DISCRETE GEOMETRIC PATH ANALYSIS IN
COMPUTATIONAL GEOMETRY

by
Amin Gheibi

A thesis submitted to
the Faculty of Graduate and Postdoctoral Affairs
in partial fulfillment of
the requirements for the degree of

DOCTOR OF PHILOSOPHY

in
Computer Science

CARLETON UNIVERSITY

Ottawa, Ontario
2015

© Copyright by Amin Gheibi, 2015

To my parents and my wife

Abstract

The geometric shortest path problem is one of the fundamental problems in Computational Geometry and related fields. In the first part of this thesis, we study the weighted region problem (WRP), which is to compute a geometric shortest path on a weighted partitioning of a plane. Recent results show that WRP is not solvable in any algebraic computation model over rational numbers. Thus, scientists have focused on approximate solutions. We first study the WRP when the input partitioning of space is an arrangement of lines. We provide a technique that makes it possible to apply the existing approximation algorithms for triangulations to arrangements of lines. Then, we formulate two qualitative criteria for weighted short paths. We show how to produce a path that is quantitatively close-to-optimal and qualitatively satisfactory. The results of our experiments carried out on triangular irregular networks (TINs) show that the proposed algorithm could save, on average, 51% in query time and 69% in memory usage, in comparison with the existing method.

In the second part of the thesis, we study some variants of the Fréchet distance. The Fréchet distance is a well-studied and commonly used measure to capture the similarity of polygonal curves. All of the problems that we studied here can be reduced to a geometric shortest path problem in configuration space. Firstly, we study a robust variant of the Fréchet distance since the standard Fréchet distance exhibits a high sensitivity to the presence of outliers. Secondly, we propose a new measure to capture similarity between polygonal curves, called the *minimum backward Fréchet distance* (MBFD). More specifically, for a given threshold ε , we are searching for a pair of walks for two entities on the two input polygonal curves such that the union of the portions of required backward movements is minimized and the distance between the two entities, at any time during the walk, is less than or equal to ε . Thirdly, we generalize MBFD to capture scenarios when the cost of backtracking on the input polygonal curves is not homogeneous. More specifically, each edge of input polygonal curves has an associated non-negative weight. The cost of backtracking on an edge is the Euclidean length of backward movement on that edge multiplied by the corresponding edge weight. Lastly, for a given graph H , a polygonal curve T , and a threshold ε , we propose a geometric algorithm that computes a path, P , in H , and a parameterization of T , that minimize the sum of the length of walks on T and P whereby the distance between the entities moving along P and T is at most ε , at any time during the walks.

Acknowledgements

Prima facie, I am grateful to the God for well-being that were necessary to complete this thesis. I wish to express my sincere thanks to Prof. Jörg-Rüdiger Sack, my supervisor, for all of his admirable guidance and support. His advice on both research as well as on my career have been priceless. I am also grateful to Prof. Anil Maheshwari, who advised me, through my academic endeavor at Carleton University. The door of his office has always been open to me and his constructive comments have improved the quality of my research significantly.

I would like to thank Prof. Carola Wenk for suggesting the map matching topic in this thesis. Also, when I was not able to travel to ACM SIGSPATIAL conference, to present our paper (due to visa issue), she kindly accepted to present our paper. Her comments and suggestions improved this thesis tremendously. I would like to thank Prof. Vida Dujmović for tremendous comments that have improved my thesis. I am also grateful to Prof. Michiel Smid, for constructive comments and suggestions. He was my first instructor on computational geometry, at Carleton University, who inspired me.

I would like to thank Dr. Patrick Boily at School of Mathematics and Statistics, Carleton University, to discuss the statistical analysis. Also, I thank Dr. Andre Pugin, Natural Resources of Canada, Prof. Dariush Motazedian and Prof. Claire Samson, Department of Earth Sciences, Carleton University, for the discussions that lead to the new qualitative measures. I am grateful to Prof. Yusu Wang for communication which contains clarifications to Theorem 4.2 in [124]. I take this opportunity to express gratitude to all of the School faculty members and staff for their help and support. I would like to thank my friends, Dr. Christian Scheffer, Dr. Hamid Zarrabi-zadeh, Dr. Kaveh Shahbaz and Dr. Masoud Omran who helped me a lot with constructive discussions. I also place on record, my sense of gratitude to one and all, who directly or indirectly, have lent their hand in this venture.

A special thanks to my parents, brothers, and family. Words cannot express how grateful I am to my mother and father for all of the sacrifices that they have made. At last, but not least, I would like to express appreciation to my beloved wife who made my cold nights in Ottawa, warm and delightful.

Table of Contents

Abstract	iv
Acknowledgements	v
Chapter 1 Introduction	1
1.1 Introduction and Motivation	1
1.1.1 Geometric Shortest Path Problem	3
1.1.2 Fréchet Distance	7
1.2 Thesis Outline and Contributions	10
1.3 Shortest Path Literature Review	15
1.3.1 Shortest Path in Polygons	15
1.3.2 Minimum Link Path	20
1.3.3 Manhattan Shortest Path	21
1.3.4 Weighted Region Problem (WRP)	22
1.3.5 Shortest Path in 3D	25
1.4 Fréchet Distance Literature Review	27
1.4.1 Hausdorff Distance	27
1.4.2 Fréchet Distance	28
1.4.3 Coupling Distance	31
1.4.4 Lower Bound	32
Chapter 2 Weighted Region Problem in Arrangement of Lines	34
2.1 Preliminaries	36
2.2 Geometric Properties	37

2.3	The Construction Algorithm	44
2.4	Minimality of the SP-Hull	48
2.5	Conclusion	49
Chapter 3	Path Refinement in Weighted Regions	50
3.1	Introduction	50
3.2	Preliminaries and Definitions	54
3.3	Algorithms	56
3.3.1	Refinement Algorithm for Triangulations	57
3.3.2	Refinement Algorithm for Parallel Lines	71
3.3.3	Refinement Algorithm for Arrangements of Lines	73
3.4	Experimental Results	74
3.4.1	Motivation	74
3.4.2	Experimental Setup	75
3.4.3	Results	79
3.4.4	Conclusions of Experiments	93
3.5	Conclusions	94
Chapter 4	Similarity of Polygonal Curves in the Presence of Outliers	95
4.1	Introduction	95
4.1.1	Preliminaries	97
4.1.2	Problem Definition	99
4.1.3	Counterexample	100
4.1.4	New Results	101
4.2	An Approximation Algorithm	102
4.3	Improvement	112

4.3.1	An Auxiliary Lemma	112
4.3.2	Construction of G^*	115
4.3.3	Improved Algorithms for the MinEx and MaxIn Problems	116
4.3.4	Is FPTAS Achievable?	118
4.4	Conclusion	121
Chapter 5	Minimum Backward Fréchet Distance	123
5.1	Introduction	123
5.2	Problem Definition	124
5.3	Algorithm	125
5.4	Improvement	129
5.5	Conclusion	132
Chapter 6	Weighted Minimum Backward Fréchet Distance	134
6.1	Introduction	134
6.2	Preliminaries and Problem Definition	136
6.3	Algorithm	137
6.4	Improvement	148
6.4.1	First Step	148
6.4.2	Second Step	152
6.5	Conclusion	155
Chapter 7	Minimizing Walking Length in Map Matching	156
7.1	Introduction	156
7.2	Preliminaries and Definitions	159
7.3	Algorithm	162

7.4	Improvement	170
7.5	Weighted Non-planar Graphs	174
7.6	Conclusion	174
Chapter 8	Open Problems and Future Work	176
Bibliography		181

List of Tables

Table 3.1	14 Triangular Irregular Networks (TINs) for experiments	76
Table 3.2	Comparing refinement process and enhanced sleeve methods	78
Table 3.3	The result of the methods on five TINs. Number of Edges of the Graph ($ E $), Pre-processing Time (T_p), Average Query Time (QT_{av}), Accuracy (AC), Average Memory Usage (M_{av}), Top 5 percent Average Memory Usage (TM_{av}), Method 1: Refinement Process, Method 2: Enhanced Sleeve, Method 3: Hybrid	81
Table 3.4	The result of the methods on other TINs. Number of Edges of the Graph ($ E $), Pre-processing Time (T_p), Average Query Time (QT_{av}), Accuracy (AC), Average Memory Usage (M_{av}), Top 5 percent Average Memory Usage (TM_{av}), Method 1: Refinement Process, Method 2: Enhanced Sleeve, Method 3: Hybrid	83
Table 3.5	The result of the methods on the Everest TIN with weights rather than slopes. In Random Everest the weights are assigned randomly. In Flat Everest all the weights are the same. Number of Edges of the Graph ($ E $), Pre-processing Time (T_p), Average Query Time (QT_{av}), Accuracy (AC), Average Memory Usage (M_{av}), Top 5 percent Average Memory Usage (TM_{av}), Method 1: Refinement Process, Method 2: Enhanced Sleeve, Method 3: Hybrid	84
Table 3.6	The result of fitting 2-parameter Weibull distribution that shape parameter k is less than 1. Also the result of Kolmogorov-Smirnov goodness-of-fit test (K-S) is reported.	91

Table 3.7	Correlation between the measures of the distributions of the TINs and the accuracy of our algorithm on the TINs.	93
-----------	---	----

List of Figures

Figure 1.1	a) A simply connected region. b) A multiply connected region. c) A polygonal domain with 17 edges and 2 holes.	5
Figure 1.2	a) The dashed line segment shows the Hausdorff distance between T_1 and T_2 and the dash dotted line segment shows the standard Fréchet distance between them. b) The dash dotted line segment shows the standard Fréchet distance between T_1 and T_2 and the dashed line segment shows the coupling distance between them.	11
Figure 1.3	Split operation of a funnel, $F(s, \overline{ab})$, with root of r and base $\overline{ab}$	17
Figure 2.1	For each line in the arrangement there are two rays (in blue). Also, each vertex of $\mathcal{CH}(P)$, denoted by c_i , has two chains, chain_i^{ccw} and chain_i^{cw} (the red dashed lines in the figure). One of the inner angles of chain_i^{ccw} is shown in the figure (incident at r_{i+3}). Furthermore, suppose the weight of f_i is “very large” and the weight of f_{i+1} is “very small”. Then, π_{st} goes outside of $\mathcal{CH}(P)$	38
Figure 2.2	a) Property 1a, the normal from x to r_j lies on the left side of the normal from x to r_i . b) Property 1b, the normals from x and y to r_i lie on the opposite sides of $\overline{xy}$. c) Property 2b, if $\overline{xh_1}$ intersects with $\overline{yh_2}$, then $h_2 < x$ and $h_1 < y$. d) Lemma 1, one of the normals, either from c_i to r_{i+1} or from c_{i+1} to r_{i+k} , lies outside of $\mathcal{CH}(P)$	39

Figure 2.3 a) Two chains, chain_i^{ccw} (the red dashed chain) and chain_j^{cw} (the blue dashed chain), and their common tangent, l_t . b) An example of the topological structure of the SP-Hull is shown in black solid lines. The red dashed line is the assumed weighted shortest path between s and t . 41

Figure 2.4 a) Proof of Theorem 1, case 1. b) Proof of Theorem 1, case 2. 47

Figure 3.1 The sub-path $\Pi[x_{i-1}, x_{i+1}]$ (the solid line) disobeys a) CAR, b) CNR. The dotted path shows a replacement that obeys a) CAR, b) CNR. . . 55

Figure 3.2 The shortest path inside a discretization of WRP by Steiner points violates CNR. 56

Figure 3.3 The line segment e is the interface between two triangles, Δ_i and Δ_j . a) The $\text{Passage}_{u,e}$ is shown by a red solid line. b) The $\text{Passage}_{u,e}$ is empty. 58

Figure 3.4 Characterization of CNR dependencies by geometrical configurations. . 59

Figure 3.5	The directed red polygonal chain shows the sub-path from x_{i+1} to x_{k_3} in the original input path. The corner x_{k_1} is the first corner in this sub-path that is disobeying the CNR. The naïve approach replaces the sub-path from x_{i+1} to x_{k_1} by the orange polygonal chain whose first segment is the only segment from the forward chain. After that replacement, the naïve algorithm continues and finds the corner x_{k_2} that disobeys CNR. The algorithm replaces the sub-path from x_{i+2} to x_{k_2} by the blue polygonal chain. If the replacement happens $n/2$ times and the number of corners between x_{i+1} and x_{k_1} is $n/2$, then the total time complexity is quadratic in the size of the input path. In this example, the corner x_{k_3} also disobeys CNR after replacing the sub-path from x_{i+2} to x_{k_2} by the blue polygonal chain. The green polygonal curve shows the final replacement of the sub-path from x_{i+1} to x_{k_3} .	60
Figure 3.6	A dependency chain $\Pi[x_i, x_{i+3}]$ is illustrated in red. The triple $\Pi[x_{i+1}, x_{i+3}]$ disobeys CNR.	61
Figure 3.7	Three different cases illustrate the relative positions of a forward (blue solid) and backward (red dashed) chain to each other.	63
Figure 3.8	$\Pi[x_i, x_{i+\ell}] = \langle x_i, \dots, x_{i+\ell} \rangle$ is a dependency chain. Before replacing $\Pi[x_i, x_{i+\ell}]$ we have $x_{i+\ell} \not\triangleright x_{i+\ell+1}$ and afterwards $x_{i+\ell} \triangleright x_{i+\ell+1}$.	64
Figure 3.9	Configurations of the inductive proof for Lemma 13.	66
Figure 3.10	Corners of forward and backward chains lie closer to v than $x_i, \dots, x_{last}$.	69
Figure 3.11	Segment $\overline{x_s^* x_{s+1}^*}$ is introduced by the merge operation. We prove $ \overline{x_s^* x_{s+1}^*} \leq \overline{x_s x_{s+1}} $.	70

Figure 3.12	a) Parallel lines with source and target points. b) Local refinement on I_i is shown. $Cone_{x_{i-1}, I_i}$ is hatched. Solid line between x_{i-1} and x_{i+1} shows the path before the translation of x_i and dashed line is the path after refinement.	71
Figure 3.13	Two different procedures to capture TINs for the experiment.	75
Figure 3.14	Top, front, and perspective view of Everest TIN	77
Figure 3.15	The accuracies of the post-processing methods on G_3 are plotted, for a) Alborz TIN, b) Damavand TIN, c) Everest TIN, d) Grand Canyon TIN, e) Uttarakhand TIN, f) Everest TIN with random associated weights to faces. In these plots, the first bin, called Goal, shows the percentage of the queries that reached the baseline quality (i.e., the path length in G_6). The second, third, and fourth bins show the percentages of the queries that reached at least 99, 95 and 90 percent of the baseline quality, respectively.	82
Figure 3.16	The improvement of the refinement post-processing method's accuracy on G_3 , by re-applying alternately from s to t and from t to s , is plotted, for a) Alborz TIN, b) Damavand TIN, c) Everest TIN, d) Grand Canyon TIN, e) Uttarakhand TIN.	86
Figure 3.17	The constructed example for showing that the local refinement may take arbitrary number of steps to converge.	87
Figure 3.18	An expensive face (i.e., steep) is adjacent to an inexpensive face (i.e., horizontal). Some of the Steiner points are shown by crosses. The solid orange line is the path in the graph of Steiner points. The dashed line is the refined path.	89

Figure 3.19	Distribution of the differences between the slope of the adjacent faces in TINs in Table 3.1. Everest and Thompson that our algorithm has the best and the worst accuracy respectively, are shown by solid lines.	90
Figure 3.20	The scatter plot of the a) scale parameter, b) shape parameter, c) expected value, d) standard deviation, and, e) median of Weibull distribution versus the accuracy of the algorithm.	92
Figure 4.1	a) A possible solution is illustrated by the connecting lines between the parameterizations for T_1 and T_2 . The sub-curves on both polygonal curves that should be ignored are illustrated by the blue, red and green sub-curves on T_1 and T_2 . So, $Q^B(T_1, T_2)$ is the summation of the lengths of the colored sub-curves and $Q^W(T_1, T_2)$ is that of the black sub-curves. b) The solution corresponds to an xy -increasing path in the deformed free-space diagram F . In this space, $Q^B(T_1, T_2)$ can be measured by summing the lengths of its subpaths going through the forbidden space (shaded gray area), measured in the L_1 -metric (similarly for $Q^W(T_1, T_2)$)—see Subsection 4.1.2 for definitions of $Q^B(T_1, T_2)$ and $Q^W(T_1, T_2)$	97
Figure 4.2	Counterexample to $\omega = O(Q^W(T_1, T_2))$. a) Two trajectories T_1 and T_2 lie parallel to each other. They have opposite directions. b) Free-space of T_1 and T_2	101
Figure 4.3	Two polygonal curves and the corresponding deformed free space diagram F for a given ε	104
Figure 4.4	Illustration of Step 2 of insertions of grid and intersection lines in F of Figure 4.3.	104

Figure 4.5	Four cases of configurations for s and t , and their corresponding grid lines that are used in the proof of Lemma 15. The path $\tilde{\pi}_{s't'}$ in G is shown in red.	107
Figure 4.6	Illustration of the proof of Lemma 16.	107
Figure 4.7	(a) The point set P is partitioned with respect to its median line m . Each $p_i \in P_{middle}$ is projected onto m (blue arrows and red points). The projections are ordered with respect to their y -coordinates (orange arrows). Each $p_j \in P_{above}$ (respectively, $p_j \in P_{below}$) is connected to m_1 (respectively, m_2) (dark green arrows). (b) Each $v \in V_{\partial\mathcal{E}}$ is connected by directed xy -increasing edges (light green edges) to all the sides of $\partial C^{i,j}$	114
Figure 4.8	A free-space diagram in which the part of π_{st} in the forbidden-space is arbitrary small, compared to the length of T_1 and T_2 , and hence the quality of the optimal solution $Q^B(T_1, T_2)$ could be arbitrary small. . .	119
Figure 4.9	Illustration of the proof of Theorem 8. The path π_{st} intersecting the parameter cell $C^{i,j}$ is represented by the black curve. The L_1 -distance between a and b is represented by the dotted-dash line. The L_1 -distance between the Steiner points s_2 and s'_2 is represented by the dotted line. . .	121
Figure 5.1	Moving backward from a to b allows to walk on T_1 and T_2 and keeping the distance between moving objects less than ε during the walk. . . .	123
Figure 5.2	(a) Two polygonal curves, T_1 and T_2 , and the leash length, ε , are shown. Also the corresponding deformed free-space diagram is drawn. (b) Two paths in the free-space are drawn: an arbitrary path Π' (black dashed line) and an optimal path $\Pi \subset \mathcal{G}_v$ (red solid line).	126

Figure 5.3	A polygonal domain is constructed by replacing elliptic curves of the boundary of W by line segments.	131
Figure 6.1	Moving backwards from a_3 to b_3 allows to walk on T_1 and T_2 and keeping the distance between moving objects at most ε during the walks while the cost is minimized.	135
Figure 6.2	The corresponding weighted deformed free-space diagram of the given polygonal curves in Figure 6.1 is drawn. Π (the black dashed path) is an arbitrary path in W . $\Pi' \subset \mathcal{G}_w$ (the red solid path) is a path in W that realizes a pair of parameterizations which gives an optimal solution for WMBFD. Π'' (the blue dashed path) is a path in W that realizes the optimal solution for MBFD.	138
Figure 6.3	a) The visibility chain from a to c (the blue solid polygonal chain), $CC_a^c : \langle a, q_1, q_2, c \rangle$. b) The visibility chain from p'_z to p_{i+1} , $CC_{p'_z}^{p_{i+1}}$ (see Algorithm 4).	142
Figure 6.4	There are 16 cases for the combination of two directed segments. . . .	145
Figure 6.5	The segment from q_r to q_{r+1} is the first line segment in $CC_{p'_z}^{p_{i+1}}$ on the right side of $L_x^\perp$ that is x -decreasing. The segment from q_u to q_{u+1} is the first line segment in $CC_{p'_z}^{p_{i+1}}$ above of $L_y^\perp$ that is y -decreasing. . . .	147
Figure 6.6	a) The directed edge $e = \langle u_1 u_2 \rangle \in \mathcal{G}_w$ intersects a sequence of intervals on the boundary of the cells (red line segments). The edge is partitioned to three sub-edges, $\langle u_1 p_1, p_1 p_2, p_2 u_2 \rangle$. Each sub-edge is in a row or a column. b) The green dashed polygonal chain shows the xy -monotone path that is constructed in the first phase. The blue dotted polygonal chain shows the xy -monotone path that is in $\mathcal{G}_w'$	149

Figure 6.7	A row of a weighted free-space diagram is drawn. The boundary of the free-space of the row is highlighted by a red curve.	152
Figure 7.1	An embedding of a planar graph, H , a polygonal curve, T , and a length ε are given. The path $P^* = [v_1, v_3, v_4, a, v_4, b, v_4, v_5]$, in H , is a part of a solution to the map matching problem instance. The edges of H that P^* lies on, are illustrated in bold.	158
Figure 7.2	The free-space diagram $\mathcal{F}_\varepsilon(T, P^*)$ is drawn. W_P is the white area and B_P is the gray area.	160
Figure 7.3	The free-space surface for the example of Figure 7.1 is drawn from two different viewpoints in $3D$. The yellow line segments show the intervals on the cell boundaries. The red dashed polygonal curve is a path on the white-surface that realizes an optimal solution to our problem setting.	162
Figure 7.4	The free-space face $\mathcal{F}_i^j$ is drawn. The endpoints of the intervals, $\mathcal{FI}_i^j(\ell)$, are shown by black points and the Type 1 Steiner points are shown by squares.	163
Figure 7.5	An example of $\mathcal{T}_{j-1}^j$, for $j = 1$, is drawn. In this example, there are four intervals that are shown by yellow color. The black points show the interval endpoints and red balls show the Type 2 Steiner points. . . .	165
Figure 7.6	a) The result of unfolding the sequence of deformed free-space faces that are intersected by the red dashed polygonal curve in Figure 7.3. It is a $2D$ free-space diagram, $\mathcal{F}_\varepsilon(SF)$. The red dashed polygonal curve is shown after unfolding. b) Illustration of case 1 in the proof of Lemma 31.	167

Figure 7.7	A cell of the free-space surface is drawn. The red solid line segments show the four intervals on the boundary of the cell. The arcs show the edges in E' that connect every two adjacent vertices of $\mathcal{G}'$, on each interval. The dashed black line segments show the edges in E' that connect a vertex with its orthogonal projection on the opposite side of the cell. The dash dotted blue line segments show some of the edges that connect endpoints of the intervals. For simplicity, we did not draw all of them.	172
Figure 7.8	The three sub-cases in the proof of Case (2), Lemma 32.	173
Figure 8.1	Suppose it is not allowed to move backward on the first and second segment of T_2 . In this example the weight of moving backward is 1 everywhere. The path Π_1 in the free-space is the L_1 shortest path from s to t . However, it is not a feasible solution to this instance of the problem since there is a backward movement on the first segment of T_2 . The path Π_2 is a solution.	180

Chapter 1

Introduction

1.1 Introduction and Motivation

The amount of digital data that is gathered and now becoming accessible is massive. One source of a large amount of data includes tracking data of moving objects. These moving objects can range from persons (e.g., tourists and athletes) , vehicles (e.g., cars, planes and ships) to animals (e.g., migrating birds and fishes), to weather fronts (e.g., hurricanes). These movements have many aspects, including the geometry and the time of the movement. This thesis will focus specifically on the geometry aspects. More precisely, suppose $\mathcal{S}$ is a geometric space. A movement is modeled by a *geometric path*.

Definition 1 (Geometric Path). *A geometric path (or a trajectory) is a continuous function, a parameterization $f : [0, 1] \rightarrow \mathcal{S}$, where $f(0)$ is the starting point and $f(1)$ is the ending point of the geometric path, and $f(t)$ is the point $\in \mathcal{S}$, on the geometric path, for the parameterization value $t \in [0, 1]$.*

A geometric path sampled by a finite sequence of points (i.e., vertices), connected by line segments in order (i.e., edges), is called a *discrete geometric path* or a *polygonal curve*.

Definition 2 (Discrete Geometric Path (Polygonal Curve)). *A discrete geometric path is a continuous function $f : [0, n] \rightarrow \mathcal{S}$, $n \in \mathbb{N}$, such that f is affine in interval $[i, i+1]$ (i.e., forms a line segment), $i = 0, \dots, n-1$. In addition, $\langle f(0), \dots, f(n) \rangle$ is the sequence of its vertices (or corners).*

Discrete geometric paths are used in many fields of application (e.g., robotics, geographical information systems, pattern recognition) to approximate movement in a geometric space. There are many approaches to analyze discrete geometric paths. The following is a list of some important questions related to the analysis of discrete geometric paths:

- How to characterize and then compute a shortest discrete geometric path to navigate in a geometric space?
- How to measure the similarity of two or more discrete geometric paths?
- How to cluster and select a representative member within a group of discrete geometric paths?
- How to compress (i.e., simplify) a discrete geometric path while preserving “most” of the information?
- How to find a discrete geometric path that has a specific topological or geometric property?
- How to spatially relate a group of paths?

These questions are addressed in different fields by different tools. In order to have a precise problem definition, it is important to interpret the questions within a context. A broad collection of problems is defined by considering various parameters and contexts. A common list of parameters and contexts includes:

- The underlying distance function (e.g., metrics such as L_1 , L_2 , L_∞)
- Cost function of a discrete geometric path
- The geometric space partitioning (e.g., triangulation, grid, polygon, polyhedra surface, etc.)

- Online vs. offline inputs
- Weighted vs. unweighted geometric space
- Exact vs. approximate algorithms
- Dynamic vs. static environment

Remark. In the remainder of this thesis, we use the terms *path* or *polygonal curve* to refer to a discrete geometric path.

This thesis will focus on the first two questions listed above. First, we discuss how to characterize and then compute a shortest path. This problem is known as the geometric shortest path problem and is defined formally in Section 1.1.1. Second, we discuss the Fréchet distance problem and its variants. The Fréchet distance is a widely used similarity measure between two polygonal curves, which is defined formally in Section 1.1.2.

1.1.1 Geometric Shortest Path Problem

In the first question, we are dealing with the *geometric shortest path* problem. The geometric shortest path problem is one of the fundamental problems in Computational Geometry and related fields. In this problem, a partitioning of the input geometric space and two points, source, s , and target, t , are given. A partitioning of a geometric space is a collection of regions in the geometric space that the union of the regions covers the geometric space and the intersection of the interior of any two regions in the collection is empty. Each region of the input partitioning has an associated weight. The Euclidean shortest path problem is a special case of the geometric shortest path, when the weight of the regions that the path could go through is 1 and the weight of the regions that the path is not allowed to traverse is set to infinity. In the Euclidean shortest path problem the underlying distance function is the Euclidean metric. In this thesis, if the weight of the shared boundary between two

regions is not mentioned explicitly, it is set to the minimum of the adjacent regions' weight. The output is a *shortest path* from s to t , which is a path with minimum “cost”. The cost function will be defined formally later in this section. In order to understand the cost function it is important to first explore other parameters of the problem. These parameters allow us to define different variants of the problem. The following is an overview of the five most commonly studied parameters for the geometric shortest path problem:

I. Geometry of the input partitioning. The input of the geometric shortest path problem could be e.g., a simple polygon, a rectilinear simple polygon, a polygonal domain, a triangulation, or a partitioning induced by an arrangement of lines.

Definition 3. [1] *A polygon with n vertices, $n \geq 3$, is a cyclically ordered sequence of n points, $p_0, \dots, p_{n-1}$, (the vertices) together with the line segments (the edges), e_i , $i = 0, \dots, n-1$, determined by pairs of vertices p_i and p_{i+1} . In this definition, the subscript i is mod n .*

Definition 4. [2] *A simple polygon is a closed region of a plane that is enclosed by a simple (i.e., non-self-intersecting) polygon.*

Definition 5. [2] *A rectilinear simple polygon (also known as orthogonal simple polygon) is a simple polygon in which each edge is parallel to either the x - or the y -axis.*

Definition 6. [3] *We say a region R is (pathwise) connected if for every two points, a and b , in R , there is a continuous function f from $[0, 1]$ to R , such that $f(0) = a$ and $f(1) = b$.*

Definition 7. [3] *We say a connected region R is simply connected if any closed curve that lies in R can be shrunk to a point continuously in R (Figure 1.1a). If R is connected but not simply, it is said to be multiply connected (Figure 1.1b).*

Definition 8. [4, 129] *A polygonal domain with n vertices and h holes is a multiply-connected bounded region whose boundary is a union of n line segments (i.e., edges) that meet on*

Figure 1.1: a) A simply connected region. b) A multiply connected region. c) A polygonal domain with 17 edges and 2 holes.

vertices. These n line segments are organized in $h+1$ closed disjoint simple polygonal chains (Figure 1.1c).

