\| \leq \frac{2\sin(\frac{\theta}{2})}{\cos(\theta)\sin(\theta)} \cdot (\|p'p\| + \|\partial A(p,q)\| + \|qq'\|).$$

If $\partial A(p,q)$ contains at least one sample point $\{a, \ldots, b\}$, then by Lemma 80, we have that

$$\mathbf{d_S}(p,a) + \mathbf{d_S}(a,b) + \mathbf{d_S}(b,q) \leq \sec(\frac{\theta}{2}) \cdot \|pq\|.$$

Bose and van Renssen [20] showed that in an environment with polygonal obstacles, a Θ-graph has a spanning ratio of at most $r_\theta = 1 + 2\sin(\theta/2)/(\cos(\theta/2) - \sin(\theta/2))$. We also need to apply the factor to traverse the boundaries of convex obstacles to account for the difference compared to the boundaries of simplified obstacles, as in Lemma 80. We obtain the following bound when $\theta < \pi/12$.

$$\frac{2\sin(\frac{\theta}{2})}{\cos(\theta)\sin(\theta)} \cdot (1 + \frac{2\sin(\frac{\theta}{2})}{\cos(\frac{\theta}{2}) - \sin(\frac{\theta}{2})}) \cdot \frac{2\sin(\frac{\theta}{2})}{\sin(\theta)} \leq \frac{1}{1 - \sin(2\theta)}$$

Given an approximation error $0 < \varepsilon < 1$, we compute the parameter $0 < \theta < \pi/12$ as the following.

$$\frac{1}{1 - \sin(2\theta)} \leq 1 + \varepsilon \implies \theta \leq \frac{\sin^{-1}(\frac{\varepsilon}{1+\varepsilon})}{2}$$

LEMMA 88. *In $\mathcal{B}$, there exists a path $P \subseteq \mathcal{E}$ between any pair of sample points (a, b) such that $\mathbf{w}(P) \leq (1 + \varepsilon) \cdot \mathbf{d}(a, b)$.*

5.4.5 Finding a shortest path amidst 0-regions and obstacles

Given the data structure $\mathcal{B} = \{\mathcal{M}(k) \mid \forall k \in [0, 2\pi/\theta), k \in \mathbb{Z}\} \cup \{\mathcal{G}, \mathcal{G}_\Theta\}$, a point s, and a point t, we query the approximate shortest path from s to t using Algorithm 4.

Using the query algorithm, we treat both s and t as convex obstacles with no interior. This preserves the properties of the trapezoidal maps and the Θ-graph, and enables us to apply the earlier lemmas.

5.4.5.1 Analysis

We now analyse the query time. In Step 2, we perform a set of operations for each of the $\mathcal{O}(1/\varepsilon)$ trapezoidal maps. In Step 2a, it takes $\mathcal{O}(\log(n/\varepsilon))$ time to find the face containing s. In Step 2b, it takes $\mathcal{O}(\log N)$ time to use a binary search to compute the intersection of a ray and a convex boundary with $\mathcal{O}(N)$ complexity. In Step 2c, it takes $\mathcal{O}(\log N)$ time to compute the distance between s and a convex region using the algorithm by Edelsbrunner [69]. In Step 2d, it takes $\mathcal{O}(\log N)$ time to compute the common tangent using the algorithms by

Algorithm 4 Query s-t weighted shortest path amidst 0-regions and obstacles

This algorithm takes as input a data structure $\mathcal{B} = \{\mathcal{M}(k) \mid \forall k \in [0, 2\pi/\theta), k \in \mathbb{Z}\} \cup \{\mathcal{G}, \mathcal{G}_\Theta\}$ storing a set of 0-regions and a set of obstacles, a point s, and a point t. It outputs a $(1+\varepsilon)$-approximate weighted shortest path from s to t. In Step 2 and Step 3, this algorithm shows how to add s to $\mathcal{B}$, and the same operations are used to add t.

(1) Add s and t to $\mathcal{V}$.
(2) For each trapezoidal map $\mathcal{M}(k)$, do the following for point s.
  (a) Query the face F containing s. Let F be adjacent to A and B, $A \neq B$.
  (b) Add the propagated sample points $a \in A$ and $b \in B$—which are generated by $r(s,k)$ and $r(s, k\theta+\pi)$, respectively—to $\mathcal{V}$.
  (c) If A is a 0-region, add $e = (s, ak(A))$ and set $\mathbf{w}(e) = \mathbf{d}(s, A)$. Do the same for B.
  (d) If A is an obstacle, add $e = (a, s)$, and set $\mathbf{w}(e) = \|as\|$. Compute the common tangents T of s and A. For each common tangent $t \in T$ touching A at point a', add a' to $\mathcal{V}$. Do the same for B.
(3) Add s and the additional sample points generated in Step 2 to $\mathcal{G}_\Theta$. For each newly added point s', using s' as the apex, we construct a set of disjoint cones with angle θ. For each point p closest to s' in each cone, add $e = (s', p)$ to $\mathcal{E}$, and set $\mathbf{w}(e) = \|s'p\|$. For every existing vertex $p \in \mathcal{G}_\Theta$, and every existing edge $(p, q) \in \mathcal{E}_\Theta$. If s' is closer to p than q is, add $e = (s', p)$ to $\mathcal{E}$, and set $\mathbf{w}(e) = \|s'p\|$.
(4) Use Dijkstra's shortest path algorithm to compute a path P' from s to t in $\mathcal{G}$. Transform P' into a path P in the original environment and return P.

Kirkpatrick and Snoeyink [116], and Guibas et al. [98]. In total, Step 2 takes $\mathcal{O}((\log(n/\varepsilon) + \log N)/\varepsilon)$ time.

Inserting s into $\mathcal{B}$ generates a constant number of sample points. In Step 3, for each additional sample point s', it takes $\mathcal{O}(n/\varepsilon^3)$ time to find the closest point of s' in each cone, and at the same time, check if s' is closest to any point p. In Step 4, it takes $\mathcal{O}(|\mathcal{E}| + |\mathcal{V}| \log |\mathcal{V}|)$ to run Dijkstra's shortest path algorithm. There are $\mathcal{O}(n/\varepsilon^2)$ vertices, and $\mathcal{O}(n/\varepsilon^3)$ edges, therefore Dijkstra's algorithm takes $\mathcal{O}(n/\varepsilon^3 + (n/\varepsilon^2)\log(n/\varepsilon))$ time to return a path P' comprised of at most $\mathcal{O}(n/\varepsilon^3)$ edges. It takes $\mathcal{O}(n/\varepsilon^3 + N)$ time to transform P' into a path P in the environment by traversing the boundaries of the regions.