Definition 9. [5] A triangulation, T , of n points (vertices) in a plane, is any maximal set of pairwise disjoint straight line segments (edges) between vertices. These edges meet on the vertices.

Definition 10. [6] Suppose L is a set of lines in a plane. An arrangement of L , $\mathcal{A}(L)$, is the induced subdivision of the plane by L . It consists of vertices, edges and faces.

II. Value range for weights. This parameter in the geometric shortest path problem is used to assign a cost for traveling in a specific region. If there is no assumption about the cost of traveling in regions, a weight equal to one is assigned to all of them.

- A basic weight assignment is as follows. The weight of 1 is assigned to the regions that the path could go through. The weight of infinity is assigned to the regions (obstacles) that the path is not allowed to travel through.
- A more complex weight assignment is a piecewise constant function that assigns positive real weights to regions. This helps to model more realistic applications. For example, in the computation of shortest paths in Geographic Information Systems (GIS), a

real-valued weight function allows users to include the slopes of the regions. Additionally, in robotics, the energy consumption of the robot in each region can be modeled in such a way. Furthermore, in Seismology, a real-valued function allows researchers to take the wave velocity factor of materials into account by assigning appropriate weights to regions.

III. The distance metric. The distance in geometric space is usually measured by a L_p metric, where p is a positive integer. Let $a = (a_1, \dots, a_d)$ and $b = (b_1, \dots, b_d)$ be two points in d -dimensional space. The definition of L_p in d -dimensional space is given in (1.1) and (1.2). The distance measured by the L_1 metric is known as the Manhattan distance. Also, the distance measured by the L_2 metric is called the Euclidean distance.

$$L_p(a, b) = \left(\sum_{i=1}^d |a_i - b_i|^p \right)^{\frac{1}{p}}, 1 \leq p < \infty \quad (1.1)$$

$$L_\infty(a, b) = \max_{i=1 \dots d} (|a_i - b_i|) \quad (1.2)$$

IV. Cost function of a path. The cost function in the geometric shortest path problem is usually defined as the length or the size of a path. The length and size of a path are defined as follows.

Definition 11 (Size and length of a path). *Let $\Pi = [s = x_0, x_1, \dots, x_k = t]$ be a path from point s to t , where x_i , $i = 0, \dots, k$ are its vertices. The size of Π is k . Also, assume that Π intersects m regions, $r_1, \dots, r_m$ and $w_j > 0$, $j = 1 \dots m$, are the weights associated to r_j , $j = 1 \dots m$. For a L_p metric, the length of Π , denoted by $|\Pi|_p$, is defined by (1.3), where $|s_j|_p$ is the length of $\Pi \cap r_j$ in the underlying L_p metric.*

$$|\Pi|_p = \sum_{j=1}^m |s_j|_p \cdot w_j \quad (1.3)$$

V. Dimension of the geometric space. It is possible to define the geometric shortest

path problem in different dimensions, i.e., 2D plane, 3D, and general d-dimensional spaces.

In addition to the above list of five parameters, the shortest path problem has also been discussed on graphs. However, the shortest path problem on graphs is not a focus of this thesis. We will however use the existing standard algorithms for shortest paths on graphs, as black boxes (see [7] for a discussion on shortest path on graphs). In the remainder of this thesis, the *shortest path problem* refers to the geometric shortest path problem, unless otherwise mentioned explicitly.

1.1.2 Fréchet Distance

Measuring the similarity between two polygonal curves is another fundamental problem in computational geometry. It poses challenges and is of high interest both from a practical and theoretical point of view. It is of practical relevance in areas such as pattern analysis [8, 9], matching [10, 12] and clustering [13]. Also, in the context of GIS the similarity of movement patterns, modeled by polygonal curves, has a variety of applications. These include animal behaviour, human movement, traffic management, sports scene analysis, and movement in abstract spaces [14, 15, 16]. In addition, it is of theoretical interest since the problems in this domain are fairly challenging and their solutions lead to innovative tools and techniques.

There are two types of similarity measures for polygonal curves: local and global. The local measures consider local features, e.g, vertices, of the input polygonal curves. The global measures consider a continuous parametrization of the input polygonal curves. *Bottleneck distance* is a local measure that is studied extensively in shape matching [10] and requires a bijective mapping between features of two polygonal curves. Assume P and Q are two sets of points that are representing the vertices of two input polygonal curves. Let $M(P, Q)$ be the set of all bijective mappings from P to Q . Then, the bottleneck distance, $b(P, Q)$ is defined in (1.4), where $L_p(.,.)$ is the L_p distance.

$$b(P, Q) = \min_{m \in M(P, Q)} \max_{a \in P} L_p(a, m(a)) \quad (1.4)$$

In practical settings, not all features in P need to have a corresponding feature in Q due to occlusion and noise. Typically, there is no bijective mapping between P and Q . Thus, a similarity measure that is often used is the *Hausdorff distance* [11]. The directed Hausdorff distance $h(P, Q)$ is defined in (1.5). The Hausdorff distance $H(P, Q)$ is the maximum of $h(P, Q)$ and $h(Q, P)$ (see (1.6)).

$$h(P, Q) = \max_{a \in P} \min_{b \in Q} L_p(a, b) \quad (1.5)$$

$$H(P, Q) = \max\{h(P, Q), h(Q, P)\} \quad (1.6)$$

It has been observed by [17, 18, 19] that measures which consider global features of the input curves (such as the *Fréchet distance*) often achieve more accurate results than local measures (such as the Hausdorff distance or the bottleneck distance). The reason is, a global measure takes continuous parametrization of the input curves into account. For this reason, the Fréchet distance is a widely used and established tool to measure and formalize the similarity between polygonal curves [22, 23, 25].

The Fréchet distance is typically illustrated via the person-dog metaphor. Assume that a person wants to walk along one curve and his/her dog on another. Each curve has a starting and an ending point. The person and the dog walk, from the starting point to the ending point along their respective curves. The two walks could have different varying (arbitrary large) speeds. The standard Fréchet distance is the minimum leash length required for the person to walk the dog without backtracking. More formally, let $T_1 : [0, n] \rightarrow \mathbb{R}^2$ and $T_2 : [0, m] \rightarrow \mathbb{R}^2$ be two polygonal curves in $\mathbb{R}^2$. A *parameterization* of T_1 (resp. T_2) is a

continuous function α_1 (resp. α_2) from $[0, 1]$ to $[0, n]$ (resp. $[0, m]$), where $\alpha_1(0) = \alpha_2(0) = 0$ and $\alpha_1(1) = n$ (resp. $\alpha_2(1) = m$). We say a parametrization $\alpha(t)$ is monotone if $\alpha(t)$ is either increasing, for all $t \in [0, 1]$, or decreasing, for all $t \in [0, 1]$. Two monotone parameterizations, α_1 and α_2 , define, for each time $t \in [0, 1]$, a matching $(T_1(\alpha_1(t)), T_2(\alpha_2(t)))$ of one point on T_1 to exactly one point on T_2 and vice-versa. The required leash length for the two parameterizations is defined as the maximum Euclidean distance of two matched points, over all times. Then, the Fréchet distance $\delta_F(T_1, T_2)$ is defined as the infimum of the required leash lengths over all possible pairs of monotone parameterizations [22]:

$$\delta_F(T_1, T_2) := \inf_{\alpha_1, \alpha_2} \max_{t \in [0, 1]} \{L_2(T_1(\alpha_1(t)), T_2(\alpha_2(t)))\}, \quad (1.7)$$

where $L_2(\cdot, \cdot)$ is the Euclidean distance. The corresponding *Fréchet distance decision problem* asks if there exist two parameterizations for a given leash length ε , realizing a Fréchet distance between T_1 and T_2 that is upper bounded by ε . In other words, it asks if it is possible to walk your dog with a given leash of length ε , such that you and your dog stay on your own curves.

A variant of the standard Fréchet distance is the *weak Fréchet distance*, also known as non-monotone Fréchet distance [22]. In this variant, backtracking is allowed during the walks (i.e., the parameterizations, α_1 and α_2 , are not monotone necessarily) and the objective is to minimize the required leash length. Another variant of the Fréchet distance is the *discrete Fréchet distance* or the *coupling distance* introduced by Eiter and Mannila [39]. It provides an approximation for the Fréchet distance of two polygonal curves. Let $\langle x_0, \dots, x_n \rangle$ be the sequence of the vertices of a polygonal curve T_1 and $\langle y_0, \dots, y_m \rangle$ be the sequence of the vertices of another polygonal curve T_2 . A coupling C between T_1 and T_2 is defined as a sequence:

$$\langle (x_{i_0}, y_{j_0}), (x_{i_1}, y_{j_1}), \dots, (x_{i_k}, y_{j_k}) \rangle \quad (1.8)$$

such that (1) $i_0 = 0$, $j_0 = 0$, $i_k = n$, and $j_k = m$ (2) for all $\ell = 0, \dots, k$ we have $i_{\ell+1} = i_\ell$ or $i_{\ell+1} = i_\ell + 1$, and $j_{\ell+1} = j_\ell$ or $j_{\ell+1} = j_\ell + 1$. These two conditions guarantee that a coupling starts from pair (x_0, y_0) and ends at (x_n, y_m) . Also, the vertices of T_1 and T_2 are paired in order. The length, $|C|$, of a coupling C is defined in (1.9).

$$|C| = \max_{\ell=0, \dots, k} L_2(x_{i_\ell}, y_{j_\ell}) \quad (1.9)$$

Finally, the coupling distance between T_1 and T_2 is defined as follows.

$$\delta_{dF}(T_1, T_2) = \min\{|C|, \text{ where } C \text{ is a coupling between } T_1 \text{ and } T_2\} \quad (1.10)$$

It is proved in [39] that the difference between the standard Fréchet distance and the coupling distance of two polygonal curves is upper bounded by the Euclidean length of the longest edge of the two polygonal curves. In Figure 1.2 two examples are illustrated. In this figure, two pairs of polygonal curves, T_1 and T_2 , are drawn. In Figure 1.2a, the dashed line segment shows the Hausdorff distance between T_1 and T_2 and the dash dotted line segment illustrates the length of the standard Fréchet distance between them. In Figure 1.2b, the dash dotted line segment shows the standard Fréchet distance between T_1 and T_2 and the dashed line segment shows the coupling distance between them.

1.2 Thesis Outline and Contributions

In Chapter 2 and 3, we study the weighted region problem (WRP), which is to compute a shortest path on a weighted partitioning of a plane when the weights of regions are positive real numbers. Many of the practical applications of the shortest path problem in geographical information systems (GIS), robotics, seismology, among others, need more elaborate weight functions than equal weights.

Figure 1.2: a) The dashed line segment shows the Hausdorff distance between T_1 and T_2 and the dash dotted line segment shows the standard Fréchet distance between them. b) The dash dotted line segment shows the standard Fréchet distance between T_1 and T_2 and the dashed line segment shows the coupling distance between them.

Recent results show that WRP is not solvable in any algebraic computation model over the rational numbers (ACMQ^1) [81]. Therefore, it is unlikely that WRP can be solved in polynomial time. Research has thus focused on determining approximate solutions for WRP. To the best of our knowledge, nobody has studied the WRP when the input partitioning of space is induced by an arrangement of lines, $\mathcal{A}$. In this problem, an arrangement of lines $\mathcal{A}$ with an associated weight for each induced face, a source s , and a target t , are given. The weights of the faces are positive real numbers. The objective is to find a weighted shortest path, π_{st} , from s to t . Existing approximation algorithms for WRP work within bounded regions (typically triangulations). To apply these algorithms to unbounded regions, such as arrangements of lines, there is a need to bound the regions. Here, we present a minimal region that contains π_{st} , called SP-Hull of $\mathcal{A}$. It is minimal in the sense that for any arrangement of lines $\mathcal{A}$, it is possible to assign weights to the faces of $\mathcal{A}$ and choose s and t such that π_{st} is arbitrary close to the boundary of the SP-Hull of $\mathcal{A}$. It is a closed polygonal region that is independent of the weights assignment. We show that SP-Hull can be constructed in $\mathcal{O}(n \log n)$ time, where n is the number of lines in the arrangement. As a direct consequence

¹The ACMQ can compute exactly any number that can be obtained from the rationals $\mathbb{Q}$ by applying a finite number of operations from $+$, $-$, $\times$, $\div$, $\sqrt[k]{}$, for any integer $k \geq 2$.

we obtain an approximation shortest path algorithm for weighted arrangements of lines. The proposed technique in this chapter can be used in other types of partitions which also have unbounded regions (e.g., farthest point Voronoi diagrams).

In addition, approximate solutions for WRP typically show qualitatively different behaviors. We first formulate two qualitative criteria for weighted short paths. Then, we show how to produce a path that is quantitatively close-to-optimal and qualitatively satisfactory. More precisely, we propose an algorithm to transform any given approximate linear path into a linear path with the same (or shorter) weighted length for which we can prove that it satisfies the required qualitative criteria. This algorithm has linear time complexity in the size of the given path. At the end, we discuss our experiments on several triangular irregular networks (TINs) from Earth’s terrain. The results show that the proposed algorithm could save, on average, 51% in query time and 69% in memory usage, in comparison with the existing method.

In the context of similarity measures between polygonal curves, we study some variants of the Fréchet distance. While the Fréchet distance is a well-studied and commonly used measure to capture the similarity of polygonal curves, it exhibits a high sensitivity to the presence of outliers. Since the presence of outliers is a frequently occurring phenomenon in practice, a robust variant of Fréchet distance is required which absorbs outliers. We study such a variant in Chapter 4. In this variant, our objective is to minimize the length of the sub-curves of two polygonal curves that need to be ignored (MinEx problem), or alternately, maximize the length of sub-curves that are preserved (MaxIn problem), to achieve a given Fréchet distance. An exact solution to one problem would imply an exact solution to the other problem. However, it is shown that these problems are not solvable by radicals over $\mathbb{Q}$ and that the degree of the polynomial equations involved is unbounded in general [49]. This motivates the search for approximate solutions. We present an algorithm which approximates, for a given input parameter δ , optimal solutions for the MinEx and MaxIn

problems up to an additive approximation error δ times the length of the input curves. The resulting running time of our algorithm is $\mathcal{O}\left(\frac{n^3}{\delta} \log\left(\frac{n}{\delta}\right)\right)$, where n is the total number of points defining the input polygonal curves.

In Chapter 5, we propose a new measure to capture similarity between polygonal curves, called the *minimum backward Fréchet distance* (MBFD). It is a natural optimization on the weak Fréchet distance, a variant of the well-known Fréchet distance. More specifically, for a given threshold ε , we are searching for a pair of walks for two entities on the two input curves, T_1 and T_2 , such that the sum of the length of backward movements is minimized and the distance between the two entities, at any time during the walk, is less than or equal to ε . Our algorithm detects if no such pair of walks exists. This natural optimization problem appears in applications (e.g., GIS, mobile networks, and robotics). We provide an exact algorithm with a time complexity of $\mathcal{O}(n^2 \log n)$ and a space complexity of $\mathcal{O}(n^2)$, where n is the maximum number of segments in the input polygonal curves.

Furthermore, in Chapter 6, we generalize MBFD to capture scenarios when the cost of backtracking on the input polygonal curves is not homogeneous. More specifically, each edge of T_1 and T_2 has an associated (possibly different) non-negative weight. The cost of backtracking on an edge is the Euclidean length of backward movement on that edge multiplied by the corresponding weight. The objective is to find a pair of walks that minimizes the sum of the costs on the edges of the curves, while guaranteeing that the curves remain at weak Fréchet distance ε . We propose an exact algorithm whose run time and space complexities are $\mathcal{O}(n^2 \log^{3/2} n)$.

In Chapter 7, we propose a geometric algorithm for a map matching problem in which the “walking length” is minimized. More specifically, in this problem, we are given a planar graph, H , with a straight-line embedding in a plane, a directed polygonal curve, T , and a distance value $\varepsilon > 0$. The task is to find a path, P , in H , and a parameterization of T , that minimize the sum of the length of walks on T and P whereby the distance between the

entities moving along P and T is at most ε , at any time during the walks. It is allowed to walk forwards as well as backwards on T and edges of H . We propose an algorithm with $\mathcal{O}(mn(m+n)\log(mn))$ time complexity and $\mathcal{O}(mn(m+n))$ space complexity, where m (n , respectively) is the number of edges of H (of T , respectively). As we show, the algorithm can be generalized to work for weighted non-planar graphs within the same time and space complexities.

At the end of this thesis, we propose open problems and discuss future work. The contributions of this thesis are summarized in the following list:

- We propose an approximation algorithm for the weighted region problem when the input partitioning of space is an arrangement of lines. To the best of our knowledge, this problem has not been studied previously. This result is published in [48] and it is joint work with Anil Maheshwari and Jörg-Rüdiger Sack.
- We formulate two qualitative criteria for weighted shortest paths. Then, we show how to produce a path that is quantitatively close-to-optimal and qualitatively satisfactory. The results of our experiments show that, on average, 51% in query time and 69% in memory usage could be saved, compared to the existing method. This result is submitted for publication and it is joint work with Anil Maheshwari, Jörg-Rüdiger Sack, and Christian Scheffer.
- We study the Fréchet distance in the presence of outliers. We propose an algorithm which approximates, for a given input parameter δ , optimal solutions up to an additive approximation error δ times the length of the input curves. The resulting running time is $\mathcal{O}\left(\frac{n^3}{\delta} \log\left(\frac{n}{\delta}\right)\right)$, where n is the total number of points defining the input polygonal curves. This result is published in [49] and it is joint work with Jean-Lou De Carufel, Anil Maheshwari, Jörg-Rüdiger Sack, and Christian Scheffer.
- We propose a new measure to capture similarity between polygonal curves, MBFD. We

provide an exact algorithm with time complexity of $\mathcal{O}(n^2 \log n)$ and space complexity of $\mathcal{O}(n^2)$, where n is the total number of points defining the input polygonal curves. This result is published in [50] and it is joint work with Anil Maheshwari, Jörg-Rüdiger Sack, and Christian Scheffer.

- We generalize MBFD to capture scenarios when the cost of backtracking on the input polygonal curves is not homogeneous. We propose an exact algorithm whose run time and space complexity is $\mathcal{O}(n^2 \log^{3/2} n)$, where n is the total number of points defining the input polygonal curves. This result is published in [51] and it is joint work with Anil Maheshwari and Jörg-Rüdiger Sack.
- We propose a geometric algorithm for a map matching problem that minimizes the walking length. Our algorithm has $\mathcal{O}(mn(m+n) \log(mn))$ time complexity and $\mathcal{O}(mn(m+n))$ space complexity, where m is the number of edges of the input graph and n is the number of edges of the input polygonal curve. The algorithm can be generalized to work also for weighted non-planar graphs within the same time and space complexities. This result is published in [52] and it is joint work with Anil Maheshwari and Jörg-Rüdiger Sack.

In the remainder of this chapter, we mention some related work.

1.3 Shortest Path Literature Review

1.3.1 Shortest Path in Polygons

Let P be a simple polygon, and s and t be two points inside, or on the boundary of, P . The basic shortest path problem is to find an Euclidean shortest path from s to t that lies inside (or on the boundary of) P . Linear time algorithms have been developed to solve this shortest path problem [54, 56, 55]. All of these algorithms start with a triangulation,

$T(P)$, of the input polygon P . Chazelle [53] proposed a deterministic linear time algorithm to triangulate simple polygons. Although this algorithm is linear, it is very complicated and no successful implementation has yet been reported. Later, a randomized algorithm to triangulate a polygon with linear expected time has been proposed by Amato et al. [128]. It is still open to design a linear time algorithm for computing a shortest path between two points in a simple polygon, without using a (complicated) linear time triangulation algorithm [4]. In [54], Chazelle proposed a linear time algorithm for the shortest path problem, when a triangulation of the polygon is given. The dual graph of a triangulation is defined as follows. Each triangle is represented by a node in the graph. There is a link (edge) between two nodes if the corresponding triangles share an edge of the triangulation. The dual graph of the triangulation of a simple polygon is always a tree. Assume v_s (v_t) is the node representing the triangle that s (t) lies in. The algorithm finds, first, a path in the tree that connects v_s to v_t . This path gives a sequence of triangles, called sleeve, that connect s to t and every two adjacent triangles share an edge. In the second step, the algorithm collapses the sleeve to a shortest path, by building some structures called “funnel”. A funnel is defined as follows. Assume $\overline{ab}$ is a diagonal in P (Figure 1.3). The funnel from s to $\overline{ab}$, $F(s, \overline{ab})$, is defined as the union of the shortest path from s to a , π_{sa} , and the shortest path from s to b , π_{sb} . The two shortest path, π_{sa} and π_{sb} , may share a subpath. The vertex in which π_{sa} and π_{sb} separate from each other is called the root, r . Also, $\overline{ab}$ is called the base of the funnel. Suppose the algorithm processed all the triangles in the sleeve up to $\triangle abc$ (see Figure 1.3). In the processing of $\triangle abc$ the funnel $F(s, \overline{ab})$ splits into $F(s, \overline{ac})$ and $F(s, \overline{bc})$. The splitting operation of funnels plays an important role in the performance of the algorithm. This operation is done by choosing the tangent line segment from c to one of π_{sa} and π_{sb} (the dashed line in Figure 1.3). One of the two new funnels is useful to find the shortest path to t . The useful funnel is the one that shared the base with the next triangle in the sleeve. Therefore, the other one will be deleted.

Figure 1.3: Split operation of a funnel, $F(s, \overline{ab})$, with root of r and base $\overline{ab}$.

A useful structure for answering shortest path queries is the shortest path map. A shortest path map, $\text{SPM}(s)$, for a source point s , encodes shortest paths from s to every point inside P . If π_{sx} is the shortest path from s to an arbitrary point x in P , the predecessor of x is defined as the vertex that is preceding x in π_{sx} . The $\text{SPM}(s)$ is a subdivision of P such that all the points in one region have the same predecessor in a shortest path from s . It is proved that $\text{SPM}(s)$ has a linear complexity [56]. In general, the boundary of regions of $\text{SPM}(s)$ are line segments or hyperbolic arcs.

Guibas et al. [55] proposed a linear time algorithm to construct $\text{SPM}(s)$ for a given source point s inside a polygon to answer single point shortest path queries. Using $\text{SPM}(s)$, after preprocessing the $\text{SPM}(s)$ for point location queries, it is possible to answer shortest path queries from s to any query point in P in $\mathcal{O}(\log n + k)$ time, where k is the size of the output. Also, a shortest path from s to each vertex of P is computed in preprocessing step. When a query arrives, the region of $\text{SPM}(s)$ that contains the query point, $region_i$, is located in $\mathcal{O}(\log n)$ time. Then, finding the shortest path will be done by concatenation of the shortest path from s to a vertex of $region_i$ with the direct line segment from that vertex to the query point. This result could be extended for the case that the source point s is also included in the query [57]. In other words, Hershberger has shown that we can preprocess

a simple polygon P in linear time and construct a data structure of size $\mathcal{O}(n)$, to answer shortest path queries for any two points $s, q \in P$ in $\mathcal{O}(\log n + k)$ time.

As we mentioned, for a given simple polygon P , it is possible to find the shortest path between any two points $x, y \in P$ in linear time. The problem is to determine the maximum shortest path length for all pairs of points $x, y \in P$. This problem is known as the geodesic diameter problem. Hershberger and Suri [58] proposed a linear time algorithm for this problem. They presented an $\mathcal{O}(n)$ time algorithm for computing row-wise maxima, or minima, of a totally-monotone matrix whose entries are shortest-path distances between pairs of vertices in a simple polygon. A matrix M is totally-monotone if, for any $i < j$ and $k < l$, (1.11) holds. Their result improved the time complexities of some other algorithms for solving several important questions in computational geometry.

$$M(i, k) < M(i, l) \Rightarrow M(j, k) < M(j, l) \quad (1.11)$$

A more general version of the shortest path problem is set in polygonal domains (see Definition 8). In this version of the problem, a polygonal domain, D , and two points $s, t \in D$, are given. D has n vertices and h holes. The output is a shortest path from s to t that lies inside D (i.e., the path is not allowed to pass through any of the holes).

To compute a shortest path in a polygonal domain, it is possible to use the visibility graph $VG(D) = \langle V, E \rangle$. It is defined as follows. The vertices of $VG(D)$ are the vertices of D . Then, two vertices of the graph, v and u , are linked by an edge if the line segment $\overline{vu}$ lies inside or on the boundary of D (i.e., they are visible to each other). There is an algorithm to construct $VG(D)$ in $\mathcal{O}(n^2)$ [60]. Also, Pocchiola and Vegter [61] proposed an optimal output-sensitive algorithm to compute the visibility graph in time $\mathcal{O}(|E| + n \log n)$ which uses $\mathcal{O}(n)$ space. Furthermore, it is proved that a shortest path uses only edges of $VG(D)$. Therefore, when $VG(D)$ is constructed, we can find a shortest path from s to t by

Dijkstra’s algorithm [62, 63] in $\mathcal{O}(|E| + |V| \log |V|)$ time.

A visibility graph may have a quadratic number of edges. To achieve a sub-quadratic running time, an alternative is to construct the shortest path map for a source point s , $\text{SPM}(s)$. Mitchell et al. [59, 64] introduced a method called “continuous Dijkstra” to build $\text{SPM}(s)$ for polygonal domains (and also polyhedral surfaces). As the name of the algorithm suggests, it is similar to Dijkstra’s algorithm for shortest paths in graphs. This algorithm simulates the wavefront propagation from a source point s . If s is the source of a wave that propagates with a constant rate in the polygonal domain, the wavefront at time t_0 is the set of all points of D that have similar length of the shortest path from s at time t_0 . We know that shortest paths from s to points on an edge of D have different lengths. However, it is possible to divide each edge of D into some intervals such that shortest paths from s to all the points in one interval have similar combinatorial structure. Such an interval is called an *interval of optimality*. In other words, for all x and y ($x \neq y$) in one interval of optimality, shortest paths from s to x and y share all the edges, except the last one. To find the intervals of optimality, during the steps of the algorithm, for each edge, a set of candidate intervals are stored. The candidate intervals do not overlap and shrink during the time when new candidate intervals are created by propagation through another part of the polygonal domain. Each candidate interval could be broken down into zero, one or some intervals of optimality. Assume a triangulation of D and a source point s are given. The continuous Dijkstra algorithm starts from s and builds candidate intervals on the edges of the triangle that s lies in. Then, at each step, it propagates one candidate interval to the adjacent triangles. The candidate intervals are created in the order of their distances from s . The challenging part of the algorithm is to keep the combinatorial structure of the wavefront updated [4]. Mitchell proposed a technique in [65] to handle this challenge in overall running time of $\mathcal{O}(n^{3/2+\varepsilon})$, for any fixed $\varepsilon > 0$, using $\mathcal{O}(n)$ space. Alternatively, Hershberger and Suri [56] gave an algorithm that runs in $\mathcal{O}(n \log n)$ time and $\mathcal{O}(n \log n)$ space.

We mentioned that the geodesic diameter problem in simple polygons can be solved in linear time. This problem can also be set inside polygonal domains. Suppose a polygonal domain with n vertices and h holes is given. The problem is to determine the maximum shortest path length for all pairs of points inside the polygonal domain. To the best of our knowledge, the only polynomial time algorithm for the geodesic diameter problem inside a polygonal domain has $\mathcal{O}(n^{7.73})$ running time [129].

1.3.2 Minimum Link Path

In the minimum link path problem the objective is to find a path between s and t that avoids obstacles (regions with the weight of infinity) so that the number of links (i.e., number of turns) in the path is minimized. In other words, the cost function of a path that we want to minimize is the size of the path (see Definition 11). This problem has been well studied due to its applications in VLSI. Suri gave an algorithm [66] to find minimum link path in a simple polygon in $\mathcal{O}(n)$ time. Mitchell et al. [67] showed how to find the minimum link path in a polygonal domain D in $\mathcal{O}(|E|\alpha(n)\log^2 n)$ time, where $|E|$ is the number of edges in $VG(D)$ and $\alpha(n)$ is the inverse of Ackermann's function (and it grows very slowly). The 3SUM-hardness of the minimum link path problem in a polygonal domain has recently been published by Mitchell et al. [68]. However, it is still open if one can solve this problem in quadratic time.

A simpler version of the problem is the rectilinear minimum link path in a rectilinear polygonal domain. A path (resp. polygonal domain) is rectilinear if each of its edges is parallel to one of the two coordinate axes. Das and Narasimhan have studied this problem [69]. They proposed an $\mathcal{O}(n\log n)$ time algorithm to find such a path. In addition, this problem has been studied in 3D by Drysdale III et al. [70] and they proposed an $\mathcal{O}(n^{5/2}\log n)$ time algorithm. Many other versions of the minimum link path problem are surveyed in [71] by Maheshwari et al.

1.3.3 Manhattan Shortest Path

In the Manhattan (distance) shortest path problem, a polygonal domain D with n vertices and h holes and two points, the source s and the target t , are given. The goal is to find a L_1 (Manhattan) shortest path from s to t (i.e., a path from s to t with minimum length in the L_1 metric). The L_1 metric is defined in (1.1), for $p = 1$. A straightforward algorithm is to build a visibility graph and find a shortest path from s to t by Dijkstra's algorithm. The visibility graph may have a quadratic number of edges. In [72], Clarkson et al. proposed an algorithm to construct a sparse version of the visibility graph for the L_1 metric. It is guaranteed that this sparse graph contains a Manhattan shortest path from s to t . The number of nodes and edges in this graph are $\mathcal{O}(n \log n)$. One can use Dijkstra's algorithm on this graph to find the Manhattan shortest path in $\mathcal{O}(n \log^2 n)$ time. Later, Clarkson et al. improved the time complexity of computing the Manhattan shortest path to $\mathcal{O}(n \log^{3/2} n)$ [73]. The property that was used by Clarkson et al. (and many other authors) to reduce the complexities of algorithms for this problem is as follows. Let a, b and c be three points in the Cartesian coordinate system in $\mathbb{R}^2$ and their x -coordinates (resp. y -coordinates) satisfy $a_x \leq b_x \leq c_x$ (resp. $a_y \leq b_y \leq c_y$). Then, concatenation of a L_1 shortest path from a to b and a L_1 shortest path from b to c is a L_1 shortest path from a to c [72, 73, 75].

To improve the algorithm proposed by Clarkson et al., the continuous Dijkstra could also be used to build $\text{SPM}(s)$ for the L_1 metric [64]. The $\text{SPM}(s)$ for L_1 can be computed in $\mathcal{O}(n \log n)$ time, using $\mathcal{O}(n)$ space. A special property of the Manhattan metric is that, in this case, the wavefront of $\text{SPM}(s)$ is piecewise linear [4]. In recent work [76], Chen and Wang showed how to compute the L_1 shortest path in a polygonal domain, using shortest path maps, in $\mathcal{O}(n + h \log h)$ time and $\mathcal{O}(n)$ space, where n is the number of vertices of the input polygonal domain and h is the number of holes.

The query version of the Manhattan shortest path problem is also well studied. In a polygonal domain, two-point queries under the Manhattan metric have been studied in [77].

Chen et al. have shown that, by a preprocessing of $\mathcal{O}(n^2 \log^2 n)$ time and $\mathcal{O}(n^2 \log n)$ space, two-points queries can be answered in $\mathcal{O}(\log^2 n)$ time. In a rectilinear simple polygon, Lingas et al. [78] proposed an optimal algorithm that achieves $\mathcal{O}(\log n + k)$ query time using linear preprocessing time and space; where k is the number of links in the output. To achieve a sub-quadratic preprocessing time, Arikati et al. [79] have proposed a general framework to obtain an approximation for two-point query in a polygonal domain for any L_p . A result of this framework is an algorithm that obtains a $(1 + \varepsilon)$ -approximation for a Manhattan shortest path with logarithmic query time.

1.3.4 Weighted Region Problem (WRP)

Let T be a partitioning of $\mathbb{R}^2$ (e.g., a triangulation or a partitioning induced by an arrangement of lines) and f be a piecewise constant function that assigns a positive real weight to each region of T . Also, two points, the source s and the target t , in $\mathbb{R}^2$, are given. The desired output is a shortest path from s to t , which is a path with minimum length (as defined in Definition 11) [80]. This problem was introduced by Mitchell and Papadimitriou [80]. They discussed the problem when the input partitioning, T , is a triangulation and the underlying metric for measuring the length of a path (Definition 11) is L_2 . They proposed a $(1 + \varepsilon)$ -approximation algorithm that has a time complexity of $\mathcal{O}(n^8 L)$, where n is the number of triangles in T , $L = \log(nNW/\varepsilon)$, N is the largest integer coordinate of the any vertex of T , and W is the maximum weight of the triangles. The complexity class of the problem is still unknown (i.e., is it in the class of polynomial-time problems or NP-hard?). Recently, it has been proven in [81] that WRP is not solvable in any *Algebraic Computation Model over the Rational Numbers* (ACM $\mathbb{Q}$). Because of the difficulty of the problem, approximation algorithms are necessary and appealing. Therefore, several approximation algorithms are proposed for weighted triangulations (cf. [80, 82, 95, 96]). The general idea of these algorithms is to discretize the underlying geometric space. One such technique, is

to build a discretization graph, $G = \langle V(G), E(G) \rangle$, by positioning Steiner points (either on the edges of the input triangulation or inside the triangles). The set $V(G)$ consists of the vertices of the input triangulation and the Steiner points. Then, pairs of vertices in $V(G)$ are linked to form the edges in $E(G)$. Different schemes are proposed for positioning Steiner points and linking them [82, 95]. We discuss some of them later in this section. At the end, the approximation solution is achieved by finding a shortest path in G , by using well-known combinatorial algorithms (e.g., the modified Dijkstra’s algorithm in [82], BUSHWHACK [96]).

In [95], Lanthier et al., proposed some schemes for placing Steiner points on the edges of the input triangulation. In their proposed “Interval Scheme”, the average number of Steiner points per edge, i , is given as an input. The interval between every two adjacent Steiner points is chosen such that the average number of Steiner points per edge is i . This interval distance is fixed for all edges. Afterwards, for each triangle, all the Steiner points on its edges (including the vertices of the triangle) are interconnected to create edges of the discretization graph G (i.e., a complete graph for each triangle). Their method has an additive approximation bound. They reported that, in practice, on average six Steiner points per edge in the Interval Scheme suffice to obtain a close-to-optimal approximation.

To the best of our knowledge, still no fully polynomial-time approximation scheme (FP-TAS) is known for WRP. There are some $(1 + \varepsilon)$ -approximation algorithms for this problem [82, 96] i.e., the length of the result is guaranteed to be no more than $(1 + \varepsilon)$ times the length of a shortest path. Since in the complexity of most of the existing algorithms different factors such as weights and aspect ratios of the regions (e.g., triangles) are involved, it is not obvious how to make a fair comparison between their complexities. Aleksandrov et al. [82] provided a table that compares the existing approximation algorithms, based on the integer bounds on the coordinates of the vertices of the regions, and maximum and minimum weights of the regions. They also proposed a $(1 + \varepsilon)$ -approximation for the shortest path problem on any

weighted polyhedral surface $\mathcal{P}$. This algorithm has the complexity of $\mathcal{O}(C(\mathcal{P})\frac{n}{\sqrt{\varepsilon}}\log\frac{n}{\varepsilon}\log\frac{1}{\varepsilon})$ time, where $C(\mathcal{P})$ captures some geometric parameters and the weights of the faces of $\mathcal{P}$. Also, Reif and Sun [96], achieved the time complexity of $\mathcal{O}(\frac{n}{\varepsilon}(\log\frac{1}{\varepsilon} + \log n)\log\frac{1}{\varepsilon})$. They used the “BUSHWACK” algorithm for finding a shortest path in the discretization graph that is built by using the scheme proposed in [101]. For an extensive comparison table between the approximation algorithms for shortest path problems on convex/non-convex weighted/unweighted polyhedral surfaces refer to [82].

Now, consider a two-point query version of the problem, in which the input is a polyhedral surface $\mathcal{P}$ consisting of n triangular faces with positive weights. The goal is to find a path on $\mathcal{P}$ from a source query point to a target query point whose length is minimized (the length is defined in Definition 11, where the underlying metric is L_2). Djidjev and Sommer [99] showed that for any $0 < \varepsilon < 1$, there exists a data structure called distance oracle, that can answer $(1+\varepsilon)$ -approximate point-to-point distance queries in $\mathcal{O}(\varepsilon^{-1}\log(1/\varepsilon) + \log\log n)$ time per query. The distance oracle has size $\mathcal{O}(n\varepsilon^{-3/2}\log^2(n/\varepsilon)\log(1/\varepsilon))$ and is computable in time $\mathcal{O}(n\varepsilon^{-2}\log^3(n/\varepsilon)\log^2(1/\varepsilon))$.

In [102], WRP with a constraint on the number of links (as defined below) has been studied and some experimental results are provided. In this problem, in addition to a weighted triangulation and s and t , a number k is given as an input. The goal is to compute a minimum length path from s to t , subject to the constraint that the path has $\mathcal{O}(k)$ links. They proposed an algorithm to generate a path of length at most $(1 + \varepsilon)$ times the k -link shortest path, while using at most $2k - 1$ (i.e., $\mathcal{O}(k)$) links. They used a technique similar to those described in [82, 95, 96]. First, their algorithm builds a discretization graph G . The time complexity of their algorithm to build G is $\mathcal{O}(n(\delta n)^4)$, where δ is the maximum number of Steiner points on an edge of the input triangulation. Once the graph has been constructed, dynamic programming is used to find a shortest path in G in $\mathcal{O}(k(\delta n)^2)$ time. It is guaranteed that a shortest path in G has a length of at most $(1 + \varepsilon)$ times the geometric

shortest path while using at most $\mathcal{O}(k)$ links.

As we mentioned, obstacles can be modeled in the WRP by regions with weights of infinity. A homotopy class of paths is a set of paths that can be continuously deformed to each other without passing over any obstacle [97]. Cheng et al. [97] proposed an algorithm that for a given path, Π_{st} , from s to t , in weighted regions with obstacles, and an error tolerance $\varepsilon \in (0, 1)$, computes a path from the same homotopy class of Π_{st} , with length at most $1 + \varepsilon$ times the optimum in that class.

In a recent work [98], Jaklin et al. discussed the WRP when weighted regions are cells of a grid. The main motivation of this work is the practical perspective of grids and widely usage of grids in path-following strategies in gaming applications and crowd simulation.

1.3.5 Shortest Path in 3D

Assume $\mathcal{D}$ is a connected polyhedral domain in 3D Euclidean space. $\mathcal{D}$ is partitioned into n tetrahedra such that the union of these tetrahedra is $\mathcal{D}$ and the intersection of any two tetrahedra is either a face, or an edge, or a vertex, or empty. Each tetrahedron has an associated positive weight in $\mathbb{R}$. The length (i.e., cost) of a path in $\mathcal{D}$ is defined as the sum of the Euclidean lengths of the sub-paths within each intersected tetrahedron multiplied by the corresponding weight of that tetrahedron. We are interested in finding a shortest path from s to t in $\mathcal{D}$. This problem is called the weighted shortest path problem in 3D (called WSP3D) [103]. Note that when the weight of each tetrahedron is either one or infinity (i.e., obstacle), then the problem is the standard Euclidean shortest path problem in 3D (called ESP3D).

In a polyhedral domain, the number of combinatorially distinct shortest paths from s to t may be exponential in the input size. Motivated by this fact, Canny and Reif [100] proved that the ESP3D under L_p metric, for any $p > 1$, is NP-hard. Also, Mitchell and Sharir [112], proved that even computing Euclidean shortest paths among stacked axis-aligned rectangles

in 3D is NP-complete. They also proved that the Manhattan shortest paths in 3D among disjoint balls is NP-complete. Their positive result is that computing a Manhattan shortest path between two given points, on or above a polyhedral terrain, is polynomial. A polyhedral terrain, $\mathcal{T}$, is a polyhedral surface in 3D for which it is possible to project $\mathcal{T}$ onto a plane without creating any new intersection between the projection of the edges [112]. Based on Mitchell’s and Sharir’s work, Zarrabi-Zadeh [113] proposed an algorithm that, for any $p \geq 1$, computes a $(c + \varepsilon)$ -approximation to a L_p -shortest path that stays on or above the given polyhedral terrain. The time complexity of the algorithm is $\mathcal{O}(\frac{n}{\varepsilon} \log n \log \log n)$ and it uses $\mathcal{O}(n \log n)$ space, where n is the number of vertices of the terrain, and $c = 2^{(p-1)/p}$.

Papadimitriou [114] proposed the first $(1 + \varepsilon)$ -approximation algorithm for the ESP3D. The time complexity of his method is $\mathcal{O}(\frac{n^4}{\varepsilon^2}(L + \log(n/\varepsilon)))$ time, where L is the number of bits of precision. Also, Asano et al. [115] gave an approximation algorithm that has logarithmic complexity in terms of L . They introduced a general technique to compute approximate solutions for optimization problems and they applied it to the ESP3D. The time complexity of their algorithm is $\mathcal{O}(n^4 \varepsilon^{-2} \log \log(2^L/OPT))$, where OPT is the value of the optimization function for the optimum solution.

Agarwal et al. [116] proposed a $(1 + \varepsilon)$ -approximation algorithm to compute a Euclidean shortest path amid a set of k convex obstacles in 3D, with a total of n faces. The running time of the algorithm is $\mathcal{O}(n + (k^4/\varepsilon^7) \log^3(k/\varepsilon))$. They are using a different approach than placing Steiner points. Their approach is based on storing a “core-set” of the input. They quickly compute a small sketch of obstacles and use that sketch in later computations. They also provide a data structure to answer queries quickly after spending preprocessing time. Since they are using the core-set approach, the size of the data structure and query time are independent of n .

To the best of our knowledge, the only $(1 + \varepsilon)$ -approximation algorithm for the WSP3D is [103]. The input is a polyhedral domain $\mathcal{D}$ in which each tetrahedron has a positive real

weight. Aleksandrov et al. proposed an approximation algorithm, based on placing Steiner points, with time complexity $\mathcal{O}(C(\mathcal{D}) \frac{n}{\varepsilon^{2.5}} \log \frac{n}{\varepsilon} \log^3 \frac{1}{\varepsilon})$, where $C(\mathcal{D})$ captures some of the geometric factors of the tetrahedra.

1.4 Fréchet Distance Literature Review

1.4.1 Hausdorff Distance

The Hausdorff distance (Equation 1.6) is one the most used measures for matching problems (see [104]). Let P and Q be two point sets in $\mathbb{R}^2$. The L_2 -based Hausdorff distance of P and Q can be computed in $\mathcal{O}((m+n) \log(m+n))$ time, where $m = |P|$ and $n = |Q|$ [11].

Huttenlocher et al. [105] showed that it is possible to find a translation vector $\vec{\ell}$ that minimizes the Hausdorff distance $H(P + \vec{\ell}, Q)$, when the underlying metric is L_p , $p = 2, 3, \dots$, in $\mathcal{O}(mn(m+n)\alpha(mn) \log(m+n))$ time, where $\alpha(n)$ is the inverse Ackermann function. For L_1 and L_∞ , the translation vector $\vec{\ell}$ can be determined in $\mathcal{O}(mn \log^2(mn))$ time [106]. In addition, a rigid motion $R(\cdot)$ that minimizes the Hausdorff distance $H(R(P), Q)$ can be computed in $\mathcal{O}((m+n)^6 \log(mn))$ time [107]. Note that a rigid motion $R(P)$ moves points in P to different locations (by translation and rotation) without altering the relative distances between points in P . Since the complexities of the Hausdorff-based matching algorithms are high, specially in higher dimensions, scientists look for approximation algorithms. We refer the reader to [104, 108, 109, 110] for details. In [110], Agarwal et al. extend the notion of the Hausdorff distance to sets of disks and balls. They proposed several exact and approximation algorithms for this new Hausdorff distance, under translation. Recently, Nutanong et al. [111] studied the Hausdorff distance in practical settings. They proposed a new incremental algorithm to compute the Hausdorff distance that outperforms the other algorithms in practical settings.

1.4.2 Fréchet Distance

The Fréchet distance was first defined by Maurice Fréchet in 1906 [21]. Then, in 1995, Alt and Godau [22] applied it to measuring the similarity of polygonal curves. They give an $\mathcal{O}(n^2 \log n)$ time algorithm to compute the standard Fréchet distance, where n is the maximum number of segments in the input polygonal curves. Very recently, Buchin et al. [24] proposed a randomized algorithm to compute the Fréchet distance between two polygonal curves in expected time $\mathcal{O}(n^2(\log \log n)^2)$ on a RAM.

In [25], Driemel and Har-Peled discuss the Fréchet distance with shortcuts. A shortcut on a polygonal curve T , replaces a sub-curve between two vertices of T , by a line segment. In this problem it is allowed to have k shortcuts between vertices of one of the two curves, where k is a constant specified as an input parameter. A k -shortcut of a polygonal curve T is an order-preserving concatenation of $k + 1$ non-overlapping (possibly empty) sub-curves of T with k shortcuts connecting the endpoints of the sub-curves. For two given polygonal curves, T_1 and T_2 , and a constant k , the goal is to find the minimum Fréchet distance among all possible k -shortcuts of T_1 and T_2 . Note that in this problem the shortcuts replace the removed sub-curves and are considered for matching during the computation of Fréchet distance. Driemel and Har-Peled [25] provided a constant factor approximation algorithm for this problem. Recently, Buchin et al. [26] studied a variant of the problem that allows shortcuts between arbitrarily chosen points on the polygonal curves (as apposed to vertices). They showed that this problem is NP-hard.

Note that the Fréchet distance is based on the maximum leash length during the walks. This property makes the Fréchet distance sensitive to outliers. It is natural to extend the Fréchet distance that instead of the maximum, takes the average, or sum, of all leash lengths, required at any time during the walks. Some definitions for average Fréchet distance are proposed in [27] and [18] which take the average over certain samples instead of taking the maximum. Also, Efrat et al. [28] proposed a variant of the integral version of the Fréchet

distance.

In [29], the problem of minimizing the Fréchet distance under translations is studied. Alt et al. proposed an algorithm for the optimization problem with $\mathcal{O}(n^8 \log n)$ time complexity. The high time complexities (mostly more than quadratic) of algorithms for computing Fréchet distance and matching under Fréchet distance, has led to the study of approximation algorithms. The Fréchet distance has also been studied for some specific families of polygonal curves. A curve is k -bounded if for any two points, x and y , on the curve, the sub-curve between x and y is not further away from either x or y than $k/2$ times the distance between x and y . In [30], Alt et al. showed that the Fréchet distance of two k -bounded curves is bounded by $k + 1$ times their Hausdorff distance. They proposed an algorithm with time complexity $\mathcal{O}((m+n)\text{polylog}(m+n))$ to compute a $(k+1)$ -approximation of the Fréchet distance. Driemel et al. [31] studied the Fréchet distance of another family of curves, c -packed curves. A curve is c -packed if the total length of the curve inside any circle is bounded by c times the radius of that circle. They showed that the Fréchet distance between two c -packed curves could be arbitrarily larger than their Hausdorff distance. They proposed a $(1 + \varepsilon)$ -approximation of the Fréchet distance between two c -packed polygonal curves with time complexity $\mathcal{O}(cn/\varepsilon + cn \log n)$.

In [32], Efrat et al. proposed two new metrics for measuring the distance between non-intersecting polygonal curves: geodesic width and link width. For two non-intersecting polygonal curves, T_1 and T_2 , the geodesic width, $GW(T_1, T_2)$, is defined as follows:

$$GW(T_1, T_2) := \min_{\alpha_1, \alpha_2} \max_{t \in [0,1]} \{d_E(T_1(\alpha_1(t)), T_2(\alpha_2(t)))\}, \quad (1.12)$$

where $d_E(a, b)$ denotes the length of a shortest path between a and b that does not cross T_1 and T_2 , and lies between the two shortest paths connecting the endpoints of T_1 and T_2 . The minimization is over continuous monotone parameterizations of T_1 and T_2 . Also the link

width, $LW(T_1, T_2)$, is defined as follows:

$$LW(T_1, T_2) := \min_{\alpha_1, \alpha_2} \max_{t \in [0, 1]} \{d_L(T_1(\alpha_1(t)), T_2(\alpha_2(t)))\}, \quad (1.13)$$

where $d_L(a, b)$ denotes the minimum number of edges in a piecewise-linear path between a and b that does not cross T_1 and T_2 . In this definition, they assumed that α_1 and α_2 are not necessarily monotone. Efrat et al. proposed algorithms to compute the geodesic width in $\mathcal{O}(n^2 \log^2 n)$ time using $\mathcal{O}(n^2)$ space and the link width in $\mathcal{O}(n^3 \log n)$ time using $\mathcal{O}(n^2)$ space, where n is the total number of edges of the input polygonal curves. The time and space complexities for computing the geodesic width were improved by Bespamyatnikh [33], to $\mathcal{O}(n^2)$ time and $\mathcal{O}(n)$ space, respectively.

In [34], Cook and Wenk have proposed an algorithm to compute the geodesic Fréchet distance between two polygonal curves, T_1 and T_2 , inside a simple polygon P . The time complexity of their algorithm for the decision problem is $\mathcal{O}(k + N^2 \log k)$, where N is the maximum of the size of T_1 and T_2 and k is the size of P . They have also proposed a randomized algorithm to solve the geodesic Fréchet optimization problem in $\mathcal{O}(k + N^2 \log k N \log k)$ expected time.

Recently, Maheshwari et al. [35, 21] studied a new generalization of the standard Fréchet distance. They considered a problem instance in which the speed of traversal along each segment of the input polygonal curves is restricted to be within a specified range. They proposed an algorithm with time complexity $\mathcal{O}(n^3 \log n)$ to find the exact Fréchet distance with speed limits.

In [36], Cheung and Daescu studied the Fréchet distance problem in weighted regions. In this problem, the distance between two matched points is the weighted Euclidean length of the shortest path between the points (see Definition 11). They proposed a $(1 + \varepsilon)$ -approximation algorithm for computing the Fréchet distance between two polygonal curves.

The time complexity of their algorithm is $\mathcal{O}(n^4 N^4 \log(nN))$, where $N = \mathcal{O}(C(\mathcal{P})(\frac{n}{\varepsilon}(\log \frac{1}{\varepsilon} + \log n) \log \frac{1}{\varepsilon}))$ and $C(\mathcal{P})$ captures some of the geometric parameters and the weights of the weighted regions.

In some applications, the weak Fréchet distance is preferable to the standard Fréchet distance. For example, in [18], Brakatsoulas et al. utilize the standard and weak Fréchet distances to design map-matching algorithms. They obtained comparable matching results for trajectories, however the theoretical time complexity of computing the weak Fréchet distance is lower. In [22], Alt and Godau proposed algorithms to compute the weak Fréchet distance in $\mathcal{O}(n^2 \log n)$ time, where n is the maximum number of segments in the input polygonal curves. The time complexity was improved by Har-Peled and Raichel [37]. They proposed an algorithm with quadratic time complexity for computing a generalization of the weak Fréchet distance. In robotics, the weak Fréchet distance is related to a measure known as ring-width, e.g., studied in [22, 38]. In the ring-width problem, the input is a closed polygon, P , and two half-lines, h_1 and h_2 . The starting point of h_1 and h_2 is on the boundary of P and they do not intersect each other or P at any other point. The objective is to find the minimum width ring such that it is possible to move P through the ring, starting from h_1 and ending at h_2 . This problem was solved for the first time by Goodman et al. [38] and an alternative easier solution was provided by Alt and Godau [22], based on the weak Fréchet distance.

1.4.3 Coupling Distance

The coupling distance (1.10) can be computed in $\mathcal{O}(n^2)$ time by a dynamic programming algorithm, proposed in [39]. Later, Agarwal et al. [40] proposed an algorithm with sub-quadratic time complexity to compute the coupling distance. Their algorithm has a time complexity of $\mathcal{O}(\frac{n^2 \log \log n}{\log n})$ and linear space complexity. Aronov et al. [41] discussed the computation of the coupling distance for some specific classes of polygonal curves, i.e., k -bounded

and *backbone* curves. Backbone curves are widely used to model molecular structures. They are polygonal curves that have the following properties: (1) For any two non-consecutive vertices, u and v , of the curve, $1 \leq L_2(u, v)$. (2) Every edge of the curve has length between two constants, c_1 and c_2 , where $c_2 > c_1 > 0$. Aronov et al. proposed some near linear time $(1 + \varepsilon)$ -approximation algorithms to compute the coupling distance for these types of curves. For example, for two backbone curves, B_1 and B_2 , in a plane, they proposed an algorithm to compute a $(1 + \varepsilon)$ -approximation of the coupling distance of B_1 and B_2 in time $\mathcal{O}((n + m)\varepsilon^{-2} \log(nm))$, where n (resp. m) is the number of vertices of B_1 (resp. B_2). If B_1 and B_2 are backbone curves in 3D, the time complexity of their algorithm is $\mathcal{O}((nm^{1/3})\varepsilon^{-2} \log(nm))$.

Very recently, Avraham et al. [42] studied the coupling distance with shortcuts. When the shortcuts are allowed only on one of the input polygonal curves, they give a randomized algorithm with expected time complexity $\mathcal{O}(n^{6/5+\varepsilon})$, for any $\varepsilon > 0$. When shortcuts are allowed in both input polygonal curves, they give a deterministic algorithm with time complexity $\mathcal{O}(n^{4/3} \log^3 n)$.

1.4.4 Lower Bound

Buchin et al. [43] proved a lower bound of $\Omega(n \log n)$ for the Fréchet distance decision problem. The lower bound also extends to the weak Fréchet and the coupling distance [43]. It is conjectured by Alt that the Fréchet distance is a 3SUM-hard problem [21, 44]. It means, if the conjecture is proved, then a strongly subquadratic algorithm for the Fréchet distance solves the 3SUM problem in a strongly subquadratic time. In the 3SUM problem, a set of n integers is given. The question is if there are three elements of the set that sum up to zero. Recently a subquadratic algorithm for the 3SUM problem is proposed in [45] whose time complexity is $\mathcal{O}(n^2/(\log n/\log \log n)^{2/3})$, however it remains open if one can solve the 3SUM problem in a strongly subquadratic time complexity (i.e., $\mathcal{O}(n^{2-\varepsilon})$, for any constant

$\varepsilon > 0$) [44].

Very Recently, Bringmann [46] obtained a conditional lower bound for the Fréchet distance problem. He assumed the *Strong Exponential Time Hypothesis (SETH)* or, more precisely, that there is no $\mathcal{O}^*((2 - \delta)^n)$ algorithm² for CNF-SAT problem, for any $\delta > 0$. He proved that there is no algorithm with time complexity of $\mathcal{O}(n^{2-\varepsilon})$ for the Fréchet distance problem unless SETH fails, where $\varepsilon > 0$ is a constant. His result also holds for the coupling distance problem. Building on that work, Bringmann and Mulzer [47] proposed a new conditional lower bound that a strongly subquadratic algorithm for the coupling distance is unlikely to exist, even in the one-dimensional problem.

²The notation $\mathcal{O}^*(.)$ hides polynomial factors in the number of variables and the number of clauses.

Chapter 2

Weighted Region Problem in Arrangement of Lines

In Section 1.3.4 we discussed the weighted region problem and related work. This problem ranks among the well-studied problems in Computational Geometry and related fields. In this problem, the input is a set of regions (often a triangulation), where each region (triangle) has a corresponding weight, and two points, source s and target t . The output is the weighted shortest path from s to t , π_{st} , which is the path with minimum cost. The cost of the path is the total sum of the Euclidean length of each segment multiplied by the corresponding region's weight (Definition 11, in Section 1.1.1, where the underlying metric is L_2).

To the best of our knowledge, nobody has studied the weighted region problem when the input partitioning is induced by an arrangement of lines. It is impossible to cover the whole length of the lines with Steiner points, because lines are infinite and we cannot afford an infinite number of Steiner points. Therefore, in this context, the first challenge is to bound the number of Steiner points which would be provided by a bound on the region that weighted shortest paths, from s to t , lie in. After establishing this bound (i.e., a closed region) the infinite lines can be clipped to bounded length segments, and the faces of the arrangement inside that region can be triangulated. Thus, by using the algorithm described in [96] a $(1 + \varepsilon)$ -approximation can be obtained.

Problem Definition. The formal problem statement is as follows: let s and t be two points in the plane $\mathbb{R}^2$ and let $\mathcal{A}$ be an arrangement of $n \geq 3$ lines l_i , $i = 1 \dots n$, in $\mathbb{R}^2$. For simplicity, assume no two lines in $\mathcal{A}$ are parallel to each other and no three lines have a common intersection. Each face of $\mathcal{A}$ is assigned positive weight w_i . By convention, the

weight of each edge of $\mathcal{A}$ is the minimum of the weights of its adjacent faces. The task is to find a closed region, $\mathcal{R}$, in $\mathbb{R}^2$ that contains a weighted shortest path from s to t , π_{st} . In particular, we want $\mathcal{R}$ to be minimal in the following sense: given the arrangement and $\mathcal{R}$, then if $\mathcal{R}'$ is a proper subset of $\mathcal{R}$, there exists a weight assignment to the faces of the arrangement and a pair of points, (s, t) , such that no weighted shortest path from s to t exists in $\mathcal{R}'$ although one exists in $\mathcal{R}$.