In total, it takes $\mathcal{O}(N + n/\varepsilon^3 + (n/\varepsilon^2)\log(n/\varepsilon) + (\log N)/\varepsilon)$ time to query the approximate s-t shortest path. Combining the above analysis with Lemmas 79, 85, 86, and 87, we have the following.

THEOREM 16. *Consider a planar subdivision induced by a plane with a weight of* 1*, consisting of two sets of convex and non-overlapping regions: 0-regions* $\mathcal{Z}$ *with a weight of* 0*, and obstacles* $\mathcal{O}$ *with a weight of* ∞*. Let* $n = |\mathcal{Z}| + |\mathcal{O}|$ *and let* N *denote the total number of vertices in* $\mathcal{Z} \cup \mathcal{O}$*. For any approximation factor* $0 < \varepsilon < 1$*, a data structure* $\mathcal{B}$ *can be constructed over* $\mathcal{Z} \cup \mathcal{O}$ *in* $\mathcal{O}(N + (n/\varepsilon^3)(\log(n/\varepsilon) + \log N))$ *expected time, with a total size of* $\mathcal{O}(N + n/\varepsilon^3)$*. When queried with arbitrary points* s *and* t*,* $\mathcal{B}$ *returns a weighted path* P *from* s *to* t *in* $\mathcal{O}(N + n/\varepsilon^3 + (n/\varepsilon^2)\log(n/\varepsilon) + (\log N)/\varepsilon)$ *time, ensuring that* $\mathbf{w}(P) \leq (1+\varepsilon) \cdot \mathbf{w}(P^*)$*, where* P^* *is the optimal weighted shortest path from* s *to* t*.*

5.5 Conclusion and future work

The main contribution of this chapter is a method for approximating the shortest path between two 0-regions when those regions are close. An interesting open question is whether the $0/1$-weighted region problem can be solved exactly in near-linear time. This is already achieved by [84] when the visibility graph of the polygonal 0-regions has near-linear complexity; the most general case, however, remains open.

Bibliography

[1] Peyman Afshani and Anne Driemel. 'On the complexity of range searching among curves'. In: *Proc. 29th ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Society for Industrial and Applied Mathematics, Jan. 2018, pp. 898–917. DOI: 10.1137/1.9781611975031.58.

[2] Pankaj K. Agarwal et al. 'Subtrajectory clustering: models and algorithms'. In: *Proc. 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS)*. New York, NY, USA: Association for Computing Machinery, May 2018, pp. 75–87. ISBN: 978-1-4503-4706-8. DOI: 10.1145/3196959.3196972.

[3] Sepideh Aghamolaei et al. 'Windowing queries using Minkowski sum and their extension to MapReduce'. In: *The Journal of Supercomputing* 77.1 (Jan. 2021), pp. 936–972. ISSN: 1573-0484. DOI: 10.1007/s11227-020-03299-7.

[4] Lyudmil Aleksandrov, Anil Maheshwari and Jörg-Rüdiger Sack. 'Approximation algorithms for geometric shortest path problems'. In: *Proc. 32nd Annual ACM Symposium on Theory of Computing (STOC)*. New York, NY, USA: Association for Computing Machinery, May 2000, pp. 286–295. ISBN: 978-1-58113-184-0. DOI: 10.1145/335305.335339.

[5] Lyudmil Aleksandrov, Anil Maheshwari and Jörg-Rüdiger Sack. 'Determining approximate shortest paths on weighted polyhedral surfaces'. In: *Journal of the ACM* 52.1 (Jan. 2005), pp. 25–53. ISSN: 0004-5411. DOI: 10.1145/1044731.1044733.

[6] Helmut Alt. 'The Computational Geometry of Comparing Shapes'. In: *Efficient Algorithms, Essays Dedicated to Kurt Mehlhorn on the Occasion of His 60th Birthday*. Ed. by Susanne Albers, Helmut Alt and Stefan Näher. Vol. 5760. Lecture Notes in Computer Science. Springer, 2009, pp. 235–248. DOI: 10.1007/978-3-642-03456-5_16.

[7] Helmut Alt and Michael Godau. ‘Computing the Fréchet distance between two polygonal curves’. In: *International Journal of Computational Geometry and Applications* 05 (Mar. 1995), pp. 75–91. ISSN: 0218-1959. DOI: 10.1142/S0218195995000064.

[8] Helmut Alt and Leonidas J. Guibas. ‘Discrete Geometric Shapes: Matching, Interpolation, and Approximation’. In: *Handbook of Computational Geometry*. Ed. by Jörg-Rüdiger Sack and Jorge Urrutia. North Holland / Elsevier, 2000. Chap. 3, pp. 121–153. DOI: 10.1016/B978-044482537-7/50004-8.

[9] Helmut Alt, Christian Knauer and Carola Wenk. ‘Matching Polygonal Curves with Respect to the Fréchet Distance’. In: *Proc. 18th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*. Ed. by Afonso Ferreira and Horst Reichel. Berlin, Heidelberg: Springer, 2001, pp. 63–74. ISBN: 978-3-540-44693-4. DOI: 10.1007/3-540-44693-1_6.

[10] Helmut Alt, Ludmila Scharf and Sven Scholz. ‘Probabilistic matching and resemblance evaluation of shapes in trademark images’. In: *Proc. 6th ACM International Conference on Image and Video Retrieval (CIVR)*. Ed. by Nicu Sebe and Marcel Worring. ACM, 2007, pp. 533–540. DOI: 10.1145/1282280.1282357.

[11] Boris Aronov et al. ‘Discrete Fréchet Distance Oracles’. In: *Proc. 40th International Symposium on Computational Geometry (SoCG)*. Ed. by Wolfgang Mulzer and Jeff M. Phillips. Vol. 293. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024, 10:1–10:14. ISBN: 978-3-95977-316-4. DOI: 10.4230/LIPIcs.SoCG.2024.10.

[12] Rinat Ben Avraham, Haim Kaplan and Micha Sharir. *A faster algorithm for the discrete Fréchet distance under translation*. 2015. DOI: 10.48550/arXiv.1501.03724. arXiv: 1501.03724 [cs.CG].

[13] Samet Ayhan and Hanan Samet. ‘Aircraft Trajectory Prediction Made Easy with Predictive Analytics’. In: *Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2016, pp. 21–30. DOI: 10.1145/2939672.2939694.

[14] Julian Baldus and Karl Bringmann. ‘A fast implementation of near neighbors queries for Fréchet distance (GIS Cup)’. In: *Proc. 25th ACM SIGSPATIAL International*

Conference on Advances in Geographic Information Systems (SIGSPATIAL). New York, NY, USA: Association for Computing Machinery, Nov. 2017, pp. 1–4. ISBN: 978-1-4503-5490-5. DOI: 10.1145/3139958.3140062.

[15] Mark de Berg et al. *Computational Geometry: Algorithms and Applications*. Berlin, Heidelberg: Springer, 2008. ISBN: 978-3-540-77973-5 978-3-540-77974-2. DOI: 10.1007/978-3-540-77974-2.

[16] Mark de Berg et al. 'Realistic input models for geometric algorithms'. In: *Proc. 13th Annual Symposium on Computational Geometry (SoCG)*. New York, NY, USA: Association for Computing Machinery, Aug. 1997, pp. 294–303. ISBN: 978-0-89791-878-7. DOI: 10.1145/262839.262986.

[17] Sarita de Berg et al. *Exact solutions to the Weighted Region Problem*. arXiv:2402.12028 [cs]. Feb. 2024. DOI: 10.48550/arXiv.2402.12028.

[18] Lotte Blank and Anne Driemel. 'A Faster Algorithm for the Fréchet Distance in 1D for the Imbalanced Case'. In: *Proc. 32nd Annual European Symposium on Algorithms (ESA)*. Ed. by Timothy M. Chan et al. Vol. 308. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024, 28:1–28:15. DOI: 10.4230/LIPIcs.ESA.2024.28.

[19] Lotte Blank et al. 'Transforming Dogs on the Line: On the Fréchet Distance Under Translation or Scaling in 1D'. In: *Proc. 41st International Symposium on Computational Geometry (SoCG)*. arXiv:2501.12821. 2025. DOI: 10.48550/arXiv.2501.12821.

[20] Prosenjit Bose and André van Renssen. 'Spanning properties of Yao and theta-graphs in the presence of constraints'. In: *International Journal of Computational Geometry and Applications* 29.02 (June 2019), pp. 95–120. ISSN: 0218-1959. DOI: 10.1142/S021819591950002X.

[21] Prosenjit Bose et al. 'On approximating shortest paths in weighted triangular tessellations'. In: *Artificial Intelligence* 318 (May 2023), p. 103898. ISSN: 0004-3702. DOI: 10.1016/j.artint.2023.103898.

[22] Prosenjit Bose et al. ‘Towards tight bounds on theta-graphs: more is not always better’. In: *Theoretical Computer Science* 616 (Feb. 2016), pp. 70–93. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2015.12.017.

[23] Karl Bringmann. ‘Why walking the dog takes time: Fréchet distance has no strongly subquadratic algorithms unless SETH fails’. In: *Proc. 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. Oct. 2014, pp. 661–670. DOI: 10.1109/FOCS.2014.76.

[24] Karl Bringmann and Bhaskar Ray Chaudhury. ‘Polyline simplification has cubic complexity’. In: *Journal of Computational Geometry* 11.2 (Jan. 2021). Number: 2, pp. 94–130. ISSN: 1920-180X. DOI: 10.20382/jocg.v11i2a5.

[25] Karl Bringmann and Marvin Künnemann. ‘Improved approximation for Fréchet distance on c-packed curves matching conditional lower bounds’. In: *International Journal of Computational Geometry and Applications* 27 (Mar. 2017), pp. 85–119. ISSN: 0218-1959. DOI: 10.1142/S0218195917600056.

[26] Karl Bringmann, Marvin Künnemann and André Nusser. ‘Discrete Fréchet Distance under Translation: Conditional Hardness and an Improved Algorithm’. In: *ACM Transactions on Algorithms* 17.3 (2021), 25:1–25:42. DOI: 10.1145/3460656.

[27] Karl Bringmann, Marvin Künnemann and André Nusser. ‘Walking the dog fast in practice: Algorithm engineering of the Fréchet distance’. In: *Journal of Computational Geometry* 12.1 (Aug. 2021). Number: 1, pp. 70–108. ISSN: 1920-180X. DOI: 10.20382/jocg.v12i1a4.

[28] Karl Bringmann, Marvin Künnemann and André Nusser. ‘When Lipschitz Walks Your Dog: Algorithm Engineering of the Discrete Fréchet Distance Under Translation’. In: *Proc. 28th Annual European Symposium on Algorithms (ESA)*. Ed. by Fabrizio Grandoni, Grzegorz Herman and Peter Sanders. Vol. 173. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020, 25:1–25:17. ISBN: 978-3-95977-162-7. DOI: 10.4230/LIPIcs.ESA.2020.25.

[29] Karl Bringmann et al. ‘Tight Bounds for Approximate Near Neighbor Searching for Time Series under the Fréchet Distance’. In: *Proc. 33rd ACM-SIAM Symposium on*

Discrete Algorithms (SODA). Society for Industrial and Applied Mathematics, Jan. 2022, pp. 517–550. DOI: 10.1137/1.9781611977073.25.

[30] Frederik Brüning, Jacobus Conradi and Anne Driemel. 'Faster approximate covering of subcurves under the Fréchet distance'. In: *Proc. 30th Annual European Symposium on Algorithms (ESA)*. Ed. by Shiri Chechik et al. Vol. 244. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, 28:1–28:16. ISBN: 978-3-95977-247-1. DOI: 10.4230/LIPIcs.ESA.2022.28.

[31] Kevin Buchin, Maike Buchin and Yusu Wang. 'Exact algorithms for partial curve matching via the Fréchet distance'. In: *Proc. 20th ACM-SIAM Symposium on Discrete Algorithms (SODA)*. USA: Society for Industrial and Applied Mathematics, Jan. 2009, pp. 645–654. DOI: 10.5555/1496770.1496841.

[32] Kevin Buchin, Tim Ophelders and Bettina Speckmann. 'SETH Says: Weak Fréchet Distance is Faster, but only if it is Continuous and in One Dimension'. In: *Proc. 30th ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Ed. by Timothy M. Chan. SIAM, 2019, pp. 2887–2901. DOI: 10.1137/1.9781611975482.179.

[33] Kevin Buchin et al. 'Approximating (k, ℓ)-center clustering for curves'. In: *Proc. 30th ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Ed. by Timothy M. Chan. SIAM, 2019, pp. 2922–2938. DOI: 10.1137/1.9781611975482.181.

[34] Kevin Buchin et al. 'Clustering trajectories for map construction'. In: *Proc. 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL)*. New York, NY, USA: Association for Computing Machinery, Nov. 2017, pp. 1–10. ISBN: 978-1-4503-5490-5. DOI: 10.1145/3139958.3139964.