A naive solution, that is not optimal, is a disk, centered at s whose radius is $|\overline{st}|$ multiplied by $w_{max} = \max_i w_i$, where $|\overline{st}|$ is the Euclidean distance between s and t . It is straightforward to see that π_{st} will be inside this circle, when all $w_i \geq 1$. However, this circle may not be a solution when there are faces with weight $0 < w_i \leq 1$. In this case, a bigger circle, centered at s , with radius of $|\overline{st}|$ multiplied by w_{max}/w_{min} , contains π_{st} , where $w_{min} = \min_i w_i$. This circle clips the lines to segments and the lengths of segments are bounded by the diameter of the circle. However, the radius of this circle is very sensitive to outliers, when w_{max} is very large or w_{min} is very small.

In this chapter, we propose an algorithm to construct a closed polygonal region, called SP-Hull (*Shortest Path Hull*), that is independent of the weights assignment. For an arrangement of lines, $\mathcal{A}$, we define the convex hull of $\mathcal{A}$ to be the convex hull of the intersection points of the lines in $\mathcal{A}$. The proposed algorithm in this chapter exploits the fact that in an arrangement of lines, the lines diverge outside of the convex hull of $\mathcal{A}$. Therefore, any shortest path that starts and ends inside of the convex hull of $\mathcal{A}$, cannot go arbitrarily far from the convex hull (i.e., there is a bound). We show that there are some polygonal chains that define this bound for shortest paths, and they intersect in a restricted way (which are characterized later in this chapter). From this, we construct the SP-Hull. We will prove that any π_{st} lies inside the SP-Hull. We also justify that this is an optimally bounded region, one in which π_{st} is located in the absence of any assumption on the weights.

The structure of this chapter is as follows. In Section 2.1, necessary preliminaries are

presented. In Section 2.2, some relevant geometric properties are discussed. The algorithm to construct SP-Hull is described and analyzed in Section 2.3. At the end, we conclude the chapter.

2.1 Preliminaries

Let $\mathcal{A}$ be an arrangement of $n \geq 3$ lines l_i , $i = 1 \dots n$, and P be the set of intersection points of l_i , $P = \{p_1, p_2, \dots, p_{n(n-1)/2}\}$. The convex hull of P is denoted by $\mathcal{CH}(P) = \langle c_1, \dots, c_H \rangle$. Each line l_i either intersects $\mathcal{CH}(P)$ twice, at $\mathbf{a}_{i_1}$ and $\mathbf{a}_{i_2}$, or contributes a segment to the boundary of the convex hull, $\partial\mathcal{CH}(P)$, from $\mathbf{a}_{i_1} \in l_i$ to $\mathbf{a}_{i_2} \in l_i$. For each l_i , $i = 1 \dots n$, we define two non-intersecting rays (subset of l_i) from $\mathbf{a}_{i_1}$ and $\mathbf{a}_{i_2}$, respectively toward infinity. Sort all the rays based on their slopes, and arrange them in a counter-clockwise order around $\mathcal{CH}(P)$. This defines an order “ $<$ ” for the rays $R = \langle r_1, r_2, \dots, r_{2n} \rangle$ (Figure 2.1) which is well-defined. Note that all the rays diverge and there is no intersection between any two of them in the exterior of $\mathcal{CH}(P)$.

Since there are at least 3 lines in $\mathcal{A}$, $\mathcal{CH}(P)$ is not empty. For simplicity, it is assumed that s and t are inside (or on the boundary of) $\mathcal{CH}(P)$. If they are not, two lines can be added to the arrangement as follows: let a_1 be the minimum slope of l_i , $i = 1 \dots n$, and a_2 be the second smallest slope of l_i , $i = 1 \dots n$. The first line, ℓ' , passes through s and has slope of $a' = (a_1 + a_2)/3$. The second line, ℓ'' , passes through t and has slope of $a'' = 2(a_1 + a_2)/3$. The line ℓ' (ℓ'') intersects all l_i , $i = 1 \dots n$, and ℓ'' (ℓ'). This ensures that s and t are not outside of $\mathcal{CH}(P)$. However, π_{st} does not necessarily lie inside $\mathcal{CH}(P)$. For example, in Figure 2.1, suppose the weight of the face f_i is “very large” and the weight of the face f_{i+1} is “very small”. Then, the shortest path from s to t goes outside of $\mathcal{CH}(P)$, as depicted in the figure.

In this chapter, each ray is identified by a pair $r = \langle \mathbf{a}, \vec{d} \rangle$, where $\mathbf{a}$ is the starting point on the boundary of $\mathcal{CH}(P)$ and $\vec{d}$ is a vector pointing away from $\mathcal{CH}(P)$. W.l.o.g., it can be assumed for the remainder of the chapter that the angle between any two consecutive rays,

$r_1 = \langle \mathbf{a}_1, \vec{d}_1 \rangle, r_2 = \langle \mathbf{a}_2, \vec{d}_2 \rangle \in R$, is less than $\frac{\pi}{2}$. If it is not, (since this angle is less than π) one extra ray $r' = \langle \mathbf{a}', \vec{d}_1 + \vec{d}_2 \rangle$ can be added in between, where $\mathbf{a}'$ is a point on the boundary of $\mathcal{CH}(P)$, between $\mathbf{a}_1$ and $\mathbf{a}_2$. The total number of such angles greater than or equal to $\frac{\pi}{2}$ in R is at most 4. Therefore, by adding a constant number of rays to R this assumption holds.

Definition 12 (Order of the points on a ray). *For two points x and y on a ray $r_i = \langle \mathbf{a}, \vec{d} \rangle$, $x < y$ if $|\vec{ax}| < |\vec{ay}|$, where $|\cdot|$ denotes the Euclidean length of a vector.*

Note that this is defined for points on a ray $r_i \subset l_j$. The point $\mathbf{a}$ is mapped to zero, and the points on the ray r_i are mapped to $\mathbb{R}^+$, in the direction of $\vec{d}$.

Definition 13 (Chains: chain_i^{ccw} and chain_i^{cw}). *Let c_i be a vertex of $\mathcal{CH}(P)$ corresponding to the intersection of rays r_{i-1} and r_i . The chain_i^{ccw} is a polygonal chain, starting from c_i , defined as follows. Let $N(c_i, r_{i+1})$ be the normal from c_i to r_{i+1} . Let $N(c_i, r_{i+1})$ and r_{i+1} intersect at the point h_{i+1} . Find the normal from h_{i+1} to r_{i+2} and repeat until, either the normal is incident on a vertex of $\mathcal{CH}(P)$ or is incident on a point in the interior of $\mathcal{CH}(P)$. Then, $\text{chain}_i^{ccw} = \langle c_i, h_{i+1}, \dots, h_j \rangle$, where $h_j \in \mathcal{CH}(P)$ (see Figure 2.1). The chain_i^{cw} is defined analogously.*

The inner angle between two consecutive segments of chain_i^{ccw} is the angle on the left-hand side, when the direction is from c_i towards h_j . Analogously, for chain_i^{cw} , it is the angle on the right-hand side.

2.2 Geometric Properties

In this section, some of the geometric properties on the order of the rays in the set R are discussed. Based on these properties, some lemmas about the chains, which are the primitive elements for constructing the SP-Hull, are proved.

Property 1. *Let $r_h < r_i < r_j$ be three rays in R and the angles between r_h and r_i , and, r_i and r_j , are both less than $\frac{\pi}{2}$. Let x be a point on r_h and y be a point on r_j .*

Figure 2.1: For each line in the arrangement there are two rays (in blue). Also, each vertex of $\mathcal{CH}(P)$, denoted by c_i , has two chains, chain_i^{ccw} and chain_i^{cw} (the red dashed lines in the figure). One of the inner angles of chain_i^{ccw} is shown in the figure (incident at r_{i+3}). Furthermore, suppose the weight of f_i is “very large” and the weight of f_{i+1} is “very small”. Then, π_{st} goes outside of $\mathcal{CH}(P)$.

- (a) Let the normal from x to r_i be directed from x toward r_i . The normal from x to r_j , lies on the left side of the normal from x to r_i . Analogously, the normal from y to r_h , lies on the right side of the normal from y to r_i , directed from y toward r_i (Figure 2.2 a).
- (b) The normals from x and y to r_i lie on the opposite sides of the straight line that connects x to y , $\overline{xy}$ (or both coincide with $\overline{xy}$) (Figure 2.2 b).

Proof. The property follows from the fact that rays in R diverge and do not intersect in the exterior of $\mathcal{CH}(P)$. □

The corollary of Lemma 1 and Property 2 is that there exists at least one chain_i^{ccw} or chain_i^{cw} , outside of $\mathcal{CH}(P)$, that intersects a ray in R .

Lemma 1. Let $c_i \in r_h$ and $c_{i+1} \in r_j$ be two consecutive vertices of $\mathcal{CH}(P)$. (i) If $r_h < r_i < r_j$, then one of the normals from c_i or c_{i+1} to r_i lies outside of $\mathcal{CH}(P)$ (or on its boundary). (ii) One of the normals, either from c_i to r_{h+1} or from c_{i+1} to r_{j-1} , lies outside of $\mathcal{CH}(P)$ (or on it) (see Figure 2.2 d).

Figure 2.2: a) Property 1a, the normal from x to r_j lies on the left side of the normal from x to r_i . b) Property 1b, the normals from x and y to r_i lie on the opposite sides of $\overline{xy}$. c) Property 2b, if $\overline{xh_1}$ intersects with $\overline{yh_2}$, then $h_2 < x$ and $h_1 < y$. d) Lemma 1, one of the normals, either from c_i to r_{i+1} or from c_{i+1} to r_{i+k} , lies outside of $\mathcal{CH}(P)$.

Proof. (i) There is an edge e of $\mathcal{CH}(P)$ which is connecting c_i and c_{i+1} . By Property 1b, normals from c_i and c_{i+1} lie on the different sides of e or they coincide. Thus, either one of the normals lies outside or both are on the boundary of $\mathcal{CH}(P)$. (ii) If the normal from c_i to r_{h+1} lies outside the lemma is proved. Otherwise, by the first part of this lemma, the normal from c_{i+1} to r_{h+1} lies outside (or on) the $\mathcal{CH}(P)$. Therefore, by Property 1a, the normal from c_{i+1} to r_{j-1} lies outside (or on it). $\square$

Property 2a, shows that the order of points on a ray, in R , is preserved when they are projected orthogonally to an adjacent ray. We use Property 2b later, in this chapter, to prove Lemma 7 and Corollary 1 which say that no two counterclockwise nor two clockwise chains intersect. Therefore, an intersection between two chains happens only between a counterclockwise chain and a clockwise chain.

Property 2. Let $r_i < r_j$ be two rays in R so that the angle between them is less than $\frac{\pi}{2}$.

(a) Let $x < y$ be two points on r_i . If the normal from x to r_j is at h_1 and the normal from y to r_j is at h_2 , then $h_1 < h_2$.

(b) Let x and y be two points on r_i and r_j , respectively. If the normal from x (y) to r_j (r_i) is at h_1 (h_2), and $\overline{xh_1}$ intersects with $\overline{yh_2}$, then $h_2 < x$ and $h_1 < y$ (Figure 2.2 c).

Proof. The proof of (a) follows directly from the fact that rays in R diverge. To prove (b) assume that the axes are rotated until r_i is horizontal. Therefore, $\overline{yh_2}$ is vertical. Since r_i and r_j diverge, if x is chosen s.t. $x < h_2$ then $h_1 < y$. It implies that there will be no intersection. Therefore, to obtain an intersection between $\overline{xh_1}$ and $\overline{yh_2}$, x should be chosen s.t. $h_2 < x$. By this selection for x the only possible choice to pick y is $h_1 < y$. $\square$

In Lemma 2, we characterize the common tangent of two intersected chains. This lemma and Lemma 3 are the base for proving Lemma 4. Lemma 4 states how to merge two intersected chains into a bigger chain between two vertices of $\mathcal{CH}(P)$.

Lemma 2. (i) All inner angles of a chain, are less than π . (ii) Furthermore, let $chain_i^{ccw} = \langle c_i, h_{i+1}, \dots, h_{s-1}, h_s, h_{s+1}, \dots \rangle$ and $chain_j^{cw} = \langle c_j, h'_{j-1}, \dots, h'_{s+1}, h'_s, h'_{s-1}, \dots \rangle$ intersect between r_s and r_{s+1} (see Figure 2.3a). Then, the common tangent l_t of $chain_i^{ccw}$ and $chain_j^{cw}$ passes through $h_s \in chain_i^{ccw}$ and $h'_{s+1} \in chain_j^{cw}$.

Proof. (i) It follows directly from the fact that the rays diverge and chains are defined by the normals to the rays. (ii) We provide a proof by contradiction for one the cases, when l_t passes through h_{s-1} and h'_{s+1} . Other cases for other pairs of vertices are analogous. Since $chain_i^{ccw}$ and $chain_j^{cw}$ are intersecting, both lie on the same side of l_t . Therefore, the normal from h_{s-1} to r_s and from h'_{s+1} to r_s both lie on the same side of l_t . This contradicts Property 1b. $\square$

Figure 2.3: a) Two chains, $\text{chain}_i^{\text{ccw}}$ (the red dashed chain) and $\text{chain}_j^{\text{cw}}$ (the blue dashed chain), and their common tangent, l_t . b) An example of the topological structure of the SP-Hull is shown in black solid lines. The red dashed line is the assumed weighted shortest path between s and t .

Definition 14 (Complete revolution). Suppose $R = \langle r_1, \dots, r_{2n} \rangle$ is the counter-clockwise order of the rays and $\text{chain}_i^{\text{ccw}}$ is initiated at $c_i \in \mathcal{CH}(P)$ where $c_i \in r_j$. A $\text{chain}_i^{\text{ccw}}$, initiated at a point $x \in r_j$, is said to achieve a complete revolution, if it successively traverse all the rays in order and returns back to r_j at a point x' such that x' is equal to x or $x < x'$. The definition of a complete revolution for a $\text{chain}_i^{\text{cw}}$ is analogous.

Lemma 3. No chain starting at a vertex $c_i \in \mathcal{CH}(P)$ achieves a complete revolution.

The proof of this lemma is based on the following observation. There always exists a circle $c_{\max}$, passing through c_i with the center inside $\mathcal{CH}(P)$ that encloses $\mathcal{CH}(P)$. We can prove that the chain initiated at this vertex lies inside $c_{\max}$. This implies that this chain does not achieve a complete revolution.

Lemma 4. Let $\text{chain}_i^{\text{ccw}} = \langle c_i, h_{i+1}, \dots, h_{s-1}, h_s, h_{s+1}, \dots \rangle$ and $\text{chain}_j^{\text{cw}} = \langle c_j, h'_{j-1}, \dots, h'_{s+1}, h'_s, h'_{s-1}, \dots \rangle$ intersect between r_s and r_{s+1} (Figure 2.3a). Then, $\text{chain}_{ij} = \langle c_i, h_{i+1}, \dots, h_{s-1}, h_s, h'_{s+1}, h'_{s+2}, \dots, h'_{j-1}, c_j \rangle$ is a polygonal chain, connecting c_i to c_j and the inner angles of chain_{ij} are less than π .

Proof. By Lemma 2, chain_{ij} from c_i to h_s and from h'_{s+1} to c_j is convex. Therefore, it suffices to show that $\angle h_{s-1}h_sh'_{s+1}$ and $\angle h_sh'_{s+1}h'_{s+2}$ are less than π . In Lemma 2 we showed that the

common tangent of chain_i^{ccw} and chain_j^{cw} , l_t , passes through h_s and h'_{s+1} . Since l_t is a straight line and both chains lie on the same side of l_t , $\angle h_{s-1}h_sh'_{s+1}$ and $\angle h_sh'_{s+1}h'_{s+2}$ are less than π . $\square$

In the remainder of this section, before continuing to the construction algorithm, we formally define two sets, CCW_{max} and CW_{max} , of chains that contain a sufficient number of chains to construct SP-Hull. We will prove, in Lemma 8, that a chain in CCW_{max} intersects exactly one chain in CW_{max} , or it is between two vertices of $\mathcal{CH}(P)$. That is critical, when we want to prove that the result of our construction algorithm is a closed polygonal region. Let $\text{CW} = \{\text{chain}_i^{cw} \mid i = 1..H\}$ and $\text{CCW} = \{\text{chain}_i^{ccw} \mid i = 1..H\}$.

Lemma 5. *Every $r_i \in R$ intersects at least one $\text{chain}_j^{ccw} \in \text{CCW}$ or one $\text{chain}_{j+1}^{cw} \in \text{CW}$.*

Proof. Every $r_i \in R$ is between two consecutive vertices of $\mathcal{CH}(P)$, c_j and c_{j+1} . By Lemma 1, one of the normals from c_j and c_{j+1} to r_i is not inside $\mathcal{CH}(P)$. W.l.o.g. assume that the normal from c_j to r_i is not inside. By Property 1a, chain_j^{ccw} lies on the left side (or on) the normal from c_j to r_i . Therefore, $\text{chain}_j^{ccw} \in \text{CCW}$ intersects r_j . $\square$

Lemma 6. *Any two chains in CW (or CCW) are either disjoint or share an end-point at a vertex of $\mathcal{CH}(P)$.*

Proof. This proof uses contradiction. Suppose two chains, chain_i^{cw} and chain_j^{cw} , intersect between two rays, r_s and r_{s+1} , not at a vertex of $\mathcal{CH}(P)$. Suppose chain_i^{cw} intersects r_s at x and r_{s+1} at h . Also, chain_j^{cw} intersects r_s at y and r_{s+1} at h' . W.l.o.g. assume $x < y$. If they intersect, it implies $h' < h$. This contradicts Property 2a. $\square$

Definition 15 (Maximal chain). *Suppose chain_i^{ccw} starts at r_j and ends at r_{j+k} , that is, chain_i^{ccw} covers rays from r_j to r_{j+k-1} . We represent chain_i^{ccw} by a range $[j, \dots, j+k-1]$. It is a subrange¹ of a circular range of integers $[1, \dots, 2n]$. We say chain_i^{ccw} is maximal if there*

¹For simplicity, we are omitting "modulo" as this is a circular range.

is no $chain_x^{ccw} \in \mathbb{CCW}$ or $chain_x^{cw} \in \mathbb{CW}$ such that its representative range fully covers the range $[j, \dots, j+k-1]$. The maximal $chain_i^{cw}$ is defined analogously.

Let $\mathbb{CCW}_{max} = \{chain_i^{ccw} \mid i = 1..H, \text{ s.t. } chain_i^{ccw} \text{ is maximal}\}$ and $\mathbb{CW}_{max} = \{chain_i^{cw} \mid i = 1..H, \text{ s.t. } chain_i^{cw} \text{ is maximal}\}$. By Lemma 6, $\mathbb{CCW}_{max}$ is a set of chains such that their representative ranges are disjoint. Analogously, the representative ranges of chains in $\mathbb{CW}_{max}$ are disjoint.

Lemma 7. *Suppose $chain_i^{ccw} \in \mathbb{CCW}_{max}$ and it covers the starting point of $chain_x^{cw} \in \mathbb{CW}_{max}$. Then, $chain_i^{ccw}$ and $chain_x^{cw}$ do not intersect.*

Proof. By definition, $chain_i^{ccw}$ starts at the boundary of $\mathcal{CH}(P)$ and ends inside. Therefore $chain_i^{ccw}$ forms a closed region with the boundary of $\mathcal{CH}(P)$. By the assumption of the lemma, $chain_x^{cw}$ starts from inside the corresponding region of $chain_i^{ccw}$. If these two chains intersect, the intersection contradicts Property 2b. $\square$

Corollary 1. *Let $chain_i^{ccw}, chain_j^{ccw} \in \mathbb{CCW}_{max}$ be two disjoint chains. There is no $chain_x^{cw} \in \mathbb{CW}_{max}$ that intersects both of them. Also, let $chain_i^{cw}, chain_j^{cw} \in \mathbb{CW}_{max}$ be two disjoint chains. There is no $chain_x^{ccw} \in \mathbb{CCW}_{max}$ that intersects both of them.*

Proof. If $chain_x^{cw}$ intersects $chain_i^{ccw}$ and $chain_j^{ccw}$ without intersecting $\mathcal{CH}(P)$, then by Lemma 7 it must intersect one of them at least twice. W.l.o.g. assume that $chain_x^{cw}$ intersects $chain_i^{ccw}$ twice, once to enter the closed region formed by $chain_i^{ccw}$ and once to leave it. The second intersection contradicts Property 2b. The proof for the second part is analogous. $\square$

Lemma 8. *Each $chain_i^{ccw} \in \mathbb{CCW}_{max}$ intersects exactly one $chain_j^{cw} \in \mathbb{CW}_{max}$, or it ends at a vertex $c_x \in \mathcal{CH}(P)$.*

Proof. By definition, $chain_i^{ccw} \in \mathbb{CCW}_{max}$ starts at a vertex of $\mathcal{CH}(P)$. We prove that if it does not end at another vertex of $\mathcal{CH}(P)$, then it intersects exactly one $chain_j^{cw} \in \mathbb{CW}_{max}$.

Suppose r_s is the ray that chain_i^{ccw} ends on. Thus, the intersection of chain_i^{ccw} and r_s is in the interior of $\mathcal{CH}(P)$. By Lemma 5, there exists another chain, chain_x , that intersects r_s outside (or on) $\mathcal{CH}(P)$. This chain_x cannot be a member of $\mathbb{CW}_{max}$ because it either intersects with chain_i^{ccw} (which contradicts Lemma 6) or fully covers chain_i^{ccw} (which contradicts maximality). Therefore, it is a member of $\mathbb{CW}$ and intersects chain_i^{ccw} . If it is maximal, we have proved that there exist at least one chain in $\mathbb{CW}_{max}$ that intersects. If it is not maximal, then there exists a maximal chain, chain_y , that fully covers chain_x . By the same reasoning, chain_y cannot be a member of $\mathbb{CW}_{max}$. Therefore, chain_y is a member of $\mathbb{CW}_{max}$ and intersects chain_i^{ccw} (if it does not intersect, it should fully cover chain_i^{ccw} which contradicts maximality of chain_i^{ccw}).

Now suppose there are two chains chain_x^{cw} and $\text{chain}_y^{cw} \in \mathbb{CW}_{max}$, that intersect chain_i^{ccw} . By Corollary 1, chain_x^{cw} and chain_y^{cw} should either intersect each other (which contradicts Lemma 6) or one should fully cover the other one (which contradicts maximality). Therefore, there exists exactly one $\text{chain}_j^{cw} \in \mathbb{CW}_{max}$ that intersects chain_i^{ccw} . $\square$

2.3 The Construction Algorithm

In this section, we present an algorithm to construct the SP-Hull (Algorithm 1). The input is an arrangement of lines $\mathcal{A}$, a source s , and a target t . The assumption is that s and t are inside $\mathcal{CH}(P)$. If they are not, we can add a constant number of lines to the input arrangement (at most 3) to bring them inside the $\mathcal{CH}(P)$. The output is a simple closed polygonal region SP-Hull that encloses $\mathcal{CH}(P)$. The idea to construct SP-Hull is to cover all vertices of $\mathcal{CH}(P)$ by some polygonal chains, chain_{ij} , which lie outside of $\mathcal{CH}(P)$ (see Figure 2.3b). We will prove that any weighted shortest path from s to t lies inside the SP-Hull. Furthermore, we will argue its minimality.

Theorem 1. *Let $\mathcal{A}$ be an arrangement of lines and s and t be two points inside $\mathcal{CH}(P)$. For any assignment of positive weights to the faces of $\mathcal{A}$, any weighted shortest path between*

Algorithm 1 SP-Hull

Input: Source ($s \in \mathcal{CH}(P)$), target ($t \in \mathcal{CH}(P)$), an arrangement of n lines ($\mathcal{A}$)

Output: A simple closed polygon, SP-Hull

- 1: P = the set of the intersection points of lines in $\mathcal{A}$;
 - 2: Compute $\mathcal{CH}(P) = \langle c_1, \dots, c_H \rangle$;
 - 3: Mark all $c_i \in \mathcal{CH}(P)$ as not covered;
 - 4: Find CCW_{max} and CW_{max} sets and sort the chains in these sets based on the index of their starting points;
 - 5: **while** all c_i 's are not covered **do**
 - 6: $\text{chain}_i^{ccw} =$ First element of CCW_{max} ;
 - 7: Find $\text{chain}_k^{cw} \in \text{CW}_{max}$ that intersects chain_i^{ccw} not at a vertex of $\mathcal{CH}(P)$;
 - 8: **if** chain_k^{cw} is not empty **then**
 - 9: $\text{chain}_{ik} = \text{Merge}(\text{chain}_i^{ccw}, \text{chain}_k^{cw})$;
 - 10: Mark all c_j ($j = i..k$) as covered;
 - 11: **else**/* chain_i^{ccw} ends at $c_x \in \mathcal{CH}(P)$ */
 - 12: $\text{chain}_{ix} = \text{chain}_i^{ccw}$;
 - 13: Mark all c_j ($j = i..x$) as covered;
 - 14: **return** the list of chain_{ij} , sorted by their first index (i.e., i), as the SP-Hull;
-

s and t lies inside SP-Hull of $\mathcal{A}$, constructed by Algorithm 1.

Proof. This proof has two main steps. First, we prove that SP-Hull, generated by Algorithm 1, is a simple polygon that encloses $\mathcal{CH}(P)$. In the second step we prove, by contradiction, that any weighted shortest path between s and t , π_{st} , does not go outside of SP-Hull, where $s, t \in \mathcal{CH}(P)$.

Based on the construction in Algorithm 1, SP-Hull is a sequence of chains, chain_{ij} , which do not overlap and cover all of the rays (Lemma 5). Figure 2.3b shows an example of the topological structure of the SP-Hull around $\mathcal{CH}(P)$. By Lemma 4, each chain_{ij} is a simple chain in which its inner angles are less than π . It starts and ends at the vertices of $\mathcal{CH}(P)$. Therefore, the SP-Hull is a closed simple polygon. Also, each chain_{ij} by definition is outside of $\mathcal{CH}(P)$. Therefore, SP-Hull encloses $\mathcal{CH}(P)$.

Before continuing the proof, let us introduce some notation. If π_x is a polygonal chain and a and b are two points on π_x , then $\pi_x[a, b]$ denotes the subpath of π_x from a to b .

In the second step of the proof, we show that no point of π_{st} lies in the exterior of

SP-Hull. We prove this by contradiction. Since s and t are inside $\mathcal{CH}(P)$, π_{st} intersects SP-Hull at least twice. Let i_1 and i_2 be the first two consecutive intersections of π_{st} with the SP-Hull (see Figure 2.3b). Our claim is $\text{SP-Hull}[i_1, i_2]$ is shorter than $\pi_{st}[i_1, i_2]$ which is a contradiction to the fact that π_{st} is a shortest path.

Suppose there are k regions between i_1 and i_2 which are separated by $k-1$ rays. W.l.o.g., let the rays in order be $\langle r_1, \dots, r_{k-1} \rangle$. The number of segments in $\text{SP-Hull}[i_1, i_2]$ is at most k . Furthermore, the number of segments in $\pi_{st}[i_1, i_2]$ is at least k , as it must traverse through k diverging regions. We will show that each segment of $\text{SP-Hull}[i_1, i_2]$, o_j , is shorter than the corresponding segment of $\pi_{st}[i_1, i_2]$ in that region, π_j . Then, the total length of $\text{SP-Hull}[i_1, i_2]$ is smaller than the total length of $\pi_{st}[i_1, i_2]$ and we will arrive at a contradiction.

From the fact that there is no intersection between $\text{SP-Hull}[i_1, i_2]$ and $\pi_{st}[i_1, i_2]$ from i_1 to i_2 , o_j and π_j do not intersect. There are two cases: the segment o_j is one of the normals in a chain that is contributing to SP-Hull, or it is a segment introduced by merging of two chains. The first case is shown in Figure 2.4a. In this case, even if π_j is perpendicular, o_j is shorter because the rays are diverging.