[35] Kevin Buchin et al. 'Computing the Fréchet distance between uncertain curves in one dimension'. In: *Computational Geometry* 109 (Feb. 2023), p. 101923. ISSN: 0925-7721. DOI: 10.1016/j.comgeo.2022.101923.

[36] Kevin Buchin et al. 'Detecting commuting patterns by clustering subtrajectories'. In: *International Journal of Computational Geometry and Applications* 21.03 (June 2011), pp. 253–282. ISSN: 0218-1959. DOI: 10.1142/S0218195911003652.

[37] Kevin Buchin et al. ‘Efficient trajectory queries under the Fréchet distance (GIS Cup)’. In: *Proc. 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL)*. New York, NY, USA: Association for Computing Machinery, Nov. 2017, pp. 1–4. ISBN: 978-1-4503-5490-5. DOI: 10.1145/3139958.3140064.

[38] Kevin Buchin et al. ‘Fréchet Distance for Uncertain Curves’. In: *ACM Transactions on Algorithms* 19.3 (July 2023), 29:1–29:47. ISSN: 1549-6325. DOI: 10.1145/3597640.

[39] Kevin Buchin et al. ‘klcluster: Center-based Clustering of Trajectories’. In: *Proc. 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL)*. New York, NY, USA: Association for Computing Machinery, Nov. 2019, pp. 496–499. ISBN: 978-1-4503-6909-1. DOI: 10.1145/3347146.3359111.

[40] Kevin Buchin et al. ‘Map matching queries under Fréchet Distance on low-density spanners’. In: *European Workshop on Computational Geometry (EuroCG)*. 2023.

[41] Kevin Buchin et al. ‘Map-Matching Queries Under Fréchet Distance on Low-Density Spanners’. In: *Proc. 40th International Symposium on Computational Geometry (SoCG)*. Ed. by Wolfgang Mulzer and Jeff M. Phillips. Vol. 293. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024, 27:1–27:15. DOI: 10.4230/LIPIcs.SoCG.2024.27.

[42] Kevin Buchin et al. ‘On Length-Sensitive Fréchet Similarity’. In: *Proc. 18th International Symposium on Algorithms and Data Structures (WADS)*. Berlin, Heidelberg: Springer-Verlag, July 2023, pp. 208–231. ISBN: 978-3-031-38905-4. DOI: 10.1007/978-3-031-38906-1_15.

[43] Maike Buchin, Anne Driemel and Dennis Rohde. ‘Approximating (k,ℓ)-Median Clustering for Polygonal Curves’. In: *ACM Transactions on Algorithms* 19.1 (Feb. 2023), 4:1–4:32. ISSN: 1549-6325. DOI: 10.1145/3559764.

[44] Maike Buchin and Lukas Plätz. *The k-outlier Fréchet distance*. 2022. DOI: 10.48550/arXiv.2202.12824. arXiv: 2202.12824 [cs.CG].

[45] Maike Buchin et al. ‘Efficient Fréchet Distance Queries for Segments’. In: *Proc. 30th Annual European Symposium on Algorithms (ESA)*. Ed. by Shiri Chechik et al. Vol. 244. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, 29:1–29:14. ISBN: 978-3-95977-247-1. DOI: 10.4230/LIPIcs.ESA.2022.29.

[46] Maike Buchin et al. ‘Segmenting Trajectories: A Framework and Algorithms Using Spatiotemporal Criteria’. In: *Journal of Spatial Information Science* 3 (2011), pp. 33–63. DOI: 10.5311/JOSIS.2011.3.66.

[47] Jean-Lou de Carufel et al. ‘A note on the unsolvability of the weighted region shortest path problem’. In: *Computational Geometry* 47.7 (Aug. 2014), pp. 724–727. ISSN: 0925-7721. DOI: 10.1016/j.comgeo.2014.02.004.

[48] Jean-Lou de Carufel et al. ‘Similarity of polygonal curves in the presence of outliers’. In: *Computational Geometry* 47.5 (July 2014), pp. 625–641. ISSN: 0925-7721. DOI: 10.1016/j.comgeo.2014.01.002.

[49] Claudia Cavallaro et al. ‘Measuring the impact of COVID-19 restrictions on mobility: A real case study from Italy’. In: *Journal of Communications and Networks* 23.5 (Oct. 2021), pp. 340–349. ISSN: 1976-5541. DOI: 10.23919/JCN.2021.000034.

[50] Timothy M. Chan, Kasper Green Larsen and Mihai Pătraşcu. ‘Orthogonal range searching on the RAM, revisited’. In: *Proc. 27th Annual Symposium on Computational Geometry (SoCG)*. New York, NY, USA: Association for Computing Machinery, June 2011, pp. 1–10. ISBN: 978-1-4503-0682-9. DOI: 10.1145/1998196.1998198.

[51] Cheng Chang and Baoyao Zhou. ‘Multi-granularity visualization of trajectory clusters using sub-trajectory clustering’. In: *Proc. 2009 IEEE International Conference on Data Mining Workshops (ICDMW)*. Dec. 2009, pp. 577–582. DOI: 10.1109/ICDMW.2009.24.

[52] Daniel Chen et al. ‘Approximate map matching with respect to the Fréchet distance’. In: *Proc. 2011 Workshop on Algorithm Engineering and Experiments (ALENEX)*. Society for Industrial and Applied Mathematics, Jan. 2011, pp. 75–83. DOI: 10.1137/1.9781611972917.8.

[53] Siu-Wing Cheng and Haoqiang Huang. 'Approximate Nearest Neighbor for Polygonal Curves Under Fréchet Distance'. In: *Proc. 50th International Colloquium on Automata, Languages, and Programming (ICALP)*. Ed. by Kousha Etessami, Uriel Feige and Gabriele Puppis. Vol. 261. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023, 40:1–40:18. ISBN: 978-3-95977-278-5. DOI: 10.4230/LIPIcs.ICALP.2023.40.

[54] Siu-Wing Cheng and Haoqiang Huang. 'Curve Simplification and Clustering under Fréchet Distance'. In: *Proc. 34th ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Society for Industrial and Applied Mathematics, Jan. 2023, pp. 1414–1432. DOI: 10.1137/1.9781611977554.ch51.

[55] Siu-Wing Cheng and Haoqiang Huang. 'Fréchet Distance in Subquadratic Time'. In: *Proc. 36th ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Society for Industrial and Applied Mathematics, Jan. 2025, pp. 5100–5113. DOI: 10.1137/1.9781611978322.173.