For case 2, assume that the endpoints of o_j are q_1 and q_2 , and the endpoints of π_j are q'_1 and q'_2 (see Figure 2.4b). Since r_j and r_{j+1} diverge, translating π_j toward $\mathcal{CH}(P)$ makes it shorter. Therefore, the shortest possible length for π_j while avoiding an intersection between o_j and π_j , is when one of the endpoints of π_j is as close as possible to one of the endpoints of o_j . Assume q'_1 is equal to q_1 . Then, o_j is shorter than π_j because of the following observation. The distance function from a point x to a ray r is a convex function (i.e., there is one line segment that connects x to a point $x_{opt} \in r$ such that it has the minimum length). $\square$

Theorem 2. *For an arrangement of n lines, the SP-Hull can be computed in $\mathcal{O}(n \log n)$ time.*

Proof. Computing the convex hull of P , that contains $n(n-1)/2$ intersection points of n lines, takes $\mathcal{O}(n \log n)$ time and its size is $\mathcal{O}(n)$ [117].

Figure 2.4: a) Proof of Theorem 1, case 1. b) Proof of Theorem 1, case 2.

The key here is that it is possible to find $\mathbb{C}CW_{max}$ (CW_{max}) in linear time without computing all chain_i^{ccw} (chain_i^{cw}), $i = 1 \dots H$. Lemma 6 implies that if $c_j \in CH$ is covered by a chain_i^{ccw} then we can skip computing chain_j^{ccw} and chain_j^{cw} , because they are not maximal. Also, members of $\mathbb{C}CW_{max}$ do not overlap. Therefore, the computation of $\mathbb{C}CW_{max}$ requires at most two traversals of the rays.

In the While-loop, $\mathbb{C}CW_{max}$ (CW_{max}) is a set of non-overlapping ranges that are sorted. Based on Lemma 8, each member of $\mathbb{C}CW_{max}$, either has exactly one intersection with a member of CW_{max} , or both endpoints of that chain are vertices of CH . Therefore, finding the intersecting chains takes constant time, by comparing only the endpoints of the first and the last chains in the sets. When an intersection is detected, then remove both chains from the sets, merge them and repeat. Since the total number of operations for merging all intersected chains is equal to the number of rays, the While-loop takes linear time. $\square$

Corollary 2. *Let $\mathcal{A}$ be an arrangement of $n \geq 3$ lines l_i , in $\mathbb{R}^2$ and s and t be two points in the plane $\mathbb{R}^2$. Each face of $\mathcal{A}$ is assigned positive weight w_i . We obtain a $(1 + \varepsilon)$ -approximation shortest path algorithm for weighted arrangements of lines, that has the time complexity $\mathcal{O}(C(\mathcal{A}) \frac{n^2}{\sqrt{\varepsilon}} \log \frac{n}{\varepsilon} \log \frac{1}{\varepsilon})$ where $C(\mathcal{A})$ captures the geometric parameters of the faces of the triangulated SP-Hull of $\mathcal{A}$.*

Proof. We can triangulate all the faces inside SP-Hull to obtain a triangulation with $\mathcal{O}(n^2)$ triangles. Then, we use the proposed algorithm in [82]. We obtain a $(1 + \varepsilon)$ -approximation shortest path algorithm for weighted arrangements of lines, that has the time complexity

$\mathcal{O}(C(\mathcal{A}) \frac{n^2}{\sqrt{\varepsilon}} \log \frac{n}{\varepsilon} \log \frac{1}{\varepsilon} + n \log n) = \mathcal{O}(C(\mathcal{A}) \frac{n^2}{\sqrt{\varepsilon}} \log \frac{n}{\varepsilon} \log \frac{1}{\varepsilon})$ where $C(\mathcal{A})$ captures the geometric parameters and the weights of the faces of the triangulated SP-Hull of $\mathcal{A}$. The dependency on the weights can be removed by modifying the algorithm of Aleksandrov et al. [82], as shown by Sun and Reif [96]. $\square$

2.4 Minimality of the SP-Hull

In Theorem 1, we have shown that π_{st} lies inside the SP-Hull, when $s, t \in \mathcal{CH}(P)$. Now we address its minimality. We show that for any arrangement of lines, $\mathcal{A}$, it is possible to assign weights to the faces of $\mathcal{A}$ and choose $s, t \in \mathcal{CH}(P)$ such that π_{st} is arbitrarily close to the boundary of SP-Hull.

The procedure is as follows. Assign the weight “infinity” to the bounded faces of $\mathcal{A}$. By this assignment, we make sure that π_{st} does not traverse these faces. Choose one of the chains in SP-Hull, say chain_{ij} . This chain is either chain_i^{ccw} , or chain_j^{cw} , or the result of merging them. Here, we prove the minimality for the merging case. The other cases are analogous.

Let chain_{ij} be the result of merging chain_i^{ccw} and chain_j^{cw} . W.l.o.g., assume that chain_i^{ccw} starts at $c_i \in \mathcal{CH}(P)$ and intersects $\mathcal{CH}(P)$ at point $x \in \partial\mathcal{CH}(P)$. Place s on c_i and t on x . Assume chain_i^{ccw} traverses k unbounded faces in order, $\langle f_1, \dots, f_k \rangle$. The weight for the other unbounded faces that are not visited by this chain, is set to infinity. To make π_{st} close enough to chain_i^{ccw} , the corresponding weights for f_i , $i = 1 \dots k$, are set in such a way that $w_1 \gg w_2 \gg \dots \gg w_k$. It suffices to set the weights of f_i , $i = 1 \dots k$, as z^i . If z goes to zero, then $w_i \gg w_{i+1}$ and π_{st} is arbitrarily close to chain_i^{ccw} . An analogous argument can be used to become as close as possible to chain_j^{cw} .

2.5 Conclusion

In this part, a geometric shortest path problem in weighted regions was discussed. An arrangement of lines $\mathcal{A}$, a source $s \in \mathcal{CH}(P)$, and a target $t \in \mathcal{CH}(P)$ are given. The objective is to find a weighted shortest path, π_{st} , from s to t . Existing approximation algorithms for weighted shortest paths work within bounded regions (typically triangulated). To apply these algorithms to unbounded regions, such as arrangements of lines, there is a need to bound the regions. Here, we presented a minimal region that contains π_{st} , called SP-Hull of $\mathcal{A}$. It is a closed polygonal region that is independent of the weights assignment. It is minimal in the sense that for any arrangement of lines $\mathcal{A}$, it is possible to assign weights to the faces of $\mathcal{A}$ and choose s and t such that π_{st} is arbitrary close to the boundary of SP-Hull of $\mathcal{A}$. We showed that SP-Hull can be constructed in $\mathcal{O}(n \log n)$ time, where n is the number of lines in the arrangement. Note that we can triangulate all the faces inside SP-Hull to obtain a triangulation with $\mathcal{O}(n^2)$ triangles. Therefore, as a direct consequence, we obtained a $(1 + \varepsilon)$ -approximation shortest path algorithm for weighted arrangements of lines.

At the beginning of this chapter, we mentioned a naive solution to find a closed region in $\mathbb{R}^2$ that contains a weighted shortest path from s to t , π_{st} . The naive solution is a disk, centered at s whose radius is $|\overline{st}|$ multiplied by $w_{max} = \max_i w_i$, where $|\overline{st}|$ is the Euclidean distance between s and t . Note that the naive solution could be inside the SP-Hull. However, the naive solution is very sensitive to outliers in weights. It means, if we have a very large maximum weight (or a very small minimum weight, close to zero) in a region, then the radius of this disk will be very large.

Chapter 3

Path Refinement in Weighted Regions

3.1 Introduction

The weighted region problem (WRP), is one of the well-studied problems in Computational Geometry and related fields. Related work is discussed in Section 1.3.4. Let $\mathcal{P}$ be a planar partitioning of $\mathbb{R}^2$ (e.g., a triangulation or a partitioning induced by an arrangement of lines) and f be a piecewise constant function that assigns a positive real weight to each region of $\mathcal{P}$. In WRP, the input consists of $\mathcal{P}$, f and two points $s, t \in P$. The output is a shortest path from s to t , which is a path with minimum length (as defined in Definition 11, Section 1.1.1) [80]. In this chapter, the underlying metric in Definition 11 is L_2 , and for simplicity, the length of a path, Π , is denoted by $\|\Pi\|$. As a shortest path is linear inside each region, we can consider only piecewise-linear paths.

Many of the practical applications of the shortest path problem in geographical information systems (GIS), robotics, seismology, among others, need more refined weight functions (not only $\{1, \infty\}$ weights). For example, in the computation of shortest paths in GIS, WRP allows a user to include the slopes of the regions or terrain types (e.g., forest, water, etc.). Additionally, in robotics, the energy consumption of the robot in each region can be modeled by a weight in WRP. Furthermore, in Seismology, it allows researchers to take the wave velocity factor of materials into account by assigning appropriate weights to regions.

There are some qualitative criteria about a generated path for WRP. These qualitative criteria are expressed via rules, specifying the geometry of the path. The most prominent rule is Snell's law (defined formally in Section 3.2) [80]. It is unknown and unlikely that

an algorithm can generate a path, in polynomial time, that obeys Snell's law when the weights of the regions are distinct. In addition, approximate solutions may not obey Snell's law. On the other hand, there are some algorithms that generate paths which obey some qualitative criteria. However, there is no guarantee (i.e., a proven bound) on the length of generated paths by these algorithms. For example, in [83], the smoothness of a path is mentioned as one of the desirable qualities of a path in a computer game. In that specific application, Nieuwenhuisen et al. want an approximate path that is C^1 -continuous and has sufficient clearance from obstacles. They believe that by realizing these criteria they achieve a natural looking motion. Also, in robotics, especially for autonomously guided vehicles (AGV), following a path with sharp turns is difficult. Therefore, the geometric constraints of steering are taken into account, to find a practical (but not necessarily a shortest) path [84, 85].

To the best of our knowledge, there is no ε -approximation algorithm for WRP that guarantees some qualitative criteria about the output. In this chapter, we propose two geometric qualitative criteria. They are weaker than Snell's law, in the sense that instead of determining the exact passage point on a shared boundary between two regions, they specify an interval. These criteria are the result of our discussions with Seismologists and Geophysicists [130]. We propose these criteria in the form of rules that a path should obey. We will prove that these criteria may also improve an approximate solution in terms of its length (or at least, not increase the length). In particular, if the given path to our proposed algorithms is an ε -approximate path, it is guaranteed that the output is also an ε -approximate path.

These rules are the *critical angle rule* (CAR) and the *crossing normal rule* (CNR), which are informally described as follows (the formal definitions are provided in Section 3.2). We refer to a boundary segment shared by two regions as an interface between those two regions. In this chapter, we consider only *linear* paths. By a linear path we mean, the path is simple

(i.e., non-self-intersecting) and its vertices, except s and t , are on the boundary of the regions of the partitioning. CAR states that, if a path passes through an interface between two regions, the incident angle at the interface should be less than or equal to a threshold, set as the critical angle of the interface. CNR states that, every three consecutive corners, x_{i-1} , x_i and x_{i+1} , of a path, should lie in one closed half-plane of the line from x_{i-1} perpendicular to the interface that x_i is on. We say a path is *refined* if it obeys both of these rules. Note that, in this chapter, to distinguish between a vertex of a partitioning and a vertex of a path, we use “corner” to refer to a vertex of a path.

Problem Definition. (Path Refinement in Weighted Regions, PRWR). Let $\mathcal{P}$ be a planar partitioning of $\mathbb{R}^2$. Each region of $\mathcal{P}$ is associated a positive real weight. Let Π be a linear path between two points $s, t \in \mathbb{R}^2$. The input to the problem consists of $\mathcal{P}$ and Π . The positioning of Π on $\mathcal{P}$ (i.e., the mapping between the corners (vertices) of Π and edges of $\mathcal{P}$) is also provided. The goal is to construct a path that is refined and has length less than or equal to the length of Π .

Our Contribution. To solve PRWR, we propose algorithms to refine a path when the partitioning is induced by (1) a triangulation, (2) a set of parallel lines, or, (3) a general arrangement of lines. Each of our proposed refinement algorithms has the following properties:

- Its output is a refined path.
- The length of the output path is at most the length of the input path.
- The time complexity of the algorithm is linear in the size of the input path (assuming that $\mathcal{P}$ is stored in memory, in the preprocessing step).

In particular, if the input is an ε -approximation for a shortest path, the output is a refined ε -approximation for a shortest path. In addition to a theoretical analysis described later, we have also implemented our refinement algorithm for triangulations due to its relevance

e.g., in GIS. To validate and test the implementation, we extracted several Triangulated Irregular Networks (TINs) from the Earth’s terrain. A TIN is a digital data structure for the representation of a surface. It is a vector-based representation of the physical land surface, made up of irregularly distributed nodes and lines with three-dimensional coordinates that are arranged in a network of non-overlapping triangles. Typically, a TIN is defined on a 2-dimensional point set $S = \{p_1, \dots, p_n\}$ as a maximally planar graph $G = \langle V, E \rangle$, where V is equal to S and edges are connecting only vertices of V . When set S is 3-dimensional, the z -coordinate is ignored for the graph construction, but is used subsequently for many operations. The z -coordinate is used to move the points up or down along the z -axis to create a triangulated surface that is 2.5-dimensional as opposed to truly 3-dimensional as any line parallel to the z -axis intersects the surface at most once. We set up experiments on the extracted TINs and analyzed the results. As we mentioned earlier, Lanthier et al. [95] reported that, in practice, six Steiner points on average per edge in their Interval Scheme suffice to obtain a close-to-optimal approximation. Their technique is appealing to practitioners (e.g., see [92, 93, 94]) due to its simplicity for implementation and good performance (time and quality of solution). We obtained the same accuracy by placing, three Steiner points on average per edge in the Interval Scheme and applying the proposed refinement algorithm as a post-processing step. The results show that, by using the proposed algorithm, on average, 51% in query time and 69% in memory usage could be saved, compared to the existing method.

Chapter Structure. This chapter is structured as follows: First, in Section 3.2, we define the geometric criteria and introduce some notation. Then, in Section 3.3, we propose efficient algorithms for the three different types of input partitioning: triangulations, parallel lines and arrangement of lines. In the end, in Section 3.4, we describe the experiments conducted to evaluate the performance of the proposed algorithms and then, analyze the results.

3.2 Preliminaries and Definitions

A planar partitioning $\mathcal{P}$ is a union of not necessarily bounded subsets of $\mathbb{R}^2$ such that the following holds: (1) The boundary of each region $R \in \mathcal{P}$ is piecewise-linear, (2) the interiors of any two regions do not intersect each other, and (3) $\bigcup_{R \in \mathcal{P}} R = \mathbb{R}^2$. Each region R has an associated weight $w \in \mathbb{R}_{>0}$. By $V(\mathcal{P})$ and $E(\mathcal{P})$ we denote the sets of vertices and edges of $\mathcal{P}$, respectively. By convention, the weight of an edge of $\mathcal{P}$ is the minimum weight of the regions that share the edge.

As mentioned, we consider only linear paths. A linear path Π is induced by a sequence $\langle s = x_0, \dots, x_L = t \rangle$ of points that lie on the edges of $\mathcal{P}$. As the weight of each region is constant, a shortest path is linear inside each region. L is the size of Π . For simplicity, we use the following notation: for $i = 0, \dots, L$, the edge of $\mathcal{P}$ that x_i lies on, is denoted by e_i . The region that contains segment $\overline{x_i x_{i+1}}$ is denoted by Δ_i . If $\overline{x_i x_{i+1}}$ is on the shared boundary of two regions, then Δ_i is the region with minimum weight. The weight of Δ_i is denoted by w_i . Thus, WRP asks for a (piecewise-linear) path Π between s and t such that its length $\|\Pi\| := \sum_{i=0}^{L-1} |\overline{x_i x_{i+1}}| \cdot w_i$ is minimized, where $|\overline{x_i x_{i+1}}|$ is the Euclidean length of $\overline{x_i x_{i+1}}$.

The sub-path of Π from vertex x_i to vertex x_j , $i, j = 0, \dots, L$, is denoted by $\Pi[x_i, x_j]$. For an edge $e \in E(\mathcal{P})$ and a point $p \in \mathbb{R}^2$, the line perpendicular to e that is passing through p is denoted by $N(p, e)$.

For a corner x_i that is not a vertex of $\mathcal{P}$, we define $\theta_i := \angle(\overline{x_{i-1} x_i}, N(x_i, e_i))$ and $\theta_o := \angle(\overline{x_i x_{i+1}}, N(x_i, e_i))$ (see Figure 3.1a). If Π is a shortest path, Snell's law implies that $\sin(\theta_i) \cdot w_{i-1} = \sin(\theta_o) \cdot w_i$. As $\theta_o \leq \frac{\pi}{2}$ holds, we have $\theta_i \leq \arcsin(w_i/w_{i-1})$. We denote the critical angle $\arcsin(w_i/w_{i-1})$ by θ_c . The critical angle induces one of the two qualitative criteria that we use to refine a given Π that is not necessarily optimal.

Based on the above notation, we define our qualitative criteria, the critical angle rule (CAR) and the crossing normal rule (CNR). To simplify the presentation, we call the sub-path $\Pi[x_{i-1}, x_{i+1}]$ a *(consecutive) triple*.

Definition 16. For each consecutive triple $\Pi[x_{i-1}, x_{i+1}]$, $i \in \{1, \dots, L-1\}$, CAR and CNR are defined as follows:

- $\Pi[x_{i-1}, x_{i+1}]$ obeys CAR if $\theta_i \leq \theta_c$ (see Figure 3.1a) or $x_i \in V(\mathcal{P})$, and
- $\Pi[x_{i-1}, x_{i+1}]$ obeys CNR, if x_i and x_{i+1} lie in the same closed half-plane of $N(x_{i-1}, e_i)$ (see Figure 3.1b) or $x_i \in V(\mathcal{P})$.

Figure 3.1: The sub-path $\Pi[x_{i-1}, x_{i+1}]$ (the solid line) disobeys a) CAR, b) CNR. The dotted path shows a replacement that obeys a) CAR, b) CNR.

Definition 17. Π is said to be refined if each consecutive triple $\Pi[x_{i-1}, x_{i+1}]$, of Π , for $i \in \{1, \dots, L-1\}$, obeys both CAR and CNR.

Corollary 3. If Π is a shortest path then it is refined.

Proof. Suppose there is a consecutive triple $\Pi[x_{i-1}, x_{i+1}]$ that does not obey CAR or CNR. Then, $\Pi[x_{i-1}, x_{i+1}]$ can be shortened, by translating x_i on e_i , because the distance function from x_{i-1} to x_{i+1} that passes through e_i is convex [80] (see Figure 3.1). This contradicts the fact that Π is a shortest path. $\square$

As we discussed in Section 3.1, existing approximation algorithms which are based on discretization may not produce refined paths. We illustrate this with an example: consider a partitioning $\mathcal{P}$ with two adjacent regions, Δ_1 and Δ_2 having weights 1 and w , respectively, where $w > 0$ is large (see Figure 3.2). Let s_1 and s_2 be two consecutive Steiner points on an edge e shared by Δ_1 and Δ_2 . We choose the source point s and the target point t as follows: suppose the point p is the intersection point of $N(s, \overline{s_1 s_2})$ and $\overline{s_1 s_2}$ and the point q is the intersection point of $N(t, \overline{s_1 s_2})$ and $\overline{s_1 s_2}$. We choose s such that p lies between s_1 and s_2 and $|\overline{ps_1}| = \frac{1}{4}|\overline{s_1 s_2}|$ and choose t such that q lies between s_1 and s_2 and $|\overline{qs_1}| = \frac{1}{2}|\overline{s_1 s_2}|$, where $|\cdot|$ is the Euclidean length. Now, consider a shortest path via Steiner points between s and t . It passes through s_1 . Clearly, it disobeys CNR.

Figure 3.2: The shortest path inside a discretization of WRP by Steiner points violates CNR.

3.3 Algorithms

In this section, we propose algorithms to refine a path when the partitioning $\mathcal{P}$ is induced by (1) a triangulation, (2) a set of parallel lines or, (3) a general arrangement of lines. As a corollary we obtain that, if the input to these refinement algorithms is generated by a $(1 + \varepsilon)$ -approximation algorithm (e.g., [101]), then the output is a refined $(1 + \varepsilon)$ -approximation of a shortest path.

3.3.1 Refinement Algorithm for Triangulations

In this section, we discuss the refinement algorithm when the input partitioning is a weighted triangulation T . This section is structured as follows. We first explain the general idea of the algorithm. Then, we present its details and pseudo-code (Algorithm 2), and establish the correctness of the algorithm.

Outline of the algorithm

We iteratively process the corners x_i , $i = 1, \dots, L - 1$, of the input path Π as follows: If $x_i \in V(T)$, we skip x_i (where $V(T)$ is the vertex set of the input triangulation T). Otherwise, we refine $\Pi[x_{i-1}, x_{i+1}]$ consecutively w.r.t. CAR and CNR. More specifically, we translate x_i on e_i such that $\Pi[x_{i-1}, x_{i+1}]$ obeys CAR and CNR, respectively (the refinement is explained later in detail). We call a refinement of triple $\Pi[x_{i-1}, x_{i+1}]$, a *(single) refinement of x_i* . Unfortunately, the refinement of x_i could undo the refinement of x_{i-1} w.r.t. CNR, which is defined geometrically as a *CNR dependency* between x_{i-1} and x_i . Hence, an additional refinement of x_{i-1} would become necessary. This could cause an additional refinement of x_{i-2} and so on. This cascading effect could lead to a quadratic running time, in the worst case.

However, to obtain a linear running time, we compute a maximal chain of corners $x_{i+1}, \dots, x_{last}$ whose refinements could influence directly or indirectly the refinement of x_i . Then, we process all the corners $x_i, \dots, x_{last}$ simultaneously, as a batch. We compute a new path, called *orthogonal chain*, from x_i to x_{last} , which is shorter than $\Pi[x_i, x_{last}]$ and is a refined path. We replace $\Pi[x_i, x_{last}]$ by the orthogonal chain from x_i to x_{last} . After that we advance i to $last$ and continue the processing of the corners from this point onwards.

Details of the algorithm

Definition 18. Let $e \in E(T)$ be incident to triangles Δ_i and Δ_j . Let $u \in \Delta_i$ be chosen arbitrarily. We define $Cone_{u,e} := \{x \in \mathbb{R}^2 \mid \angle(\overline{ux}, N(u,e)) \leq \theta_c\}$ and $Passage_{u,e} = Cone_{u,e} \cap e$

(see Figure 3.3).

Figure 3.3: The line segment e is the interface between two triangles, Δ_i and Δ_j . a) The $Passage_{u,e}$ is shown by a red solid line. b) The $Passage_{u,e}$ is empty.

Single Refinement. First, if $\Pi[x_{i-1}, x_{i+1}]$ disobeys CAR, we refine it w.r.t. CAR as follows: We translate x_i to the closest point, to x_i , on $Passage_{x_{i-1}, e_i}$. Second, if $\Pi[x_{i-1}, x_{i+1}]$ disobeys CNR, we refine it w.r.t. CNR as follows: we translate x_i to $N(x_{i-1}, e_i) \cap e_i$. If $Passage_{x_{i-1}, e_i} = \emptyset$ or $N(x_{i-1}, e_i) \cap e_i = \emptyset$, we translate x_i to the endpoint of e_i that is closer to $N(x_{i-1}, e_i)$.

Observation 1. If $\Pi[x_{i-1}, x_{i+1}]$ obeys CAR (CNR), then, after refinement for CNR (CAR), it still obeys CAR (CNR).

Observation 2. If x_i is translated to refine $\Pi[x_{i-1}, x_{i+1}]$ w.r.t. CNR, then $\Pi[x_{i-1}, x_{i+1}]$ remains refined after the translation and the refinement of $\Pi[x_i, x_{i+2}]$ will not affect it.

Refinement of a chain of corners. We present an approach to maintain refinements made for $\Pi[x_0, x_{i-1}]$ while refining $\Pi[x_{i-1}, x_{i+1}]$. The geometrical configuration that leads to a dependency between refinement of x_{i-1} and x_i is characterized in the following definition.

Definition 19. There is a CNR dependency between x_{i-1} and x_i , denoted by $x_{i-1} \triangleright x_i$, if $N(x_{i-2}, e_{i-1}) \cap N(x_{i-1}, e_i) \in \Delta_{i-1}$. Otherwise, we say there is no CNR dependency between x_{i-1} and x_i ; we denote it by $x_{i-1} \ntriangleright x_i$ (see Figure 3.4).

Figure 3.4: Characterization of CNR dependencies by geometrical configurations.

If there is a CNR dependency between x_i and x_{i+1} , then $\Pi[x_{i-1}, x_{i+1}]$ and $\Pi[x_i, x_{i+2}]$ have to be refined simultaneously:

Lemma 9. *Assume $\Pi[x_{i-2}, x_i]$ obeys CNR. If $x_{i-1} \triangleright x_i$, refining $\Pi[x_{i-1}, x_{i+1}]$ w.r.t. CNR may cause $\Pi[x_{i-2}, x_i]$ to disobey CNR.*

Proof. As $x_{i-1} \triangleright x_i$, we have $N(x_{i-2}, e_{i-1}) \cap N(x_{i-1}, e_i) \in \Delta_{i-1}$, see Figure 3.4(a). As $\Pi[x_{i-2}, x_i]$ obeys CNR, x_{i-1} and x_i lie on the same side of $N(x_{i-2}, e_{i-1})$. Thus, translating x_i to $N(x_{i-1}, e_i) \cap e_i$ crosses $N(x_{i-2}, e_{i-1})$ which implies that x_{i-1} and x_i lie on different sides of $N(x_{i-2}, e_{i-1})$, which means $\Pi[x_{i-2}, x_i]$ disobeys CNR. $\square$

Therefore, we can propose a naïve algorithm to refine a linear path as follows: refine only $\Pi[x_{i-1}, x_{i+1}]$ w.r.t. CNR and CAR. If this causes $\Pi[x_{i-2}, x_i]$ to disobey CNR, refine $\Pi[x_{i-2}, x_i]$ recursively. This approach may lead to a quadratic running time (see Figure 3.5 for an example). In order to achieve a linear time refinement algorithm, we have to ensure that $x_{i-1} \ntriangleright x_i$, before refining x_i . In addition, if there is dependency, we perform the refinement of adjacent dependent corners as one bulk operation.

Definition 20. *For $d > 1$, $\langle x_i, \dots, x_{i+d} \rangle$ is a CNR dependency chain if $x_{i+j} \triangleright x_{i+j+1}$ for all $j = 1, \dots, d-1$.*

We consider a maximal CNR dependency chain $\Pi[x_i, x_{i+d}] = \langle x_i, \dots, x_{i+d} \rangle$ that $x_{i+d} \not\vdash x_{i+d+1}$ and replace $\Pi[x_i, x_{i+d}]$ by a refined chain. Hence, by induction, we guarantee that the final result is a refined path and a refinement of x_{i+d+1} does not affect the refinements made for $\Pi[x_i, x_{i+d}]$.

Figure 3.5: The directed red polygonal chain shows the sub-path from x_{i+1} to x_{k_3} in the original input path. The corner x_{k_1} is the first corner in this sub-path that is disobeying the CNR. The naïve approach replaces the sub-path from x_{i+1} to x_{k_1} by the orange polygonal chain whose first segment is the only segment from the forward chain. After that replacement, the naïve algorithm continues and finds the corner x_{k_2} that disobeys CNR. The algorithm replaces the sub-path from x_{i+2} to x_{k_2} by the blue polygonal chain. If the replacement happens $n/2$ times and the number of corners between x_{i+1} and x_{k_1} is $n/2$, then the total time complexity is quadratic in the size of the input path. In this example, the corner x_{k_3} also disobeys CNR after replacing the sub-path from x_{i+2} to x_{k_2} by the blue polygonal chain. The green polygonal curve shows the final replacement of the sub-path from x_{i+1} to x_{k_3} .

To ensure that replacing sub-paths of Π does not increase the length of Π , we need to ensure that the edges of the triangulation T that the replaced corners lie on, have a vertex $v \in V(T)$ in common. The existence of such a shared vertex can only be guaranteed if the last triple of the corners that has to be replaced disobeys CNR (see Figure 3.6). Thus, the sub-path that we replace is defined as follows: Let $x_{i+\ell}$ be the first corner of $x_i, \dots, x_{i+d}$ such

that $\Pi[x_{i+\ell-1}, x_{i+\ell+1}]$ disobeys CNR. We replace $\Pi[x_i, x_{\ell+1}]$ by a refined path. In particular, if all corners $x_i, \dots, x_{i+d}$ obey CNR nothing has to be done and we process x_{i+d+1} next. Before we describe how to refine $\Pi[x_i, x_{i+\ell+1}]$ we need some technical preparation. Let $N(x_{i-1}, e_i)$ be oriented from line segment e_i towards point x_{i-1} . Therefore, left and right of $N(x_{i-1}, e_i)$ are well-defined. The following two lemmas are needed to ensure that our replacement paths are well-defined.

Figure 3.6: A dependency chain $\Pi[x_i, x_{i+3}]$ is illustrated in red. The triple $\Pi[x_{i+1}, x_{i+3}]$ disobeys CNR.

Lemma 10. *If these three conditions hold (see Figure 3.4(a)): (1) x_{i-1} is on the left (right, respectively) of $N(x_{i-2}, e_{i-1})$, (2) $N(x_{i-2}, e_{i-1}) \cap e_{i-1} \neq \emptyset$, and (3) $x_{i-1} \triangleright x_i$, then v is on the right (left, respectively) side of $N(x_{i-2}, e_{i-1})$, where v is the vertex shared between e_{i-1} and e_i .*

Proof. W.l.o.g., we assume that x_{i-1} lies to the left of $N(x_{i-2}, e_{i-1})$. The case that x_{i-1} lies to the right of $N(x_{i-2}, e_{i-1})$ is symmetric. For the sake of contradiction, we assume that v lies to the left of $N(x_{i-2}, e_{i-1})$. As x_{i-1} lies to the left of $N(x_{i-2}, e_{i-1})$, it follows that either (a) $N(x_{i-2}, e_{i-1}) \cap e_{i-1} = \emptyset$ which contradicts the second condition of this lemma, or (b) the

intersection of the two straight lines $N(x_{i-2}, e_{i-1}) \cap N(x_{i-1}, e_i)$ lies outside of Δ_{i-1} which contradicts $x_{i-1} \triangleright x_i$. $\square$

Lemma 11. *Let $\langle x_i, \dots, x_{i+\ell} \rangle$ be a CNR dependency chain so that $\Pi[x_{i+\ell-2}, x_{i+\ell}]$ is the first triple that disobeys CNR. All the edges $e_{i+1}, \dots, e_{i+\ell-1}$ share a vertex $v \in V(T)$.*

Proof. W.l.o.g., we assume that x_{i+1} lies to the left of $N(x_i, e_{i+1})$. Let $v \in V(T)$ be the shared vertex between e_{i+1} and e_{i+2} . By Lemma 10, it follows that v lies on the right side of $N(x_i, e_{i+1})$. As the triple $\Pi[x_{i+1}, x_{i+3}]$ obeys CNR, it follows that x_{i+2} lies to the left of $N(x_{i+1}, e_{i+2})$. Lemma 10 implies that v is shared by e_{i+2} and e_{i+3} . By applying induction, we conclude the proof. $\square$

Now we are ready to give the definition of orthogonal chains that are our refined replacement chains. For simplicity, for two points x and y on an edge, $e = (u, v) \in E(T)$, we denote $x <_v y$, if $|\overline{vx}| < |\overline{vy}|$, where $|\cdot|$ is the Euclidean length of a line segment. Let $v \in V(T)$ and $\langle e_i, \dots, e_{i+\ell} \rangle$ be a list of edges of T . Suppose the edges $\langle e_{i+1}, \dots, e_{i+\ell-1} \rangle$ share the vertex v and the angle between any pair of adjacent edges is less than $\frac{\pi}{2}$.

Definition 21. (a) *Forward chain $\vec{C}$: For any point $f_0 \in e_i \setminus \{v\}$, let $\vec{C}(f_0) := \langle f_0, f_1, \dots, f_{\ell-1} \rangle$ where $f_j := N(f_{j-1}, e_{i+j}) \cap e_{i+j}$, for $j = 1, \dots, \ell - 1$ (see Figure 3.7).*

(b) *Backward chain $\overleftarrow{C}$: For any point $b_\ell \in e_{i+\ell} \setminus \{v\}$, let $\overleftarrow{C}(b_\ell) := \langle b_1, \dots, b_{\ell-1}, b_\ell \rangle$ where $b_j := N(b_{j+1}, e_{i+j-1}) \cap e_{i+j-1}$, for $j = \ell - 1, \dots, 1$.*

The two chains $\vec{C}(f_0)$ and $\overleftarrow{C}(b_\ell)$ intersect each other either once or not at all (Lemma 8 in Chapter 2). We define the result of merging two chains as follows:

(c) *Merged chain $\overleftrightarrow{C}$: For any $f_0 \in e_i \setminus \{v\}$ and any $b_\ell \in e_{i+\ell} \setminus \{v\}$:*

(1) *If $N(b_\ell, e_{i+\ell-1}) \cap e_{i+\ell-1} <_v f_{\ell-1}$ (Figure 3.7(a)), then $\overleftrightarrow{C}(f_0, b_\ell) := \langle f_0, \dots, f_{\ell-1}, b_\ell \rangle$.*

(2) *If $N(f_0, e_{i+1}) \cap e_{i+1} <_v b_1$ (Figure 3.7(b)), then $\overleftrightarrow{C}(f_0, b_\ell) := \langle f_0, b_1, \dots, b_\ell \rangle$.*

(3) Otherwise, suppose $\vec{C}(f_0)$ and $\overleftarrow{C}(b_\ell)$ intersect between the edges e_s and e_{s+1} (Figure 3.7(c)). Then, $\overleftrightarrow{C}(f_0, b_\ell) := \langle f_0, \dots, f_s, b_{s+1}, \dots, b_\ell \rangle$.

(d) *Orthogonal chain $\overleftrightarrow{OC}$* : For any $f_0 \in e_i \setminus \{v\}$ and any $b_\ell \in e_{i+\ell} \setminus \{v\}$, let $\overleftrightarrow{OC}(f_0, b_\ell)$ be the result of traversing the corners of $\overleftrightarrow{C}(f_0, b_\ell)$ from f_0 to b_ℓ and refining them so as to obey CAR.

Figure 3.7: Three different cases illustrate the relative positions of a forward (blue solid) and backward (red dashed) chain to each other.

Observation 3. Let $\langle x_i, \dots, x_{i+\ell} \rangle$ be a forward, backward, merged, or orthogonal chain. Every consecutive triple of $\langle x_i, \dots, x_{i+\ell} \rangle$ is refined w.r.t. CNR.

Unfortunately, the replacement of $\Pi[x_i, x_{i+\ell}]$ by $\overleftrightarrow{OC}(x_i, x_{i+\ell})$ may cause a new dependency, $x_{i+\ell} \triangleright x_{i+\ell+1}$, that did not exist before the replacement. An example is shown in Figure 3.8. Now, the question is: can we process the dependency chain $\Pi[x_i, x_{i+\ell}]$ such that refining $\Pi[x_{i+\ell-1}, x_{i+\ell+1}]$ has no influence on the refinement of $\Pi[x_i, x_{i+\ell}]$? The answer is, it depends on $\overleftrightarrow{OC}(x_i, x_{i+\ell})$.

There are three cases, described in Definition 21(c). If the condition of case (1) holds, we can process the dependency chain $\Pi[x_i, x_{i+\ell}]$ independently and replace it by $\overleftrightarrow{OC}(x_i, x_{i+\ell})$. By Observation 2, the corners of $\overleftrightarrow{OC}(x_i, x_{i+\ell})$ remain refined until the termination of the algorithm and it is not necessary to backtrack and refine them again. In Lemma 13, we

Figure 3.8: $\Pi[x_i, x_{i+l}] = \langle x_i, \dots, x_{i+l} \rangle$ is a dependency chain. Before replacing $\Pi[x_i, x_{i+l}]$ we have $x_{i+l} \not\triangleright x_{i+l+1}$ and afterwards $x_{i+l} \triangleright x_{i+l+1}$.

prove that case (2) of Definition 21(c) never occurs in our settings. Therefore, there is no need to handle this case. However, in case (3) of Definition 21(c), processing each dependency chain $\Pi[x_i, x_{i+l}]$ independently, leads to a quadratic running time (same argument for Figure 3.5). Thus, we apply the following approach iteratively, in this case: If $x_{i+l} \triangleright x_{i+l+1}$ after replacing $\Pi[x_i, x_{i+l}]$ by $\overleftrightarrow{OC}(x_i, x_{i+l})$, then, iteratively, we compute a dependency chain $\Pi[x_{i+l-1}, x_{i+l+\ell'}]$, where $\Pi[x_{i+l+\ell'-2}, x_{i+l+\ell'}]$ is the first triple of corners in $\Pi[x_{i+l-1}, x_{i+l+\ell'}]$ that disobeys CNR. Then, we substitute $\Pi[x_i, x_{i+l+\ell'}]$ by $\overleftrightarrow{OC}(x_i, x_{i+l+\ell'})$. Finally, we set $\ell := \ell + \ell'$ and repeat the above approach iteratively until $x_{i+l} \not\triangleright x_{i+l+1}$ holds. The last piece of the algorithm that is required to obtain a linear running time, is to decide if $x_{i+l} \not\triangleright x_{i+l+1}$ in constant time, without computing the whole $\overleftrightarrow{OC}(x_i, x_{i+l})$. Let x_{temp} be the intersection of $N(x_{i+l}, e_{i+l-1})$ and e_{i+l-1} . The geometric test is, if $N(x_{temp}, e_{i+l})$ and $N(x_{i+l}, e_{i+l+1})$ do not intersect inside Δ_{i+l} , then $x_{i+l} \not\triangleright x_{i+l+1}$, after the replacement.

In order to analyze the mentioned iterative approach for case (3) of Definition 21(c),

w.l.o.g., assume that x_i lies on the left side of $N(x_{i-1}, e_i)$. Note that in case (3) of Definition 21(c), after replacing $\Pi[x_i, x_{i+\ell}]$ by $\overleftarrow{OC}(x_i, x_{i+\ell})$, it follows that $x_{i+\ell}$ lies on the left side of $N(x_{i+\ell-1}, e_{i+\ell})$. We have the following lemma.

Lemma 12. *Assume x_i lies on the left side of $N(x_{i-1}, e_i)$. After replacing $\Pi[x_i, x_{i+\ell}]$ by $\overleftarrow{OC}(x_i, x_{i+\ell})$, if $x_{i+\ell}$ lies on the left side of $N(x_{i+\ell-1}, e_{i+\ell})$, then all the edges $e_{i+1}, \dots, e_{i+\ell'+1}$ share a vertex $v \in V(T)$, where $\Pi[x_{i+\ell-1}, x_{i+\ell'+1}]$ is a dependency chain and $\Pi[x_{i+\ell'+2}, x_{i+\ell'+1}]$ is the first triple of corners in $\Pi[x_{i+\ell-1}, x_{i+\ell'+1}]$ that disobeys CNR.*

Proof. By Lemmas 10 and 11, it follows that $e_i + 1, \dots, e_{i+\ell-1}$ share a vertex $v \in V(T)$ and v lies on the right side of $N(x_{i+\ell-2}, e_{i+\ell-1})$. Also, Lemmas 10 and 11 imply that the edges $e_{i+\ell}, \dots, e_{i+\ell'+1}$ share a vertex $v' \in V(T)$. We show that $v = v'$. As $x_{i+\ell}$ lies on the left side of $N(x_{i+\ell-1}, e_{i+\ell})$ and $x_{i+\ell} \triangleright x_{i+\ell+1}$, by Lemma 10, we have that v' lies on the right side of $N(x_{i+\ell-1}, e_{i+\ell})$. Therefore, as $N(x_{i+\ell-1}, e_{i+\ell}) \cap e_{i+\ell} \neq \emptyset$, $v = v'$. $\square$

Lemma 13. *Let $\Pi[x_i, x_{i+\ell}]$ be a dependency chain at which $\Pi[x_{i+\ell-2}, x_{i+\ell}]$ is the only consecutive triple that disobeys CNR. Then, $b_1 <_v N(f_0, e_{i+1}) \cap e_{i+1}$, where $\overrightarrow{C}(x_i = f_0) = \langle f_0, f_1, \dots, f_\ell \rangle$ and $\overleftarrow{C}(x_{i+\ell} = b_\ell) = \langle b_1, \dots, b_\ell \rangle$.*

Proof. Lemma 11 implies that the edges $e_{i+1}, \dots, e_{i+\ell-1}$ share a vertex $v \in V(T)$. Let $p_j := N(x_{i+j}, e_{i+j+1}) \cap e_{i+j+1}$ for $j = 0, \dots, \ell - 2$. By induction, we show that $b_{j+1} <_v p_j$. This concludes the proof for the case that $b_1 <_v p_0 = N(f_0, e_{i+1}) \cap e_{i+1}$.

W.l.o.g., assume that x_i lies on the left side of $N(x_{i-1}, e_i)$. For the base case of the induction, since $\Pi[x_{i+\ell-2}, x_{i+\ell}]$ is the only triple that disobeys CNR, we know that $x_{i+\ell-1}$ lies on the left side of $N(x_{i+\ell-2}, e_{i+\ell-1})$ and $x_{i+\ell}$ lies on the right side of $N(x_{i+\ell-2}, e_{i+\ell-1})$. This implies $b_{\ell-1} <_v p_{\ell-2}$ because $\overline{b_{\ell-1}x_{i+\ell}}$ lies parallel to $N(x_{i+\ell-2}, e_{i+\ell-1})$ (see Figure 3.9(a)). Now, assume $b_{j+2} <_v p_{j+1}$. Since we assumed that x_i lies on the left side of $N(x_{i-1}, e_i)$ and $\Pi[x_{i+\ell-2}, x_{i+\ell}]$ is the only triple that disobeys CNR, we have that x_{i+j+2} lies on the left side of $N(x_{i+j+1}, e_{i+j+2})$, for a $j < \ell - 3$. In addition, p_{j+1} lies on the right side of $N(x_{i+j}, e_{i+j+1})$,

because $x_{i+j+1} \triangleright x_{i+j+2}$. Combining that with $b_{j+2} <_v p_{j+1}$ implies that b_{j+2} lies on the right side of $N(x_{i+j}, e_{i+j+1})$. As $\overline{b_{j+2}b_{j+1}}$ lies parallel to $N(x_{i+j}, e_{i+j+1})$, we obtain $b_{j+1} <_v p_j$ (see Figure 3.9(b)).

Figure 3.9: Configurations of the inductive proof for Lemma 13.

□

The proposed algorithm in this section is summarized in Algorithm 2.

Correctness proof of the algorithm

Lemma 14. *The path, Π' , computed by Algorithm 2 is a refined path and $\|\Pi'\| \leq \|\Pi\|$.*

Proof. First we show that Π' is refined. Then, we prove $\|\Pi'\| \leq \|\Pi\|$.

Each corner is refined at least once w.r.t. CAR. By Observation 1 it follows that the refinement of a corner w.r.t. CAR is maintained for all subsequent refinement steps of the algorithm. Assume that $\Pi[x_{i-1}, x_{i+1}]$ is the last triple that was refined w.r.t. CNR by the **for**-loop of Algorithm 2 such that the following holds: All consecutive triples of $\Pi[x_0, x_i]$ obey CAR and CNR and a possible refinement of $\Pi[x_i, x_{i+2}]$ has no influence on

Algorithm 2 Refinement Algorithm for Triangulations

Input: Triangulation T , source and target points s and t , linear path $\Pi = [x_0, \dots, x_L]$

Output: Refined path $\Pi' = [x_0, \dots, x_L]$ such that $\|\Pi'\| \leq \|\Pi\|$

```
1: for  $i=1$  to  $L-1$  do
2:   if  $x_i \notin V(T)$  then
3:     if  $w_{i-1} > w_i$  then
4:       if  $x_i$  disobeys CAR then
5:          $x_i$  = closest point from  $Passage_{x_{i-1}, e_i}$  to  $x_i$ 
6:     if  $x_i$  disobeys CNR then
7:        $x_i = N(x_{i-1}, e_i) \cap e_i$ 
8:       Mark  $x_i$  as refined for CNR
9:   else
10:     $d = 1$ 
11:     $last = NIL$ 
12:    while  $x_{i+d-1} \triangleright x_{i+d}$  do
13:      if  $x_{i+d} \notin Passage_{x_{i+d-1}, e_{i+d}}$  then
14:         $x_{i+d}$  = closest point from  $Passage_{x_{i+d-1}, e_{i+d}}$  to  $x_{i+d}$ 
15:      if  $\Pi[x_{i+d-1}, x_{i+d+1}]$  disobeys CNR then
16:         $last = i + d + 1$ 
17:         $x_{i+d-1} = N(x_{i+d}, e_{i+d-1}) \cap e_{i+d-1}$ 
18:         $v = e_{i+d-1} \cap e_{i+d}$ 
19:        if  $x_{i+d-1} <_v \vec{C}(x_i) \cap e_{i+d-1}$  then
20:          break
21:         $d = d + 1$ 
22:    if  $last \neq NIL$  then
23:      Replace  $\Pi[x_i, x_{last}]$  by  $\overleftarrow{OC}(x_i, x_{last})$ 
24:    Continue the for loop with index  $i = i + d$ 
25: return  $\Pi$ 
```

the refinement of $\Pi[x_{i-1}, x_{i+1}]$. Based on this, we show the following: (1) After the current iteration of the **for**-loop, all consecutive triples of $\Pi[x_i, x_{last}]$ also obey CNR and CAR. (2) These refinements have no influence on the refinements of the triples from $\Pi[x_0, x_{i+1}]$. (3) Furthermore, a possible refinement of a sub-path $\Pi[x_{\ell-1}, x_{\ell+1}]$ with $\ell \geq last$ has no influence on the refinements of $\Pi[x_0, x_{last}]$. Based on (1), (2), and (3), by induction, it follows that Π' is refined.

(1) In Line 14, if necessary, we refine $\Pi[x_i, x_{i+2}]$ w.r.t. CAR. After that, we scan Π from

x_i to x_{last} . To replace the subpath between x_i and x_{last} by $\overleftrightarrow{OC}(x_i, x_{last})$, the corners $x_{i+1}, \dots, x_{last-1}$ are computed as a sequence of subsequences. Each subsequence is a dependency chain starting from the last corner of the previous subsequence. The last triple of each subsequence disobeys CNR. By applying Lemma 13, we obtain that all the edges which are traversed by the subpath from x_{i+1} to x_{last-1} share a common vertex $v \in V(T)$. Thus, we can construct an orthogonal chain (since all share $v \in V(T)$). By Observation 3, we obtain that each consecutive triple of $\Pi[x_i, x_{last}]$ is refined, after replacing the subpath from x_i to x_{last} by $\overleftrightarrow{OC}(x_i, x_{last})$.

- (2) We assumed that refining $\Pi[x_i, x_{i+2}]$ has no influence on the refinement of $\Pi[x_{i-1}, x_{i+1}]$. Thus, by induction, it follows that refining any triple of $\Pi[x_i, x_{last}]$ has no effect on the refinement of any triple of $\Pi[x_0, x_{i+1}]$.
- (3) In the current iteration of the **for**-loop, the subpath $\Pi[x_i, x_{last}]$ is replaced by the corresponding orthogonal chain. This orthogonal chain is the result of merging a forward chain $\overrightarrow{C}(x_i)$ and a backward chain $\overleftarrow{C}(x_{last})$. We distinguish between Cases 1 and 3 of Definition 21(c):

Case 1) We know that, after replacing $\Pi[x_i, x_{last}]$ by $\overleftrightarrow{OC}(x_i, x_{last})$, the corner x_{last-1} is the intersection of $N(x_{last-2}, e_{last-1})$ and e_{last-1} . Thus, by Observation 2, refining $\Pi[x_{last-1}, x_{last+1}]$ in following iterations of the **for**-loop will not effect the refinement of $\Pi[x_{last-2}, x_{last}]$. This proves (3), by induction.

Case 3) By the condition of Line 12, there is no CNR dependency between x_{last-1} and x_{last} . Thus, refining $\Pi[x_{last-1}, x_{last+1}]$ has no influence on the refinement of $\Pi[x_{last-2}, x_{last}]$. This fact, by induction, proves (3).

Now, we prove $\|\Pi'\| \leq \|\Pi\|$. As refining any triple of Π w.r.t. CAR and/or CNR does not increase $\|\Pi\|$, it remains to show the following: Let $\Pi[x_i, x_{last}]$ be a sub-path that is replaced

by $\overleftrightarrow{OC}(x_i, x_{last})$, in Line 23 of the algorithm. We prove $\|\overleftrightarrow{OC}(x_i, x_{last})\| \leq \|\Pi[x_i, x_{last}]\|$ that concludes the proof.

As $\|\overleftrightarrow{C}(x_i, x_{last})\| \geq \|\overleftrightarrow{OC}(x_i, x_{last})\|$, it suffices to show $\|\Pi[x_i, x_{last}]\| \geq \|\overleftrightarrow{C}(x_i, x_{last})\|$. To do so, we first show that (I) $x_l^* \prec_v x_l$, for all $l = i, \dots, last$, where x_l^* is the corner from $\overleftrightarrow{C}(x_i, x_{last})$ that corresponds to x_l . Then, we show (II) $|\overline{x_l x_{l+1}}| \geq |\overline{x_l^* x_{l+1}^*}|$, where $|\cdot|$ is the Euclidean length. This implies $\|\Pi[x_i, x_{last}]\| \geq \|\overleftrightarrow{C}(x_i, x_{last})\|$.

(I) Let $\Pi[x_i, x_{k_1}]$ and $\Pi[x_{k_1}, x_{k_2}]$ be two dependency chains, which are subsequences of $\Pi[x_i, x_{last}]$. Let $\Pi[x_{k_1-2}, x_{k_1}]$ and $\Pi[x_{k_2-2}, x_{k_2}]$ disobey CNR. These two subsequences share a corner x_{k_1} , see Figure 3.10. By Lemma 10, it follows that $x_l^* \prec_v x_l$ holds for all the corners of $\vec{C}(x_i)$, for $l = i, \dots, k_1 - 1$. Since there is a dependency between $x_{temp} = N(x_{k_1}, e_{k_1-1}) \cap e_{k_1-1}$ and x_{k_1} , it follows that x_{k_1} lies to the left of $N(x_{temp}, e_{k_1})$ (Figure 3.10). By combining this with $x_{k_1-1}^* \prec_v x_{k_1-1}$, we obtain that x_{k_1} lies on the left side of $N(x_{k_1-1}^*, e_{k_1})$. Therefore, $x_{k_1}^* \prec_v x_{k_1}$. By induction it follows $x^* \prec_v x$ for each corner x of $\Pi[x_i, x_{last}]$. A symmetric argumentation applies to $\overleftarrow{C}(x_{last})$.

Figure 3.10: Corners of forward and backward chains lie closer to v than $x_i, \dots, x_{last}$.

(II) Each segment $\overline{x_l^* x_{l+1}^*}$, $l = i, \dots, last - 1$, of $\vec{C}(x_i, x_{last})$, is either (a) one of the segments in $\vec{C}(x_i)$ or $\overleftarrow{C}(x_{last})$ or (b) it is introduced by merging of the forward and backward chain. In case (a), the segment $\overline{x_l^* x_{l+1}^*}$ is the shortest segment either from x_l^* to e_{l+1} or

from x_{l+1}^* to e_l . Combining this with $x_l^* <_v x_l$ and $x_{l+1}^* <_v x_{l+1}$ implies $|\overline{x_l^* x_{l+1}^*}| \leq |\overline{x_l x_{l+1}}|$. Since there is at most one intersection between $\vec{C}(x_i)$ and $\overleftarrow{C}(x_{last})$, there is at most one segment of case (b) in $\overleftrightarrow{C}(x_i, x_{last})$. We denote this segment by $\overline{x_s^* x_{s+1}^*}$ (Figure 3.11). Note that the distance function from a point to a line segment is a convex function. Therefore, $|\overline{x_s^* x_{s+1}^*}| \leq |\overline{x_s x_{s+1}}|$ as $x_s^* <_v x_s$ ($x_{s+1}^* <_v x_{s+1}$, resp.) which makes x_s^* (x_{s+1}^* , resp.) closer to $N(x_{s+1}^*, e_s) \cap e_s$ ($N(x_s^*, e_{s+1}) \cap e_{s+1}$, resp.) in comparison with x_s (x_{s+1} , resp.).

Figure 3.11: Segment $\overline{x_s^* x_{s+1}^*}$ is introduced by the merge operation. We prove $|\overline{x_s^* x_{s+1}^*}| \leq |\overline{x_s x_{s+1}}|$.

□

Theorem 3. *Let T be a weighted triangulation and s and t be two points in T . Suppose a linear path Π from s to t , is given as the input to Algorithm 2. Then, Algorithm 2 produces a refined path Π' , in linear time in the size of the input path, such that $||\Pi'|| \leq ||\Pi||$.*

Proof. The correctness follows directly from Lemma 14. To establish the time complexity, note that at each iteration of the **for**-loop in Algorithm 2, at least one corner of the input path is processed. Since a corner is refined at most a constant number of times (at constant cost) and the time complexity of the replacement operation of a dependency chain is linear in the size of the dependency chain, the total time complexity is linear in the size of the input path, Π . □

Corollary 4. *If the input path Π to the refinement algorithm is an $(1 + \varepsilon)$ -approximation of a shortest path from s to t in T , then the output path Π' is a refined $(1 + \varepsilon)$ -approximation of the shortest path.*

3.3.2 Refinement Algorithm for Parallel Lines

In this section, we discuss the PRWR problem when the partitioning of $\mathbb{R}^2$ is induced by n parallel lines. Without loss of generality, we assume that these n parallel lines are horizontal in the Cartesian coordinate system. More formally, assume a sorted sequence of n parallel lines, $\langle I_1 \parallel I_2 \parallel \dots \parallel I_n \rangle$ is given. The line I_i is higher than the line I_{i+1} , in the Cartesian coordinate system. Between each two consecutive lines, I_i and I_{i+1} , a real positive weight, w_i , is assigned. The open half-plane above I_1 has weight $w_0 > 0$. The open half-plane below I_n has weight $w_n > 0$. By convention, the weight of line I_i is the minimum of w_{i-1} and w_i . Also two points, the source point, s , above I_1 , and the target point, t , below I_n , are given (see Figure 3.12a).

Figure 3.12: a) Parallel lines with source and target points. b) Local refinement on I_i is shown. $Cone_{x_{i-1}, I_i}$ is hatched. Solid line between x_{i-1} and x_{i+1} shows the path before the translation of x_i and dashed line is the path after refinement.

In this section, since the partitioning is induced by n horizontal parallel lines, the following observations are easily seen.

Observation 4. *Let s_x be the x -coordinate of the source point and t_x be the x -coordinate of the target point. Assume that $t_x < s_x$. Then, the x -coordinates of the vertices of the shortest path from s to t are sorted in decreasing order from s_x to t_x . Therefore, the shortest path lies between two vertical lines, $N(s, I_1)$ and $N(t, I_1)$ (Figure 3.12a).*

Observation 5. *The shortest path from s to t intersects each line only at one point.*

Therefore, we assume that the input to the refinement algorithm intersects each line only at one point and the x -coordinates of the corners of the path are sorted from s_x to t_x . Because of these assumptions, the refinement algorithm for n parallel lines is much simpler than the refinement algorithm for triangulations. The refinement algorithm for parallel lines is summarized in Algorithm 3. It is easy to see that there is no need to maintain the CNR dependencies, when the input partitioning is a set of parallel lines (by Observation 4 and 5). It is guaranteed that this algorithm does not increase the length of the input path.

More formally, suppose $\Pi = [s = x_0, x_1, \dots, x_n, x_{n+1} = t]$ is the input path to Algorithm 3, where $x_i \in I_i$, for $i = 1, \dots, n$. The algorithm traverses the corners, x_i , $i = 1, \dots, n$, of the input path sequentially. It fixes the position of the corner x_i only based on its adjacent corner (i.e., x_{i-1} and x_{i+1}). At the i -th step of this refinement algorithm, x_i may be translated twice on I_i : one refinement to obey CAR and one to obey CNR. Note that the sub-path from s to x_{i-1} and the sub-path from x_{i+1} to t do not change at the step i of the algorithm (Figure 3.12b). The only substituted sub-path is from x_{i-1} to x_{i+1} which is shorter or has the same length.

Although, at the $(i+1)$ -st step of the algorithm, x_{i+1} may be translated, it will not make the refinement for x_i invalid. The reason is, firstly, for CAR, the $Passage_{x_{i-1}, I_i}$ is defined based on only the position of x_{i-1} . Therefore, $[x_{i-1}, x_i, x_{i+1}]$ still obeys CAR, independent

of the position of x_{i+1} . Secondly, for CNR, the translation of x_{i+1} does not pass over the perpendicular line from x_i to I_{i+1} , $N(x_i, I_{i+1})$. Therefore, both x_i and x_{i+1} remain in one closed half-plane of the perpendicular line from x_{i-1} to I_i . Thus, $[x_{i-1}, x_i, x_{i+1}]$ still obeys CNR. Therefore, at the $(i+1)$ -st step, all previous refinements remain valid. Note that, the time complexity of the refinement algorithm is linear in the size of the input path.