[56] Siu-Wing Cheng and Haoqiang Huang. 'Solving Fréchet Distance Problems by Algebraic Geometric Methods'. In: *Proc. 35th ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Society for Industrial and Applied Mathematics, Jan. 2024, pp. 4502–4513. DOI: 10.1137/1.9781611977912.158.

[57] Siu-Wing Cheng, Haoqiang Huang and Shuo Zhang. 'Constant Approximation of Fréchet Distance in Strongly Subquadratic Time'. In: *Proc. 57th Annual ACM Symposium on Theory of Computing (STOC)*. New York, NY, USA: Association for Computing Machinery, June 2025, pp. 2329–2340. ISBN: 979-8-4007-1510-5. DOI: 10.1145/3717823.3718157.

[58] Siu-Wing Cheng, Jiongxin Jin and Antoine Vigneron. 'Triangulation Refinement and Approximate Shortest Paths in Weighted Regions'. In: *Proc. 26th ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Society for Industrial and Applied Mathematics, Dec. 2014, pp. 1626–1640. ISBN: 978-1-61197-374-7. DOI: 10.1137/1.9781611973730.108.

[59] Kenneth L. Clarkson. 'Approximation algorithms for shortest path motion planning'. In: *Proc. 19th Annual ACM Symposium on Theory of Computing (STOC)*. New York,

NY, USA: Association for Computing Machinery, Jan. 1987, pp. 56–65. ISBN: 978-0-89791-221-1. DOI: 10.1145/28395.28402.

[60] Kenneth L. Clarkson and Peter W. Shor. 'Applications of Random Sampling in Computational Geometry, II'. In: *Discrete and Computational Geometry* 4.5 (1989), pp. 387–421. DOI: 10.1007/BF02187740.

[61] Connor Colombe and Kyle Fox. 'Approximating the (Continuous) Fréchet Distance'. In: *Proc. 37th International Symposium on Computational Geometry (SoCG)*. Ed. by Kevin Buchin and Éric Colin de Verdière. Vol. 189. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021, 26:1–26:14. ISBN: 978-3-95977-184-9. DOI: 10.4230/LIPIcs.SoCG.2021.26.

[62] Jacobus Conradi and Anne Driemel. 'On computing the k-shortcut Fréchet distance'. In: *Proc. 49th International Colloquium on Automata, Languages, and Programming (ICALP)*. Ed. by Mikołaj Bojańczyk, Emanuela Merelli and David P. Woodruff. Vol. 229. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, 46:1–46:20. ISBN: 978-3-95977-235-8. DOI: 10.4230/LIPIcs.ICALP.2022.46.

[63] Eduardo Davalos et al. '3D Gaze Tracking for Studying Collaborative Interactions in Mixed-Reality Environments'. In: *Companion Proc. 26th International Conference on Multimodal Interaction (ICMI)*. New York, NY, USA: Association for Computing Machinery, Nov. 2024, pp. 175–183. ISBN: 9798400704635. DOI: 10.1145/3686215.3688380.

[64] Krzysztof Diks and Piotr Sankowski. 'Dynamic Plane Transitive Closure'. In: *Proc. 15th Annual European Symposium on Algorithms (ESA)*. Ed. by Lars Arge, Michael Hoffmann and Emo Welzl. Berlin, Heidelberg: Springer, 2007, pp. 594–604. ISBN: 978-3-540-75520-3. DOI: 10.1007/978-3-540-75520-3_53.

[65] Anne Driemel, Sariel Har-Peled and Carola Wenk. 'Approximating the Fréchet distance for realistic curves in near linear time'. In: *Discrete & Computational Geometry* 48.1 (July 2012), pp. 94–127. ISSN: 1432-0444. DOI: 10.1007/s00454-012-9402-z.

[66] Anne Driemel, Ivor van der Hoog and Eva Rotenberg. 'On the Discrete Fréchet Distance in a Graph'. In: *Proc. 38th International Symposium on Computational Geometry (SoCG)*. Ed. by Xavier Goaoc and Michael Kerber. Vol. 224. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, 36:1–36:18. ISBN: 978-3-95977-227-3. DOI: 10.4230/LIPIcs.SoCG.2022.36.

[67] Fabian Dütsch and Jan Vahrenhold. 'A Filter-and-Refinement-Algorithm for Range Queries Based on the Fréchet Distance (GIS Cup)'. In: *Proc. 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL)*. New York, NY, USA: Association for Computing Machinery, Nov. 2017, pp. 1–4. ISBN: 978-1-4503-5490-5. DOI: 10.1145/3139958.3140063.

[68] Herbert Edelsbrunner. *Algorithms in Combinatorial Geometry*. Vol. 10. EATCS Monographs on Theoretical Computer Science. Springer, 1987. DOI: 10.1007/978-3-642-61568-9.

[69] Herbert Edelsbrunner. 'Computing the extreme distances between two convex polygons'. In: *Journal of Algorithms* 6.2 (June 1985), pp. 213–224. ISSN: 0196-6774. DOI: 10.1016/0196-6774(85)90039-2.

[70] Alon Efrat, Piotr Indyk and Suresh Venkatasubramanian. 'Pattern Matching for Sets of Segments'. In: *Algorithmica* 40.3 (2004), pp. 147–160. DOI: 10.1007/s00453-004-1089-y.

[71] Thomas Eiter and Heikki Mannila. *Computing Discrete Frechet Distance*. Tech. rep. CD-TR 94/64. Technische Universität Wien, 1994. URL: http://www.kr.tuwien.ac.at/staff/eiter/et-archive/cdtr9464.pdf.

[72] Jeff Erickson. *Algorithms*. First. June 2019.

[73] Chenglin Fan and Benjamin Raichel. 'Computing the Fréchet Gap Distance'. In: *Discrete & Computational Geometry* 65.4 (June 2021), pp. 1244–1274. ISSN: 1432-0444. DOI: 10.1007/s00454-020-00224-w.

[74] Arnold Filtser and Omrit Filtser. 'Static and Streaming Data Structures for Fréchet Distance Queries'. In: *ACM Transactions on Algorithms* 19.4 (Oct. 2023), 39:1–39:36. ISSN: 1549-6325. DOI: 10.1145/3610227.

[75] Arnold Filtser, Omrit Filtser and Matthew J. Katz. 'Approximate Nearest Neighbor for Curves - Simple, Efficient, and Deterministic'. In: *Proc. 47th International Colloquium on Automata, Languages, and Programming (ICALP)*. Ed. by Artur Czumaj, Anuj Dawar and Emanuela Merelli. Vol. 168. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020, 48:1–48:19. ISBN: 978-3-95977-138-2. DOI: 10.4230/LIPIcs.ICALP.2020.48.

[76] Arnold Filtser, Omrit Filtser and Matthew J. Katz. 'Approximate Nearest Neighbor for Curves: Simple, Efficient, and Deterministic'. In: *Algorithmica* 85.5 (2023), pp. 1490–1519. DOI: 10.1007/s00453-022-01080-1.

[77] Omrit Filtser et al. 'On Flipping the Fréchet Distance'. In: *Proc. 14th Innovations in Theoretical Computer Science Conference (ITCS)*. Ed. by Yael Tauman Kalai. Vol. 251. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023, 51:1–51:22. ISBN: 978-3-95977-263-1. DOI: 10.4230/LIPIcs.ITCS.2023.51.

[78] Steven Fortune. 'A Sweepline Algorithm for Voronoi Diagrams'. In: *Algorithmica* 2.1–4 (1987), pp. 153–174. DOI: 10.1007/BF01840357.

[79] Emily Fox et al. 'Fréchet Edit Distance'. In: *Proc. 40th International Symposium on Computational Geometry (SoCG)*. Vol. 293. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024, 58:1–58:15. DOI: 10.4230/LIPIcs.SoCG.2024.58.

[80] M. Maurice Fréchet. 'Sur quelques points du calcul fonctionnel'. In: *Rendiconti del Circolo Matematico di Palermo (1884-1940)* 22.1 (Dec. 1906), pp. 1–72. ISSN: 0009-725X. DOI: 10.1007/BF03018603.

[81] Anka Gajentaan and Mark H. Overmars. 'On a class of $O(n^2)$ problems in computational geometry'. In: *Computational Geometry* 45.4 (May 2012), pp. 140–152. ISSN: 0925-7721. DOI: 10.1016/j.comgeo.2011.11.006.

[82] Jiawei Gao et al. 'Completeness for First-order Properties on Sparse Structures with Algorithmic Applications'. In: *ACM Transactions on Algorithms* 15.2 (Dec. 2018), 23:1–23:35. ISSN: 1549-6325. DOI: 10.1145/3196275.

[83] Alexander Gavruskin et al. ‘Dynamic algorithms for monotonic interval scheduling problem’. In: *Theoretical Computer Science* 562 (Jan. 2015), pp. 227–242. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2014.09.046.

[84] Laxmi P. Gewali et al. ‘Path planning in $0/1/\infty$ weighted regions with applications’. In: *Proc. 4th Annual Symposium on Computational Geometry (SoCG)*. New York, NY, USA: Association for Computing Machinery, Jan. 1988, pp. 266–278. ISBN: 978-0-89791-270-9. DOI: 10.1145/73393.73421.

[85] Jacob E. Goodman, Joseph O’Rourke and Csaba D. Tóth, eds. *Handbook of Discrete and Computational Geometry*. 3rd. CRC Press, 2017. DOI: 10.1201/9781315119601.

[86] Joachim Gudmundsson, Patrick Laube and Thomas Wolle. ‘Computational Movement Analysis’. In: *Springer Handbook of Geographic Information*. Springer, 2012, pp. 423–438. DOI: 10.1007/978-3-540-72680-7_22.

[87] Joachim Gudmundsson, Martin P. Seybold and John Pfeifer. ‘On practical nearest sub-trajectory queries under the Fréchet distance’. In: *Proc. 29th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL)*. New York, NY, USA: Association for Computing Machinery, Nov. 2021, pp. 596–605. ISBN: 978-1-4503-8664-7. DOI: 10.1145/3474717.3484264.

[88] Joachim Gudmundsson, Martin P. Seybold and Sampson Wong. ‘Map Matching Queries on Realistic Input Graphs Under the Fréchet Distance’. In: *ACM Transactions on Algorithms* 20.2 (Mar. 2024), 14:1–14:33. ISSN: 1549-6325. DOI: 10.1145/3643683.

[89] Joachim Gudmundsson, Martin P. Seybold and Sampson Wong. ‘Map matching queries on realistic input graphs under the Fréchet distance’. In: *Proc. 34th ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Ed. by Nikhil Bansal and Viswanath Nagarajan. SIAM, 2023, pp. 1464–1492. DOI: 10.1137/1.9781611977554.ch53.

[90] Joachim Gudmundsson, Yuan Sha and Sampson Wong. ‘Approximating the packedness of polygonal curves’. In: *Computational Geometry* 108 (Jan. 2023), p. 101920. ISSN: 0925-7721. DOI: 10.1016/j.comgeo.2022.101920.

[91] Joachim Gudmundsson and Michiel Smid. 'Fréchet queries in geometric trees'. In: *Proc. 21st Annual European Symposium on Algorithms (ESA)*. Ed. by Hans L. Bodlaender and Giuseppe F. Italiano. Berlin, Heidelberg: Springer, 2013, pp. 565–576. ISBN: 978-3-642-40450-4. DOI: 10.1007/978-3-642-40450-4_48.

[92] Joachim Gudmundsson, Andreas Thom and Jan Vahrenhold. 'Of motifs and goals: mining trajectory data'. In: *Proc. 20th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL)*. New York, NY, USA: Association for Computing Machinery, Nov. 2012, pp. 129–138. ISBN: 978-1-4503-1691-0. DOI: 10.1145/2424321.2424339.

[93] Joachim Gudmundsson and Nacho Valladares. 'A GPU approach to subtrajectory clustering using the Fréchet distance'. In: *IEEE Transactions on Parallel and Distributed Systems* 26.4 (Apr. 2015), pp. 924–937. ISSN: 1558-2183. DOI: 10.1109/TPDS.2014.2317713.

[94] Joachim Gudmundsson and Thomas Wolle. 'Football analysis using spatio-temporal tools'. In: *Proc. 20th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL)*. New York, NY, USA: Association for Computing Machinery, Nov. 2012, pp. 566–569. ISBN: 978-1-4503-1691-0. DOI: 10.1145/2424321.2424417.

[95] Joachim Gudmundsson and Sampson Wong. 'Cubic upper and lower bounds for subtrajectory clustering under the continuous Fréchet distance'. In: *Proc. 33rd ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Society for Industrial and Applied Mathematics, Jan. 2022, pp. 173–189. DOI: 10.1137/1.9781611977073.9.

[96] Joachim Gudmundsson et al. 'Computing a Subtrajectory Cluster from c-Packed Trajectories'. In: *Proc. 34th International Symposium on Algorithms and Computation (ISAAC)*. Ed. by Satoru Iwata and Naonori Kakimura. Vol. 283. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023, 34:1–34:15. ISBN: 978-3-95977-289-1. DOI: 10.4230/LIPIcs.ISAAC.2023.34.