Theorem 4. *Let $\langle I_1 \parallel I_2 \parallel \dots \parallel I_n \rangle$ be a sequence of n parallel lines. Between each two consecutive lines, I_i and I_{i+1} , a real positive weight, w_i , is assigned. Also, suppose Π is a linear input path to Algorithm 3 that intersects I_i at point x_i , $i = 1, \dots, n$. The output path, Π' , of Algorithm 3, is a refined path such that $\|\Pi'\| \leq \|\Pi\|$. In addition, the time complexity of this algorithm is $\mathcal{O}(n)$.*

Algorithm 3 Path Refinement Algorithm for Parallel Lines

Input: source (s), target (t), a sequence of n parallel lines $\langle I_1 \parallel I_2 \parallel \dots \parallel I_n \rangle$, a linear path $\Pi = [s = x_0, x_1, \dots, x_n, x_{n+1} = t]$ that intersects I_i at point x_i

Output: A refined path Π' s.t. $\|\Pi'\| \leq \|\Pi\|$

```

1:  $x_0 = s$ .
2:  $x_{n+1} = t$ .
3: for  $i=1:n$  do
4:   if  $w_i < w_{i-1}$  then
5:     Find  $Passage_{x_{i-1}, I_i}$ ;
6:     if  $x_i$  is not inside the  $Passage_{x_{i-1}, I_i}$  then
7:        $bestEnd = \text{find closest endpoint of } Passage_{x_{i-1}, I_i} \text{ to } x_i$ .
8:        $x_i = bestEnd$ .
9:   if the subpath  $[x_{i-1}, x_i, x_{i+1}]$  is not in one closed half-plane of  $N(x_{i-1}, I_i)$  then
10:     $bestEnd = \text{the intersection of } N(x_{i-1}, I_i) \text{ and } I_i$ ;
11:     $x_i = bestEnd$ .
```

3.3.3 Refinement Algorithm for Arrangements of Lines

In this section, we discuss the refinement algorithm when the partitioning of $\mathbb{R}^2$ is induced by an arrangement of n lines, I_i , $i = 1 \dots n$. Before continuing in this section, note that if we find a closed region in $\mathbb{R}^2$ that contains a shortest path from s to t , then we can clip

the unbounded faces of the arrangement and by triangulating the faces, we can use the algorithms mentioned in the previous section. As we mentioned, in the case that all lines are parallel, Observation 4 constrains the regions that a shortest path lies in. We need a similar closed region for a general arrangement of lines.

Therefore, the formal statement of the problem that we are facing is as follows. Let s and t be two points in the plane $\mathbb{R}^2$ and let $\mathcal{A}$ be an arrangement of n lines I_i , $i = 1 \dots n$. For simplicity, assume no two lines in $\mathcal{A}$ are parallel to each other and no three lines have a common intersection. Each face of $\mathcal{A}$ is assigned positive weight w_i . By convention, the weight of each edge of $\mathcal{A}$ is the minimum of the weights of its adjacent faces. The task is to find a closed region in $\mathbb{R}^2$ that contains a shortest path from s to t .

In Chapter 2, we presented a minimal region that contains a shortest path from s to t , called SP-Hull of $\mathcal{A}$. It is a closed polygonal region that is independent of the weights assignment. After bounding the area, by triangulating the bounded faces it is possible to use Algorithm 2 to refine an input linear path.

Theorem 5. *Let $\mathcal{A}$ be an arrangement of n lines I_i , $i = 1 \dots n$ and s and t be two points in the plane. Also, suppose Π is a linear input path to the refinement algorithm. Then:*

- *The output, Π' , of the algorithm is a refined path such that $\|\Pi'\| \leq \|\Pi\|$.*
- *The time complexity for the refinement algorithm is linear in the size of the input path.*

3.4 Experimental Results

3.4.1 Motivation

In order to examine the experimental performance of the refinement algorithm in a practical setting, we implemented it for triangulations. The goal in this experiment is to find the answer to the following questions: Can a local refinement compensate for a coarser discretization of the geometric space? In other words, is a coarse discretization followed by

refinement as good as a finer discretization? If so, how much memory and query time will be saved when the quality of the output is to remain close to optimal?

The result of this experiment shows that the saving in query time and memory usage by refinement algorithm is significant.

3.4.2 Experimental Setup

Data

We gathered fourteen Triangular Irregular Networks (TINs). Since TIN generation process could potentially have an influence, we used two different processes. Figure 3.13 shows these two processes to capture TINs. The TINs are listed in Table 3.1. They are from different parts of the Earth’s terrain and have different geographic characteristic (for example, Everest TIN shown in Figure 3.14 has steep faces compared to Damavand TIN). These TINs are available at [87]¹. For each TIN in Table 3.1, 120 queries (i.e., pairs of source and target points) are generated randomly, in three different categories: short, medium and long distances.

Figure 3.13: Two different procedures to capture TINs for the experiment.

¹The format is raw text and each line corresponds to a triangle with 3 vertices in 3D space.

Table 3.1: 14 Triangular Irregular Networks (TINs) for experiments

TIN name	Number of faces
Alborz	3026
Damavand	3319
Everest	2817
Grand Canyon	3032
Uttarkhand	3471
val-des-monts	3710
Token Peak	4664
Matterhorn	3214
Heart Peaks	3624
Tseax Cone	3810
K2	3322
Thompson	3559
Honolulu	3099
Helens	4852

Implementation

To implement the algorithms we used:

1. C++, Microsoft Visual Studio®
2. CGAL (V3.8) for the triangulation and basic geometric operations
3. Boost library (V1.46.1) for graph representations and Dijkstra's Algorithm
4. Qt library (V4.7.3) for graphical user interface

The experiment environment was a PC with the following specifications:

1. CPU: Intel Core® i7, 2.67 GHz
2. Operating System: Microsoft Windows® 7, 64-bit
3. RAM: 12.0 GB

Figure 3.14: Top, front, and perspective view of Everest TIN

To evaluate the performance of the refinement algorithm, we re-implemented the algorithm proposed in [95]. The reason to choose this algorithm is, first, it is very easy to implement, and second, it is widely used in different fields [92, 93]. Lanthier et al. [95] reported that, in practice, on average six Steiner points per edge in Interval scheme suffice to obtain a close-to-optimal approximation. In this scheme, the length of the intervals between Steiner points are chosen such that the average number of Steiner points per edge is six. Afterwards, for each triangle, all Steiner points on its edges (including the vertices of the triangle) are connected together to form a locally complete graph for each triangle of the triangulation. The graph generated by the Interval Scheme using, on average, i Steiner points per edge is called G_i . We take G_6 as a baseline for comparisons.

In this experiment, we have three different post-processing methods. We compare the results of these methods in terms of resources usage and accuracy. The accuracy of each method is defined as the percentage of queries for which the resulting path is better (i.e., shorter) than in G_6 . These three methods are as follows:

Table 3.2: Comparing refinement process and enhanced sleeve methods

	Advantages	Disadvantages
refinement process	<ul style="list-style-type: none"> • Fast refinement process • No extra memory usage • Applicable to a part of a path 	<ul style="list-style-type: none"> • No improvement when the input path travels through the vertices
enhanced sleeve	<ul style="list-style-type: none"> • Explore around the vertices if the path passes through them • User control on the number of Steiner points 	<ul style="list-style-type: none"> • Needs extra memory per query • limited to the position of Steiner points

- **Refinement Process:** The first method is refining the resulting path for the queries in G_i , $i = 2, \dots, 5$.
- **Enhanced Sleeve:** The second method is sleeve computation [95] as a post-processing on the result of the query in G_i , $i = 2, \dots, 5$. In this method, the triangles that coincide with the path are discretized for the second time with more Steiner points. Here, the average number of Steiner points per edge in the sleeve is twice the average number of Steiner points in G_i , $i = 2, \dots, 5$ (but due to the restriction, proposed by the sleeve, the total number of Steiner points in the sleeve is much less).
- **Hybrid:** The third method is the combination of the two first methods. It applies the refinement process on the resulting path produced by the enhanced sleeve method.

Table 3.2 compares the refinement process and the enhanced sleeve methods. The hybrid method does have all of the benefits.

In this experiment, the weight of a triangle in TIN is defined as the slope of that face. We also tested the performance of the algorithm when the weights are random numbers (not

slopes).

3.4.3 Results

First, we ran experiments on the first five TINs in Table 3.1 to determine which of the G_i , $i = 2, \dots, 5$, achieves a good trade-off between resource usage (time and memory) and accuracy. Those results are shown in Table 3.3. G_3 followed by the hybrid method has good accuracy and shows significant savings in query time (close to 50 percent) and memory usage (close to 70 percent). Based on that, G_3 was chosen. Then, we ran the experiments using G_3 on the other TINs. The result is summarized in Table 3.4.

In addition, in Figure 3.15a, b, c, d and e, the accuracies of the post-processing methods on G_3 are plotted, for the first five TINs. In these plots, for each TIN, there are five bins. The first bin, called Goal, shows the percentage of the queries that reached the baseline quality (i.e., the path length in G_6). The second, third, and fourth bins show the percentages of the queries that reached at least 99, 95 and 90 percent of the baseline quality, respectively.

In Table 3.3, a row is interpreted as follows: The first column shows which G_i , $i = 2, \dots, 6$, is used to answer random path queries. From second to fifth column, are the result of the answering queries without applying any of the three post-processing methods, mentioned in Subsection 3.4.2. We report the average query time (in milliseconds) and average memory usage (the size of the graph in terms of the number of edges). For each of the random queries, the path length in G_6 is measured. Then, the result of applying each of the three post-processing methods are shown in terms of the percentage of the baseline. For a query in the first method, the query time is the sum of the query time in G_i , $i = 2, \dots, 5$, and the time for refinement process. Also, the memory usage in this method is equal to the size of G_i , $i = 2, \dots, 5$, (i.e., number of edges of the graph). In the second method, the query time is the sum of the query time in G_i , $i = 2, \dots, 5$, and the times to compute the sleeve and find the path in its corresponding graph. The memory usage in the enhanced sleeve method is

the sum of the sizes of two graphs: one is G_i , $i = 2, \dots, 5$ and the other one is the graph for the sleeve. The third method, hybrid, has the same memory usage as the enhanced sleeve method. We reported the averages of memory usage for both all queries and top 5 percent memory usages. However, the query time includes an extra refinement processing time. For example, the fourth row in Table 3.3 shows the results for queries on G_3 , on Alborz TIN. The fifth column shows that the accuracy of using G_3 to answering queries without applying any of the post-processing methods is less than 1 percent (i.e. 0.83 percent). The next three columns show that by applying the refinement process the accuracy increases to 45 percent while more than 70 percent is saved on memory usage and query time. Based on this row, the accuracy of the enhanced sleeve method is 5 percent less than the first method, while it uses more resources. By applying the hybrid method, the accuracy increases significantly to 84 percent. In comparison to the baseline, this saves 70 percent in memory usage and is 46 percent faster. The last column of the table shows the top 5 percent memory usages among all 120 queries, which is 35 percent.

The results on the Everest TIN with weights rather than slopes are also shown in Table 3.5. We had two different input weighted TINs: Random Everest and Flat Everest. For Random Everest, we assigned random weights to the faces of the Everest TIN (see accuracy plot in Figure 3.15f). For Flat Everest, we projected the Everest TIN onto the xy plane in Cartesian coordinate system, followed by a rotation of the xy plane by 25 degrees. By the 25 degrees rotation, we assure that the weight (i.e., slope) of each triangle is not zero. This created a Flat TIN on which all faces have equal weights. On Flat Everest, the shortest path is the straight line from source point to target point.

Table 3.3: The result of the methods on five TINs. Number of Edges of the Graph ($|E|$), Pre-processing Time (T_p), Average Query Time (QT_{av}), Accuracy (AC), Average Memory Usage (M_{av}), Top 5 percent Average Memory Usage (TM_{av}), Method 1: Refinement Process, Method 2: Enhanced Sleeve, Method 3: Hybrid

	$ E $	T_p	Refinement Process			Enhanced Sleeve			Hybrid						
			M_{av} %	QT_{av} %	AC %	M_{av} %	QT_{av} %	AC %	M_{av} %	QT_{av} %	AC %	TM_{av} %			
Alborz															
G_6	365793	13828.00	3929.90	-	-	-	-	-	-	-	-	-	-	-	
G_5	261098	10261.00	2854.40	19.83	72.92	70.49	71.38	72.92	79.02	115.74	93.44	79.02	116.13	100.00	90.92
G_4	174045	7175.00	1924.78	14.88	49.25	73.77	47.58	49.25	52.65	80.69	74.59	52.65	81.06	98.36	60.55
G_3	102081	4631.00	1176.27	0.83	30.19	45.08	27.91	30.19	30.87	52.79	40.16	30.87	53.13	84.43	35.48
G_2	47593	2807.00	572.81	0.83	14.85	19.67	13.01	14.85	14.45	29.43	8.20	14.45	29.75	64.75	16.70
Damavand															
G_6	391388	14810.00	4203.40	-	-	-	-	-	-	-	-	-	-	-	-
G_5	272552	10483.00	2959.33	29.84	70.73	87.10	69.64	70.73	77.13	114.84	96.77	77.13	115.29	100.00	88.17
G_4	182619	7423.00	2027.43	16.94	48.56	74.19	46.66	48.56	51.62	80.12	86.29	51.62	80.56	96.77	59.02
G_3	108549	4850.00	1245.37	4.03	29.93	49.19	27.73	29.93	30.62	51.46	43.55	30.62	51.87	95.97	34.98
G_2	51156	2780.00	617.43	0.81	14.96	21.77	13.07	14.96	14.47	30.00	14.52	14.47	30.31	63.71	16.59
Everest															
G_6	341559	13009.00	3617.17	-	-	-	-	-	-	-	-	-	-	-	-
G_5	239948	9317.00	2622.29	25.42	72.91	89.83	70.25	72.91	79.27	124.98	96.03	79.27	125.50	99.21	91.30
G_4	157325	6502.00	1765.43	10.83	49.21	75.40	46.06	49.21	51.95	86.90	94.44	51.95	87.38	98.41	59.89
G_3	92372	4474.00	1099.56	1.68	30.75	49.21	27.04	30.75	30.52	56.63	55.56	30.52	57.07	96.83	35.14
G_2	44506	2570.00	535.45	1.68	15.14	18.25	13.03	15.14	14.67	32.80	18.25	14.67	33.20	73.81	16.82
GrandCanyon															
G_6	372532	13836.00	3963.36	-	-	-	-	-	-	-	-	-	-	-	-
G_5	258613	10052.00	2799.55	29.75	70.93	91.74	69.42	70.93	82.79	119.61	86.78	82.79	119.99	98.35	95.31
G_4	172263	6996.00	1919.18	15.70	48.71	72.73	46.24	48.71	55.11	83.29	76.03	55.11	83.67	97.52	63.39
G_3	100931	4606.00	1170.74	4.13	29.80	51.24	27.09	29.80	32.31	52.42	44.63	32.31	52.79	91.74	37.21
G_2	48827	2758.00	588.26	6.61	15.09	33.88	13.11	15.09	15.60	30.27	20.66	15.60	30.57	71.90	18.01
Uttarakhand															
G_6	414758	15647.00	4445.72	-	-	-	-	-	-	-	-	-	-	-	-
G_5	294690	11270.00	3184.71	30.65	71.92	90.32	71.05	71.92	76.87	105.88	94.35	76.87	106.22	100.00	87.01
G_4	195103	7979.00	2160.77	15.32	48.85	72.58	47.04	48.85	50.84	73.51	80.65	50.84	73.79	95.16	57.47
G_3	114537	5145.00	1316.59	3.23	29.85	57.26	27.62	29.85	29.85	46.71	39.52	29.85	46.99	91.13	33.70
G_2	53163	3024.00	638.55	0.00	14.57	29.84	12.82	14.57	13.90	26.12	9.68	13.90	26.36	67.74	15.77

Figure 3.15: The accuracies of the post-processing methods on G_3 are plotted, for a) Alborz TIN, b) Damavand TIN, c) Everest TIN, d) Grand Canyon TIN, e) Uttarakhand TIN, f) Everest TIN with random associated weights to faces. In these plots, the first bin, called Goal, shows the percentage of the queries that reached the baseline quality (i.e., the path length in G_6). The second, third, and fourth bins show the percentages of the queries that reached at least 99, 95 and 90 percent of the baseline quality, respectively.

Table 3.4: The result of the methods on other TINs. Number of Edges of the Graph ($|E|$), Pre-processing Time (T_p), Average Query Time (QT_{av}), Accuracy (AC), Average Memory Usage (M_{av}), Top 5 percent Average Memory Usage (TM_{av}), Method 1: Refinement Process, Method 2: Enhanced Sleeve, Method 3: Hybrid

	$ E $	Refinement Process			Enhanced Sleeve			Hybrid						
		M_{av} %	QT_{av} %	AC %	M_{av} %	QT_{av} %	AC %	M_{av} %	QT_{av} %	AC %	TM_{av} %			
Heart Peaks														
G_6	436914	16941	4661.43	-	-	-	-	-	-	-	-	-	-	
G_3	124761	6066	1453.59	3.33	28.56	31.46	34.43	31.12	52.08	36.89	31.12	53.27	84.43	34.04
K2														
G_6	396876	15333	4264.75	-	-	-	-	-	-	-	-	-	-	-
G_3	105138	5285	1227.34	8.20	26.49	28.99	52.55	28.41	43.24	61.31	28.41	44.29	94.89	31.57
Matterhorn														
G_6	400234	15156	4299.79	-	-	-	-	-	-	-	-	-	-	-
G_3	106122	5152	1246.40	2.40	26.51	29.22	48.00	28.65	45.42	59.20	28.65	45.69	93.60	31.81
Thompson														
G_6	450777	17419	4835.40	-	-	-	-	-	-	-	-	-	-	-
G_3	130443	6317	1493.11	5.88	28.94	31.25	39.17	32.18	59.85	23.33	32.18	62.28	84.17	35.70
Token Peak														
G_6	546650	21007	5769.70	-	-	-	-	-	-	-	-	-	-	-
G_3	150273	7480	1772.83	0.00	27.49	30.98	47.50	29.76	50.08	63.33	29.76	51.09	91.53	32.22
Tseax cone														
G_6	457815	17783	4899.44	-	-	-	-	-	-	-	-	-	-	-
G_3	130816	6242	1560.59	2.50	28.57	32.09	43.33	30.71	49.85	35.83	30.71	50.74	85.83	33.30
val-des-monts														
G_6	437966	16961	4787.39	-	-	-	-	-	-	-	-	-	-	-
G_3	124288	6133	1445.96	6.57	28.38	30.47	59.85	30.73	49.23	18.98	30.73	50.78	87.59	34.27
Honolulu														
G_6	586878	32778	6121.95	-	-	-	-	-	-	-	-	-	-	-
G_3	152647	6708	1661.94	11.66	26.01	27.28	66.67	27.37	36.27	37.50	27.37	36.42	89.17	30.55
Helens														
G_6	674552	30971	7085.53	-	-	-	-	-	-	-	-	-	-	-
G_3	180916	11523	2056.91	3.33	26.82	29.19	47.50	28.38	41.69	40.83	28.38	41.88	91.67	31.02

Table 3.5: The result of the methods on the Everest TIN with weights rather than slopes. In Random Everest the weights are assigned randomly. In Flat Everest all the weights are the same. Number of Edges of the Graph ($|E|$), Pre-processing Time (T_p), Average Query Time (QT_{av}), Accuracy (AC), Average Memory Usage (M_{av}), Top 5 percent Average Memory Usage (TM_{av}), Method 1: Refinement Process, Method 2: Enhanced Sleeve, Method 3: Hybrid

		E	T_p	QT_{av}	AC %	Refinement Process			Enhanced Sleeve			Hybrid		
						M_{av} %	QT_{av} %	AC %	M_{av} %	QT_{av} %	AC %	M_{av} %	QT_{av} %	AC %
Random Everest														
G_6	341559		13619.00	3730.51	-	-	-	-	-	-	-	-	-	-
G_3	92372		4684.00	1068.92	7.94	30.35	28.99	41.26	30.35	54.20	52.38	30.35	54.66	88.09
Flat Everest														
G_6	346039		13285.00	3672.22	-	-	-	-	-	-	-	-	-	-
G_3	95384		4793.00	1106.33	0.00	27.56	30.41	29.37	29.79	45.44	28.57	29.79	45.76	81.75

It is possible to apply the refinement algorithm on the path alternatingly from s to t and from t to s . Every time, the path may get improved in terms of length. Now, the question is, how many times the refinement algorithm should be applied alternatingly, from s to t and from t to s , to converge to an specific length. It is not difficult to construct an example that the refinement of vertices of the path could be repeated many times (unbounded in terms of input size) and every time it improves the path length a bit (see the following subsection). However, in a practical setting, we expect that the number of repetitions of the refinement algorithm to be bounded. Therefore, we designed a new experiment. The goal of this experiment is to measure how many times we should repeat the first post-processing method (i.e., the refinement process) to converge and no improvement happens afterward. We used the same 120 queries. In Figure 3.16, the achieved accuracy is plotted per number of times that the refinement process has been applied alternatingly on the query paths. The accuracy is analogously defined as the percentage of queries for which the resulting path is better (i.e., shorter) than in G_6 . The result is, in most of the cases, after applying the refinement process five times alternatingly, from s to t and from t to s , any subsequent improvement is negligible.

Local Refinement is Unbounded

It is possible to apply the refinement algorithm on the path alternatingly from s to t and from t to s . Every time, the length of the path may decrease. Here, we construct an example showing that any local refinement of corners of the path could be repeated many times (unbounded in terms of input size) and every time the length of the path decreases. In Figure 3.17, three triangles are drawn, Δ_1 , Δ_2 and Δ_3 . The input path to the refinement algorithm starts at x_i and ends at x_{i+3} , $\Pi = [s = x_i, x_{i+1}, x_{i+2}, x_{i+3} = t]$ (the red solid polygonal chain in Figure 3.17). The sub-path $[x_{i+1}, x_{i+2}, x_{i+3}]$ disobeys CNR. Therefore, x_{i+2} will be translated to the intersection point of $N(x_{i+1}, e_{i+2})$ (the dashed line in Figure 3.17) and e_{i+2} .

Figure 3.16: The improvement of the refinement post-processing method's accuracy on G_3 , by re-applying alternately from s to t and from t to s , is plotted, for a) Alborz TIN, b) Damavand TIN, c) Everest TIN, d) Grand Canyon TIN, e) Uttarakhand TIN.

Figure 3.17: The constructed example for showing that the local refinement may take arbitrary number of steps to converge.

Then, the first run of the refinement algorithm is over. Now, the second run traverses Π in a reverse order from x_{i+3} to x_i . In the second run, the sub-path $[x_{i+2}, x_{i+1}, x_i]$ disobeys CNR. Therefore, x_{i+1} will be translated to the intersection point of $N(x_{i+2}, e_{i+1})$ (the dotted line in Figure 3.17) and e_{i+1} . Then, in the next run from x_i to x_{i+3} , again the sub-path $[x_{i+1}, x_{i+2}, x_{i+3}]$ disobeys CNR. This pattern could be repeated as it is a function of the geometric properties of the triangles.

This example could be generalized for any local refinement rules that guarantee reduction in the length of the path. We define a local rule to be a rule that takes $k < L$ corners of the input path into account, where L is the size of the path. We discuss when $k = 3$. In order to generalize it for a larger k , it is enough to divide Δ_2 in Figure 3.17 into k smaller triangles, by k edges through the corner of the triangulation v and assign the weight ∞ to all the new triangles.

Suppose $\mathcal{L}$ is a set of local rules that takes 3 corners, $[x_i, x_{i+1}, x_{i+2}]$, of the input path into

account. $\mathcal{L}$ refines the sup-path $[x_i, x_{i+1}, x_{i+2}]$ and makes it closer to the local optimum path from x_i to x_{i+2} (to make it shorter). Since w_2 is an arbitrary large number, $\mathcal{L}$ translates x_{i+1} to be closer to the intersection point of $N(x_{i+2}, e_{i+1})$ and e_{i+1} (because w_2 is very expensive). Then, in the next step, $\mathcal{L}$ refines the sup-path $[x_{i+1}, x_{i+2}, x_{i+3}]$ and makes it closer to the local optimum path from x_{i+1} to x_{i+3} . Since w_2 is ∞ , $\mathcal{L}$ translates x_{i+2} to be closer to the intersection point of $N(x_{i+3}, e_{i+2})$ and e_{i+2} . These two translations could be arbitrary small. They depend on the geometric properties (i.e., the angles and length of edges) of the triangles. Therefore, the number of times that the refinement by $\mathcal{L}$ could improve the path length is unbounded.

Analysis

In the following, we explain the statistical analysis to describe the relation between the properties of a TIN and the accuracy of our proposed algorithm (i.e., the hybrid method). Recall that the weight of a triangle in a TIN is determined by the slope of that face. Therefore, an expensive face is very steep and a cheap face is close to horizontal. The accuracy of the algorithm is a function of the geometry of the TIN and the weights (i.e., slopes) of the faces.

Since we use the Interval Scheme to discretize the triangulation to a graph, the length is not an important factor. If an edge is long, there will be more Steiner points on that edge. However, the adjacency of triangles along an edge with different weights is crucial. Note that a small change of the path in an expensive face could improve the result more than in the other faces. Suppose an expensive triangle is adjacent to a cheap triangle (i.e., they share an edge, like in Figure 3.18). The cheap face in a TIN is attractive for paths to go through, if the length of the path from the source to that face and from that face to the target is not relatively large. If the path goes through the adjacent expensive face too, then the refinement process improves the path significantly by even a very small change in the position of the incident point of the path on the shared edge. Based on this intuition, we

measured the difference between the weights (i.e., slopes) of the adjacent faces in our TINs. We made bins of size 0.05 (~ 2.86 degree) and distributed the measured numbers in these bins. The resulting chart for the 14 TINs in Table 3.1 is shown in Figure 3.19.

Figure 3.18: An expensive face (i.e., steep) is adjacent to an inexpensive face (i.e., horizontal). Some of the Steiner points are shown by crosses. The solid orange line is the path in the graph of Steiner points. The dashed line is the refined path.

These distributions imply that usually the adjacent faces have very close slopes. To model this distribution of thousands of values, we used three known distributions in statistical analysis, namely Exponential, Gamma and 2-parameter Weibull distribution [88]. It is a fundamental problem in statistics to develop a model based on observed data so that further analysis can be carried out with statistical techniques [89]. The reason for choosing these three distributions was the shape implied by the chart (Figure 3.19). The 2-parameter Weibull distribution with the shape parameter (i.e., k) less than 1, was the best fit. The Weibull distribution is one of the used parametric models in the analysis of spatial data. For example, Diggle reported the usage of this model in the analysis of heather-incident data (page 763 [90]). In their study, Weibull was reported to be a good fit for the density function of the distance from an arbitrary point to the nearest point occupied by heather plant. The Weibull distribution has the form shown in Equation 3.1, where λ is the scale parameter and

Figure 3.19: Distribution of the differences between the slope of the adjacent faces in TINs in Table 3.1. Everest and Thompson that our algorithm has the best and the worst accuracy respectively, are shown by solid lines.

k is the shape parameter.

$$f(x; \lambda, k) = \frac{k}{\lambda} \left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k}, \quad x \geq 0 \quad (3.1)$$

The result of the fit is in Table 3.6. In this table, the descriptive measures (i.e., expected value, standard deviation and median) of the Weibull distributions are reported. Also, the result of the Kolmogorov-Smirnov goodness-of-fit test [88] is provided. It shows more than 90 % fit in all of the TINs. The used method for fitting Weibull was the Least Squares Method [88].

To find a relationship between the measures of distributions and the accuracy of our

Table 3.6: The result of fitting 2-parameter Weibull distribution that shape parameter k is less than 1. Also the result of Kolmogorov-Smirnov goodness-of-fit test (K-S) is reported.