[97] Joachim Gudmundsson et al. ‘Translation invariant Fréchet distance queries’. In: *Algorithmica* 83.11 (Aug. 2021), pp. 3514–3533. DOI: 10.1007/s00453-021-00865-0.

[98] Leonidas Guibas, John Hershberger and Jack Snoeyink. ‘Compact interval trees: a data structure for convex hulls’. In: *International Journal of Computational Geometry and Applications* 01.01 (Mar. 1991), pp. 1–22. ISSN: 0218-1959. DOI: 10.1142/S0218195991000025.

[99] Leonidas J. Guibas et al. ‘Approximating polygons and subdivisions with minimum link paths’. In: *Proc. 2nd International Symposium on Algorithms (ISA)*. Ed. by Wen-Lian Hsu and R. C. T. Lee. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 1991, pp. 151–162. ISBN: 978-3-540-46600-0. DOI: 10.1007/3-540-54945-5_59.

[100] Dan Halperin and Micha Sharir. ‘Arrangements’. In: *Handbook of Discrete and Computational Geometry*. 3rd ed. Num Pages: 40. Chapman and Hall/CRC, 2017. ISBN: 978-1-315-11960-1.

[101] Sariel Har-Peled. *Geometric Approximation Algorithms*. USA: American Mathematical Society, 2011. ISBN: 978-0-8218-4911-8. DOI: 10.1090/surv/173.

[102] Sariel Har-Peled and Benjamin Raichel. ‘The fréchet distance revisited and extended’. In: *ACM Transactions on Algorithms* 10.1 (Jan. 2014), 3:1–3:22. ISSN: 1549-6325. DOI: 10.1145/2532646.

[103] Felix Hausdorff. *Grundzüge der Mengenlehre*. Leipzig: Veit & Comp., 1914.

[104] John Hershberger and Subhash Suri. ‘An Optimal Algorithm for Euclidean Shortest Paths in the Plane’. In: *SIAM Journal on Computing* 28.6 (Jan. 1999), pp. 2215–2256. ISSN: 0097-5397. DOI: 10.1137/S0097539795289604.

[105] Thijs van der Horst and Tim Ophelders. ‘Faster Fréchet Distance Approximation Through Truncated Smoothing’. In: *Proc. 40th International Symposium on Computational Geometry (SoCG)*. Ed. by Wolfgang Mulzer and Jeff M. Phillips. Vol. 293. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024, 63:1–63:15. ISBN: 978-3-95977-316-4. DOI: 10.4230/LIPIcs.SoCG.2024.63.

[106] Thijs van der Horst et al. ‘A Subquadratic nε-approximation for the Continuous Fréchet Distance’. In: *Proc. 34th ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Society for Industrial and Applied Mathematics, Jan. 2023, pp. 1759–1776. DOI: 10.1137/1.9781611977554.ch67.

[107] Amin Hosseinpoor Milaghardan et al. ‘An activity-based framework for detecting human movement patterns in an urban environment’. In: *Transactions in GIS* 25.4 (2021), pp. 1825–1848. ISSN: 1467-9671. DOI: 10.1111/tgis.12749.

[108] Zehao Huang et al. *Measuring eye-tracking accuracy and its impact on usability in apple vision pro*. arXiv:2406.00255 [cs]. Aug. 2024. DOI: 10.48550/arXiv.2406.00255.

[109] Minghui Jiang, Ying Xu and Binhai Zhu. ‘Protein Structure-Structure Alignment with Discrete Fréchet Distance’. In: *Proc. 5th Asia-Pacific Bioinformatics Conference (APBC)*. Ed. by David Sankoff, Lusheng Wang and Francis Y. L. Chin. Vol. 5. Advances in Bioinformatics and Computational Biology. Imperial College Press, 2007, pp. 131–141. DOI: 10.1142/S0219720008003278.

[110] Andrew B. Kahng et al. *VLSI Physical Design: From Graph Partitioning to Timing Closure*. Springer, 2011. DOI: 10.1007/978-90-481-9591-6.

[111] Sertac Karaman and Emilio Frazzoli. ‘Sampling-based algorithms for optimal motion planning’. In: *The International Journal of Robotics Research* 30.7 (2011), pp. 846–894. DOI: 10.1177/0278364911406761.

[112] Adam Karczmarz and Marcin Smulewicz. ‘Fully Dynamic Strongly Connected Components in Planar Digraphs’. In: *Proc. 51st International Colloquium on Automata, Languages, and Programming (ICALP)*. Ed. by Karl Bringmann et al. Vol. 297. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024, 95:1–95:20. ISBN: 978-3-95977-322-5. DOI: 10.4230/LIPIcs.ICALP.2024.95.

[113] Roland Kays, James Flowers and Suzanne Kennedy-Stoskopf. *Cat tracker project*. 2016. URL: www.movebank.org.

[114] Fengfeng Ke et al. ‘Using eye-tracking in education: review of empirical research and technology’. In: *Educational technology research and development* 72.3 (June 2024), pp. 1383–1418. ISSN: 1556-6501. DOI: 10.1007/s11423-024-10342-4.

[115] Mees van de Kerkhof et al. ‘Global Curve Simplification’. In: *Proc. 27th Annual European Symposium on Algorithms (ESA)*. Ed. by Michael A. Bender, Ola Svensson and Grzegorz Herman. Vol. 144. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019, 67:1–67:14. ISBN: 978-3-95977-124-5. DOI: 10.4230/LIPIcs.ESA.2019.67.

[116] David Kirkpatrick and Jack Snoeyink. ‘Computing common tangents without a separating line’. In: *Proc. 4th Workshop on Algorithms and Data Structures (WADS)*. Ed. by Gerhard Goos et al. Vol. 955. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 183–193. ISBN: 978-3-540-60220-0 978-3-540-44747-4. DOI: 10.1007/3-540-60220-8_61.

[117] Christian Knauer. ‘Algorithms for Comparing Geometric Patterns’. PhD thesis. 2002. DOI: 10.17169/refubium-16312.

[118] Marc van Kreveld, Maarten Löffler and Lionov Wiratma. ‘On optimal polyline simplification using the Hausdorff and Fréchet distance’. In: *Journal of Computational Geometry* 11.1 (Mar. 2020). Number: 1, pp. 1–25. ISSN: 1920-180X. DOI: 10.20382/jocg.v11i1a1.

[119] Mark Lanthier, Anil Maheshwari and Jörg-Rüdiger Sack. ‘Approximating weighted shortest paths on polyhedral surfaces’. In: *Proc. 13th Annual Symposium on Computational Geometry (SoCG)*. New York, NY, USA: Association for Computing Machinery, Aug. 1997, pp. 274–283. ISBN: 978-0-89791-878-7. DOI: 10.1145/262839.262984.

[120] Jae-Gil Lee, Jiawei Han and Kyu-Young Whang. ‘Trajectory clustering: a partition-and-group framework’. In: *Proc. 2007 ACM SIGMOD International Conference on Management of Data (SIGMOD)*. New York, NY, USA: Association for Computing Machinery, June 2007, pp. 593–604. ISBN: 978-1-59593-686-8. DOI: 10.1145/1247480.1247546.

[121] Alon Lerner, Yiorgos Chrysanthou and Dani Lischinski. 'Crowds by Example'. In: *Computer Graphics Forum* 26.3 (2007), pp. 655–664. DOI: 10.1111/j.1467-8659.2007.01089.x.

[122] Jesse Levinson et al. 'Towards Fully Autonomous Driving: Systems and Algorithms'. In: *Proc. IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2011, pp. 163–168. DOI: 10.1109/IVS.2011.5940562.

[123] William E. Lorensen and Harvey E. Cline. 'Marching Cubes: A High Resolution 3D Surface Construction Algorithm'. In: *Proc. 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*. ACM, 1987, pp. 163–169. DOI: 10.1145/37401.37422.

[124] Anil Maheshwari et al. 'Fréchet distance with speed limits'. In: *Computational Geometry* 44.2 (Feb. 2011). Special issue of selected papers from the 21st Annual Canadian Conference on Computational Geometry, pp. 110–120. ISSN: 0925-7721. DOI: 10.1016/j.comgeo.2010.09.008.

[125] Joseph S. B. Mitchell and Christos H. Papadimitriou. 'The weighted region problem: finding shortest paths through a weighted planar subdivision'. In: *Journal of the ACM* 38.1 (Jan. 1991), pp. 18–73. ISSN: 0004-5411. DOI: 10.1145/102782.102784.

[126] Giri Narasimhan and Michiel Smid. *Geometric Spanner Network*. Cambridge University Press, 2007. ISBN: 978-0-511-54688-4. DOI: 10.1017/CBO9780511546884.

[127] Abhinandan Nath and Erin Taylor. 'k-Median Clustering Under Discrete Fréchet and Hausdorff Distances'. In: *Proc. 36th International Symposium on Computational Geometry (SoCG)*. Ed. by Sergio Cabello and Danny Z. Chen. Vol. 164. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020, 58:1–58:15. ISBN: 978-3-95977-143-6. DOI: 10.4230/LIPIcs.SoCG.2020.58.

[128] Giuliana Pallotta, Michele Vespe and Karna Bryan. 'Vessel Pattern Knowledge Discovery from AIS Data: A Framework for Anomaly Detection and Route Prediction'. In: *Entropy* 15.6 (2013), pp. 2218–2245. DOI: 10.3390/e15062218.

[129] Franco P. Preparata and Michael Ian Shamos. *Computational Geometry: An Introduction*. Springer, 1985. DOI: 10.1007/978-1-4612-1098-6.

[130] Anna Sebernegg, Peter Kán and Hannes Kaufmann. *Motion Similarity Modeling – A State of the Art Report*. arXiv:2008.05872 [cs]. Aug. 2020. DOI: 10.48550/arXiv.2008.05872.

[131] Michael Ian Shamos. 'Computational Geometry'. PhD thesis. Yale University, 1978.

[132] Daniel D. Sleator and Robert Endre Tarjan. 'A data structure for dynamic trees'. In: *Journal of Computer and System Sciences* 26.3 (June 1983), pp. 362–391. ISSN: 0022-0000. DOI: 10.1016/0022-0000(83)90006-5.

[133] E. Sriraghavendra, K. Karthik and Chiranjib Bhattacharyya. 'Fréchet Distance Based Approach for Searching Online Handwritten Documents'. In: *Proc. 9th International Conference on Document Analysis and Recognition (ICDAR)*. IEEE Computer Society, 2007, pp. 461–465. DOI: 10.1109/ICDAR.2007.4378752.

[134] Sairam Subramanian. 'A Fully Dynamic Data Structure for Reachability in Planar Digraphs'. In: *Proc. 1st Annual European Symposium on Algorithms (ESA)*. Ed. by Thomas Lengauer. Vol. 726. Lecture Notes in Computer Science. Springer, 1993, pp. 372–383. DOI: 10.1007/3-540-57273-2_72.

[135] Zheng Sun and John H. Reif. 'On finding approximate optimal paths in weighted regions'. In: *Journal of Algorithms* 58.1 (Jan. 2006), pp. 1–32. ISSN: 0196-6774. DOI: 10.1016/j.jalgor.2004.07.004.

[136] Panagiotis Tampakis et al. 'Scalable distributed subtrajectory clustering'. In: *Proc. 2019 IEEE International Conference on Big Data (BigData)*. IEEE Computer Society, Dec. 2019, pp. 950–959. ISBN: 978-1-7281-0858-2. DOI: 10.1109/BigData47090.2019.9005563.

[137] Victor Andreevich Toponogov. *Differential Geometry of Curves and Surfaces: A Concise Guide*. Birkhauser. DOI: 10.1007/b137116.

[138] R.C. Veltkamp. 'Shape matching: similarity measures and algorithms'. In: *Proc. International Conference on Shape Modeling and Applications (SMI)*. May 2001, pp. 188–197. DOI: 10.1109/SMA.2001.923389.

[139] Remco C. Veltkamp and Michiel Hagedoorn. 'State of the Art in Shape Matching'. In: *Principles of Visual Information Retrieval*. Ed. by Michael S. Lew. London: Springer,

2001, pp. 87–119. ISBN: 978-1-4471-3702-3. DOI: 10.1007/978-1-4471-3702-3_4.

[140] Zheng Wang, Cheng Long and Gao Cong. 'Similar sports play retrieval with deep reinforcement learning'. In: *IEEE Transactions on Knowledge and Data Engineering* (2021), pp. 1–1. ISSN: 1558-2191. DOI: 10.1109/TKDE.2021.3136881.

[141] Carola Wenk. 'Shape Matching in Higher Dimensions'. PhD thesis. 2003. DOI: 10.17169/refubium-8310.

[142] Yu Zheng. 'Trajectory Data Mining: An Overview'. In: *ACM Transactions on Intelligent Systems and Technology* 6.3 (2015), 29:1–29:41. DOI: 10.1145/2743025.