TIN	λ	k	Expected Value	Std. Dev.	Median	K-S test
Alborz	0.0277	0.4736	0.0616	0.1488	0.0128	0.09316
Damavand	0.0944	0.8471	0.1029	0.1220	0.0612	0.01539
Everest	0.1492	0.7715	0.1736	0.2277	0.0928	0.01903
Grand Canyon	0.1625	0.7618	0.1911	0.2541	0.1004	0.02389
Uttarkhand	0.1139	0.8696	0.1222	0.1410	0.0747	0.02077
val-des-monts	0.0398	0.7410	0.0479	0.0657	0.0243	0.03926
Token	0.0982	0.8395	0.1077	0.1289	0.0635	0.02602
Matterhorn	0.1130	0.6531	0.1535	0.2435	0.0645	0.06907
Heart Peaks	0.0455	0.6598	0.0611	0.0958	0.0261	0.05403
Tseax Cone	0.0410	0.5834	0.0640	0.1165	0.0219	0.09701
K2	0.1299	0.6545	0.1761	0.2785	0.0742	0.05948
Thompson	0.0244	0.4558	0.0587	0.1501	0.0109	0.09981
Honolulu	0.1198	0.8201	0.1335	0.1638	0.0766	0.02386
Helen	0.1420	0.7219	0.1747	0.2467	0.0855	0.04236

algorithm we used Pearson's correlation coefficient and Spearman's rank correlation. Spearman's correlation is a nonparametric technique for evaluating the degree of linear association between two independent variables. It is similar to Pearson's correlation except that it operates on the ranks of data. Since it is nonparametric, it is unaffected by the distribution of the population. Also, it is relatively insensitive to outliers because it acts on ranks of data. It can be used with very small sample size [91]. If we knew that the data were normally distributed, Pearson's correlation is more suitable for analysis. However, this is not the case for our data. The correlation coefficients are shown in Table 3.7. This result shows that the parameters of the Weibull distribution and the accuracy (achieved by the Hybrid method) are highly correlated. The scatter plot for the measures of the distributions versus the accuracy of our algorithm is shown in Figure 3.20.

Figure 3.20: The scatter plot of the a) scale parameter, b) shape parameter, c) expected value, d) standard deviation, and, e) median of Weibull distribution versus the accuracy of the algorithm.

Table 3.7: Correlation between the measures of the distributions of the TINs and the accuracy of our algorithm on the TINs.

	Pearson's	Spearman's
Scale parameter and accuracy	0.81637	0.747253
Shape parameter and accuracy	0.629806	0.496703
Expected Value and accuracy	0.777033	0.747253
Std. Dev. and accuracy	0.580279	0.514286
Median and accuracy	0.815751	0.703297

3.4.4 Conclusions of Experiments

In conclusion, based on these experiments, discretizing the geometric space by on average three Steiner points per edge, i.e., producing G_3 , followed by the Hybrid post-processing method, brought a good trade-off between resource usage (i.e., query time and memory) and accuracy. Among all experiments on the TINs, the best accuracy was obtained for Everest TIN. On this TIN, using G_3 followed by the Hybrid post-processing method, results in a 43 % saving in query time and 69 % saving in memory usage while keeping on accuracy close to 97 %. In terms of accuracy, the worst result on the TINs was obtained for Thompson TIN. The same method on Thompson results in a 38 % saving in query time and 68 % saving in memory usage while keeping on accuracy close to 84 %.

To understand the behavior of the algorithm on different TINs, we analyzed the distribution of the differences between the slope of the adjacent faces in TINs. We first, fitted a parametric model (i.e., 2-parameter Weibull distribution with shape parameter less than 1). Then, we measured the correlation between the descriptive measures of the distribution of TINs and the accuracy of the proposed algorithm on TINs. We utilized two parametric and non-parametric techniques (i.e., Pearson's and Spearman's) to evaluate the degree of linear association between them. The result of the analysis showed that there is a strong positive correlation between the descriptive measures of the distribution of TINs (i.e., scale parameter, shape parameter, expected value, standard deviation, and median) and the accuracy of

the algorithm.

Another lesson from this experiment is that the performance of the proposed algorithm decreases, when the weight assignment pushes the shortest path to travel on the edges and through the vertices of a TIN. This case happens when the weights are assigned randomly to faces because two adjacent faces could get very different weights, whereas in "real" TINs slopes tend to change more gently. That is the main reason for the accuracy of the algorithm on Random Everest (see Table 3.5).

3.5 Conclusions

In this chapter, we studied graph-based approximation algorithms for the geometric shortest path problem. In the existing literature, the quality of the output of these algorithms are measured in terms of the length of the output path. Here, we proposed some qualitative criteria, CAR and CNR. We proved that these criteria could also make paths shorter. In particular, if the input path is an ε -approximation of a shortest path, the output is a refined ε -approximation. We also proposed a linear time algorithm to refine any linear input path. The results of our experiments on TINs suggest that the proposed algorithms should perform well in practice. The savings as measured are, on average, 51% in query time and 69% in memory usage, while maintaining an accuracy that is close-to-optimal.

Chapter 4

Similarity of Polygonal Curves in the Presence of Outliers

4.1 Introduction

As we mentioned in Section 1.1.2, the Fréchet distance - one of the widely used measures for similarity between curves - is intuitive and takes into account global features of the curves [22]. Despite being a high quality similarity measure for polygonal curves, it is very sensitive to the presence of outliers. Consequently, researches have been carried out to formalize the notion of similarity among a set of polygonal curves that tolerate outliers. They are based on intersection of curves in local neighborhood [118], topological features [119], or adding flexibility to incorporate the existence of outliers [25]. In [25], Driemel and Har-Peled discuss a new notion of robust Fréchet distance, where they allow k shortcuts between vertices of one of the two curves, where k is a constant specified as an input parameter. They provide a constant factor approximation algorithm for finding the minimum Fréchet distance among all possible k -shortcuts. One drawback of their approach is that a shortcut is selected without considering the length of the ignored part. Consequently, such shortcuts may remove a significant portion of a curve. As a result, substantial information about the similarity of the original curves could be ignored. A second drawback of their approach is that the shortcuts are only allowed to one of the curves. Since noise could be present in both curves, shortcuts may be required on both to achieve a good result. For example, Figure 4.1a shows two polygonal curves that both need simultaneously shortcuts to become similar.

Here we discuss an alternative Fréchet distance measure to tolerate outliers; it incorporates the length of the curves and allows the possibility of shortcuts on one or both curves.

We consider two natural dual perspectives of this problem. They are outlined as follows using the common dog-leash metaphor for Fréchet distance.

Assume that a person wants to walk along one curve and his/her dog on another one. Let $\varepsilon \geq 0$ be a given constant. The **Min-Exclusion (MinEx) Problem** is to determine a walk that minimizes the total length of all parts of the curves for which a leash length bigger than ε is needed. The **Max-Inclusion (MaxIn) Problem** is to determine a walk that maximizes the total length of all parts of the curves for which a leash length at most ε is sufficient.

Observe that the solution for the **MinEx** problem leads to a solution for the **MaxIn** problem and vice versa. An exact solution for these problems is presented in [120], where the distances are measured using (more restrictive and much simpler) the L_1 and L_∞ metrics. In particular, they solved exactly the **MinEx** (respectively the **MaxIn**) Problem under L_1 or L_∞ metric in $\mathcal{O}(n_1 \cdot n_2 \cdot (n_1 + n_2) \cdot \log(n_1 \cdot n_2))$ time, while consuming $\mathcal{O}(n_1 \cdot n_2 \cdot (n_1 + n_2))$ space, where n_1 denotes the complexity of the input polygonal curve T_1 and n_2 denotes the complexity of the input polygonal curve T_2 . In [49], using Galois theory, we show that these problems are not solvable by radicals over $\mathbb{Q}$, when distances are measured using the L_2 -metric. (It is natural to study Fréchet distance problems in L_2 -metric, see e.g. [22].) This suggests that we should look for approximation algorithms. Har-Peled and Wang [124] proposed an approximation algorithm for the **MaxIn** problem whose running time is $\mathcal{O}\left(\frac{n^4}{\delta^2}\right)$, where n is the size of the input polygonal curves. They claimed that their approach is a $(1 - \delta)$ -approximation, that is, the quality of their solution is greater than $(1 - \delta)$ times the optimal one ([124], Theorem 4.2). However, we show via a counterexample that this claim is not true (see Section 4.1.3). Therefore, to the best of our knowledge, no FPTAS (Fully Polynomial-Time Approximation Scheme) exists for this problem. Here, we provide algorithms that approximate solutions for the **MinEx** and **MaxIn** problems up to an additive approximation error δ times the length of the input curves.

Figure 4.1: a) A possible solution is illustrated by the connecting lines between the parameterizations for T_1 and T_2 . The sub-curves on both polygonal curves that should be ignored are illustrated by the blue, red and green sub-curves on T_1 and T_2 . So, $Q^B(T_1, T_2)$ is the summation of the lengths of the colored sub-curves and $Q^W(T_1, T_2)$ is that of the black sub-curves. b) The solution corresponds to an xy -increasing path in the deformed free-space diagram F . In this space, $Q^B(T_1, T_2)$ can be measured by summing the lengths of its subpaths going through the forbidden space (shaded gray area), measured in the L_1 -metric (similarly for $Q^W(T_1, T_2)$)—see Subsection 4.1.2 for definitions of $Q^B(T_1, T_2)$ and $Q^W(T_1, T_2)$.

4.1.1 Preliminaries

Let $T_1 : [0, n_1] \rightarrow \mathbb{R}^2$ and $T_2 : [0, n_2] \rightarrow \mathbb{R}^2$ be two polygonal curves. We have defined the Fréchet distance $\delta_F(T_1, T_2)$ in Section 1.1.2, Equation 1.7. For simplicity, we say that in this chapter all considered parameterizations, $\alpha_1 : [0, 1] \rightarrow [0, n_1]$ and $\alpha_2 : [0, 1] \rightarrow [0, n_2]$, are monotone (only forward movements are allowed). We refer to a pair of parameterizations (α_1, α_2) for T_1 and T_2 w.r.t. ε , as a possible *solution* for T_1 and T_2 .

A useful structure to decide whether the Fréchet distance between two polygonal curves is upper-bounded by a given ε , is the free-space diagram [22]. For two polygonal curves, T_1 and T_2 , and two corresponding parameterizations, α_1 and α_2 , the *free-space* is defined formally by (4.1), where $L_2(\cdot, \cdot)$ is the Euclidean distance.

$$W = \{(t_1, t_2) \in [0, 1] \times [0, 1] \mid L_2(T_1(\alpha_1(t_1)), T_2(\alpha_2(t_2))) \leq \varepsilon\} \quad (4.1)$$

The *free-space diagram* is the rectangle $[0, 1] \times [0, 1]$, partitioned into n_1 columns and

n_2 rows. It consists of $n_1 n_2$ parameter cells $C^{i,j}$, for $i = 1, \dots, n_1$ and $j = 1, \dots, n_2$, whose interiors do not intersect with each other. The cell $C^{i,j}$ represents the multiplication of two subranges of $[0, 1]$ that are mapped to the edge between vertices $T_1(i-1)$ and $T_1(i)$ and the edge between vertices $T_2(j-1)$ and $T_2(j)$. For each parameter cell $C^{i,j}$, there exists an ellipse such that the intersection of the area bounded by this ellipse with $C^{i,j}$ is equal to the free-space region of that cell. We denote the free-space of $C^{i,j}$ by $C_W^{i,j}$ and its complement, the forbidden space, by $C_B^{i,j}$. The boundary of this ellipse and the boundary of the cell, $C^{i,j}$, intersect at most eight times (i.e., at most two intersections per side of $C^{i,j}$). These intersection points form at most four intervals on the boundary of $C^{i,j}$. These intervals could be empty or contains only one point. In addition, two adjacent cells have the same interval on the shared side between the cells. The union of all cells' free-space builds the *free-space* (or *white-space*) of the diagram and is denoted by W . The complement of W is the *forbidden-space* (or *black-space*) of the diagram and is denoted by B .

In this thesis, we stretch and compress the columns and rows of the free-space diagram, such that their widths and heights are equal to the lengths of the corresponding segments of T_1 and T_2 , respectively. The resulting diagram is called the *deformed free-space diagram* and is denoted by F . Since we consider the worst case running time, we assume $n := \max\{n_1, n_2\} = n_1 = n_2$. We consider the free-space diagram in the Cartesian plane and assume that it lies axes aligned. We say that a point lies to the left (resp. right) of another point, if its x -coordinate is smaller (resp. greater) than the second one. Analogously, we say, that a point lies below (resp. above) another point, if its y -coordinate is smaller (resp. greater) than the second one. We define a point s *dominates* another point s' , if and only if s' does not lie to the right or above s .

For simplification, we say that paths, curves, edges, etc. are xy -increasing if they are nondecreasing in both x - and y -coordinates. We know that any pair of monotone parameterizations (α_1, α_2) corresponds to an xy -increasing path in free-space diagram, π_{st} , connecting

the bottom left corner of the diagram, $\mathfrak{s}$, to the upper right corner, $\mathfrak{t}$. So, deciding if the Fréchet distance of T_1 and T_2 is upper bounded by ε , is equivalent to the decision: Does there exist an xy -increasing path in the free-space diagram for ε , connecting $\mathfrak{s}$ to $\mathfrak{t}$ and avoiding the forbidden space [22]?

4.1.2 Problem Definition

Let T_1 and T_2 be two polygonal curves, each consisting of at most n line segments; and $\varepsilon \geq 0$ be a constant. For the **MinEx** (respectively, **MaxIn**) problem, the *quality* of a solution is the sum of the lengths of the sub-curves of T_1 and T_2 that lie in the forbidden (respectively, free) space (see Figure 4.1 for an example). Formally, for a given pair of parameterizations (α_1, α_2) , let $\mathcal{B}_{\alpha_1\alpha_2} \subseteq [0, 1]$ be the closure of the set of times, t , such that $\|\alpha_1(t) - \alpha_2(t)\| > \varepsilon$, and $\mathcal{W}_{\alpha_1\alpha_2} \subseteq [0, 1]$ be the closure of the set of times, t , such that $\|\alpha_1(t) - \alpha_2(t)\| \leq \varepsilon$. We define the quality of a solution (α_1, α_2) for the **MinEx** problem $Q_{\alpha_1\alpha_2}^B$ and for the **MaxIn** problem $Q_{\alpha_1\alpha_2}^W$ as follows (see Figure 4.1a):

$$\begin{aligned} Q_{\alpha_1\alpha_2}^B &:= \int_{t \in \mathcal{B}_{\alpha_1\alpha_2}} \|(T_1(\alpha_1(t)))'\| dt + \int_{t \in \mathcal{B}_{\alpha_1\alpha_2}} \|(T_2(\alpha_2(t)))'\| dt \\ Q_{\alpha_1\alpha_2}^W &:= \int_{t \in \mathcal{W}_{\alpha_1\alpha_2}} \|(T_1(\alpha_1(t)))'\| dt + \int_{t \in \mathcal{W}_{\alpha_1\alpha_2}} \|(T_2(\alpha_2(t)))'\| dt \end{aligned} \quad (4.2)$$

where $\|v\|$ is L_2 norm of a vector v . We call (α_1, α_2) *optimal* if it gives the infimum of $Q_{\alpha_1\alpha_2}^B$ or supremum of $Q_{\alpha_1\alpha_2}^W$ and define the *quality of T_1 and T_2 w.r.t. ε* as its value, i.e.,

$$\begin{aligned} Q^B(T_1, T_2) &:= \inf_{\alpha_1, \alpha_2: [0,1] \rightarrow [0,1]} Q_{\alpha_1\alpha_2}^B \\ Q^W(T_1, T_2) &:= \sup_{\alpha_1, \alpha_2: [0,1] \rightarrow [0,1]} Q_{\alpha_1\alpha_2}^W \end{aligned} \quad (4.3)$$

This means that the quality of T_1 and T_2 is the infimum (respectively, supremum) sum of the lengths of curves on T_1 and T_2 to be ignored (respectively, matched), to obtain a Fréchet distance not greater than ε . For a given ε , we would like to find a solution whose quality is

not much worse than an optimal solution. To do so, we transform this problem setting into a weighted xy -increasing path problem, between $\mathfrak{s}$ and $\mathfrak{t}$, in the free-space diagram of T_1 and T_2 .

To solve the **MinEx** problem, we look for an xy -increasing path $\pi_{\mathfrak{s}\mathfrak{t}} \subset F$ from $\mathfrak{s}$ to $\mathfrak{t}$, where our goal is to minimize the length of $\pi_{\mathfrak{s}\mathfrak{t}}$, lying in the forbidden space of F . Also, to solve the **MaxIn** problem, we look for an xy -increasing path $\pi_{\mathfrak{s}\mathfrak{t}} \subset F$ from $\mathfrak{s}$ to $\mathfrak{t}$, where our goal is to maximize the length of $\pi_{\mathfrak{s}\mathfrak{t}}$, lying in the free space of F . We have the following observation.

Observation 6. *Let T_1 and T_2 be two arbitrary polygonal curves in $\mathbb{R}^2$ and let F be the corresponding deformed free-space diagram for a leash length of ε . Let $\pi_{\mathfrak{s}\mathfrak{t}} \subset F$ be a path corresponding to a pair of parameterizations (α_1, α_2) of T_1 and T_2 w.r.t. ε . Then, the sum of the lengths of the forbidden (respectively, free) paths of $\pi_{\mathfrak{s}\mathfrak{t}}$, measured under L_1 -metric, is equal to $Q_{\alpha_1\alpha_2}^B$ (respectively, $Q_{\alpha_1\alpha_2}^W$).*

Now it is easy to see that the **MinEx** and **MaxIn** problems are transformed to the following path problems.

Weighted shortest xy -increasing path (wShortMP) problem: Compute an xy -increasing weighted shortest path from $\mathfrak{s}$ to $\mathfrak{t}$ in F , where the weight in the forbidden-space is one and the weight in the free-space is zero. The length of a path is defined as the sum of the lengths (measured in L_1 -metric) of the part of the path lying in the forbidden space.

Weighted longest xy -increasing path (wLongMP) problem: Compute an xy -increasing weighted longest path from $\mathfrak{s}$ to $\mathfrak{t}$ in F , where the weight in the forbidden-space is zero and the weight in the free-space is one. The length of a path is defined as the sum of the lengths (measured in L_1 -metric) of the part of the path lying in the free space.

4.1.3 Counterexample

In this section, before continuing to the discussion about algorithms, we show that the length of the boundary of the free-space of an arbitrary parameter cell is not bounded by

Figure 4.2: Counterexample to $\omega = O(Q^W(T_1, T_2))$. a) Two trajectories T_1 and T_2 lie parallel to each other. They have opposite directions. b) Free-space of T_1 and T_2 .

$O(Q^W(T_1, T_2))$ in general. Consider the following example where each trajectory T_1 and T_2 consists of a single line segment (Figure 4.2(a)). The trajectories T_1 and T_2 have the same length, they are parallel to each other at distance ε , and they have opposite directions. Consider the free-space diagram F of T_1 and T_2 with respect to Fréchet distance ε . The free-space W of F corresponds to the diagonal connecting the upper left corner of F to its bottom right corner. Suppose that ω is the length of the boundary of W . By moving T_1 towards T_2 , W becomes 2-dimensional. This free-space W can be chosen arbitrarily thin. Therefore, we can make the part of the path that lies in W arbitrarily short (Figure 4.2(b)). Therefore, the ratio $\frac{\omega}{Q^W(T_1, T_2)}$ can be made arbitrarily big. This is a contradiction to $\omega = O(Q^W(T_1, T_2))$, which was an assumption in Theorem 4.2 in [124].

4.1.4 New Results

In [49], we establish that the **MinEx** and **MaxIn** problems are not solvable exactly by radicals over $\mathbb{Q}$. This is proved using Observation 6 and showing that the **wShortMP** problem is unsolvable within the *Algebraic Computation Model over the Rational Numbers* (**ACM $\mathbb{Q}$**). In this model we can compute exactly any number that can be obtained from the rationals $\mathbb{Q}$ by applying a finite number of operations from $+, -, \times, \div, \sqrt[k]{}$, for any integer $k \geq 2$. The proof is based on Galois theory. Motivated by that, we turn our attention towards approximation algorithms for the **MinEx** and the **MaxIn** problems.

In Section 4.2, we transform the **MinEx** problem to the **wShortMP** problem, which in turn is transformed to a shortest path problem in directed acyclic graphs (Lemma 17). We propose an algorithm that approximates the weighted xy -increasing shortest path up to an additive error. This error is related to the lengths of the curves T_1 and T_2 . The running time of this algorithm is $\mathcal{O}\left(\frac{n^4}{\delta^2}\right)$, where δ is the approximation parameter (Theorem 6). This algorithm also provides an approximate solution for the **MaxIn** problem and with the same approximation quality (Corollary 5).

In Section 4.3, we improve the running time of this algorithm to $\mathcal{O}\left(\frac{n^3}{\delta} \log\left(\frac{n}{\delta}\right)\right)$ (Theorem 7). To do so, we solve a subproblem related to forming a ‘small’ graph over a convex set of points that preserves L_1 -distances between certain pairs of points. In Section 4.3.4, we discuss why a FPTAS for the **MinEx** and **MaxIn** problems may not be feasible. However, for the **MaxIn** problem, we are able to design a $(1 - \delta)$ -approximation algorithm running in polynomial time, and its complexity depends on the size of the input n , approximation factor δ , and an additional parameter γ defined as follows. Consider the **MaxIn** problem in the setting where distances are measured in the L_1 -metric, i.e., the distance between a pair of points, one on trajectory T_1 and other on trajectory T_2 , is measured using the L_1 -metric. It turns out that the free space within a cell for a given leash length is still convex, but its boundary is composed of straight line segments, instead of that of ellipses. We define γ to be the length of the optimal solution for **MaxIn** problem in L_1 -metric. Furthermore, Buchin et al. [120] have shown that γ can be computed in polynomial time.

4.2 An Approximation Algorithm

In this section, we present an approximation algorithm with an additive error for the **MinEx** problem and we show that the computed approximate solution is an approximate solution for the **MaxIn** problem as well. This algorithm provides the basis for our algorithms developed in Section 4.3. The algorithms of Section 4.3 have improved running times, while maintaining

the accuracy we obtain in this section.

The input to the problem consists of two polygonal curves T_1 and T_2 , an arbitrary fixed leash length of $\varepsilon \geq 0$ and an approximation parameter $\delta > 0$. We want to compute a pair of parameterizations $(\tilde{\alpha}_1, \tilde{\alpha}_2)$ and its quality $Q_{\tilde{\alpha}_1 \tilde{\alpha}_2}^B$ for T_1 and T_2 , such that $Q_{\tilde{\alpha}_1 \tilde{\alpha}_2}^B$ is a “good” approximation of $Q^B(T_1, T_2)$. We also want to construct two polygonal curves, T'_1 and T'_2 such that $\delta_F(T'_1, T'_2) \leq \varepsilon$, that correspond to a solution for the **MinEx** problem. We denote the deformed free-space diagram by F , its free-space by W and its forbidden-space by B . Recall that F consists of $O(n^2)$ cells and each cell is a rectangle, whose free space is (a portion of) an ellipse. Our approach is as follows.

We saw in Section 4.1.2 how to transform the **MinEx** problem into the **wShortMP** problem in the deformed free space diagram F . To design an approximation algorithm for the **wShortMP** problem, we will define a directed acyclic graph G over F and show that for each xy -increasing path π_{st} in F , there exists a path $\tilde{\pi}_{st}$ in G which stays close to π_{st} . Thus, the paths in G approximate xy -increasing weighted shortest paths in F . Once we have a shortest path in G , we will explain how to embed this path in F in order to deduce an approximate solution for the **MinEx** problem. Though we are casting a geometric problem into a combinatorial setting, to simplify our notation, we may refer to a point $p \in F$ as a vertex p in G - the meaning will be clear from the context. Our approximation algorithm for the **wShortMP** problem is as follows.

Step 1: Construct F . First compute the free-space diagram using the algorithm of Alt and Godau [22]. In [22], the computation is made only on the free-space intervals, which lie on the boundaries of the cells. However, in each cell, it is possible to determine the equation of the ellipse in constant time (see, for example [21]). To obtain the deformed free-space diagram of F , stretch the columns and rows of the diagram, such that their sizes are respectively equal to the lengths of the corresponding segments (see Figure 4.3).

Step 2: Construct Grid. Insert $\frac{n}{\delta}$ equidistant *vertical grid lines* and $\frac{n}{\delta}$ *horizontal grid*

Figure 4.3: Two polygonal curves and the corresponding deformed free space diagram F for a given ε .

Figure 4.4: Illustration of Step 2 of insertions of grid and intersection lines in F of Figure 4.3.

lines in F (see Figure 4.4(a)). Find the intersection of every vertical grid line ℓ_{v_i} with the boundary of each ellipse. For each of these intersection points, add a new horizontal intersection line, passing through that point (see Figure 4.4(b)). Perform analogous steps for every horizontal grid line ℓ_{h_i} . Note that the lines defining the boundary of each parameter cell in F are also considered to be grid lines.

Step 3: Construct G . Compute the arrangement A induced by all of the grid lines, the intersection lines and the boundaries of ellipses. The vertices of the directed graph G are the vertices in A . There is a directed edge from a vertex $p \in G$ to a vertex $q \in G$ whenever $\vec{pq}$ is an xy -increasing (i.e., non-decreasing in both x and y) edge in A . The weight of this edge is equal to its length with respect to the L_1 -metric if it is lying in the forbidden space, otherwise it is zero.

Step 4: Compute Path. Compute a weighted shortest path $\tilde{\pi}_{st}$ from s to t in G , and output its corresponding geometric embedding as the desired approximate solution.

Observe that G is acyclic as all of its edges are directed and are xy -increasing. In the next three lemmas, we establish that G can be used to provide an approximation algorithm for the wShortMP problem. First we need the following definition. Recall that a point $s' = (x', y')$ is dominated by $s = (x, y)$, if $x' \leq x$ and $y' \leq y$. A vertex $s' \in G$ is *directly dominated* by a point $s \in F$ if and only if s' is dominated by s and there exists no vertex of G in the interior of the line segment ss' .

Lemma 15. *Let $s, t \in W$ be two points inside a parameter cell. Assume that $s \in \ell_s$ and $t \in \ell_t$, where ℓ_s and ℓ_t are grid lines. Assume that s is dominated by t . Let $s' \in \ell_s$ be a vertex of G that is directly dominated by s and $t' \in \ell_t$ be a vertex of G that is directly dominated by t . Furthermore, assume that $s', t' \in W$. Then, there exists a path $\tilde{\pi}_{s't'}$ in G from s' to t' , such that $\tilde{\pi}_{s't'} \subset W$.*

Proof. Let $\mathcal{E}$ be the ellipse containing s and t in this parameter cell. Notice that if $s \in \partial\mathcal{E}$, then $s' = s$. Four different cases, which we discuss below, can arise.

a) The grid lines ℓ_s and ℓ_t are vertical (Figure 4.5(a)). Let $\ell_{s'}$ be the horizontal line containing s' . Notice that $\ell_{s'}$ is either a grid line or an intersection line. Let a be the intersection point of $\ell_{s'}$ with ℓ_t . We walk on $\ell_{s'}$, until either we leave $\mathcal{E}$ at a point $x \in \partial\mathcal{E}$ or we reach a . Suppose we encounter a first. Since t' is directly dominated by t , a is not above t' . Since $\mathcal{E}$ is convex, the line segment $at \in \mathcal{E}$. Therefore, we walk on ℓ_t until we reach t' .

Otherwise, $a \notin \mathcal{E}$ and we need to travel from x to t' (Figure 4.5(a)). Let a' be the intersection point of $\partial\mathcal{E}$ with the line segment at . Since t' is directly dominated by t , a' is not above t' . Since $\mathcal{E}$ is convex, the line segment $a't \in \mathcal{E}$. Moreover, the part of $\partial\mathcal{E}$ from x to a' is xy -increasing, because $t \in \mathcal{E}$. Therefore, we travel on $\partial\mathcal{E}$ from x to a' . Then, we walk on ℓ_t until we reach t' .

- b) The grid lines ℓ_s and ℓ_t are horizontal (Figure 4.5(b)). Use the same argument as in a), but first walk upwards.
- c) The grid line ℓ_s is vertical and the grid line ℓ_t is horizontal (Figure 4.5(c)). Use the same argument as in a), but first move to s and then walk upwards.
- d) The grid line ℓ_s is horizontal and the grid line ℓ_t is vertical (Figure 4.5(d)). Use the same argument as in a), but first move to s and then walk to the right.

□

For the next lemma, we need the following notations. Each parameter cell $C^{i,j}$ of F defines a rectangle, having left side $C_L^{i,j}$, right side $C_R^{i,j}$, bottom side $C_B^{i,j}$ and top side $C_T^{i,j}$. Furthermore, let $\|\pi\|$ be the weighted length of a path π in F , with respect to the L_1 -metric. Let T be a polygonal curve, with vertices $\langle t_0, t_1, \dots, t_{k-1}, t_k \rangle$, composed of the k line segments $t_0t_1, t_1t_2, \dots, t_{k-1}t_k$. We denote the length of T by $|T| = \sum_{i=1}^k |t_{i-1}t_i|$.

on the intersection of two grid lines, let ℓ_{v_a} be any of them. Let $v'_a \in \ell_{v_a}$ be a vertex directly dominated by v_a (if v_a is on the boundary of $\mathcal{E}$, we take $v'_a = v_a$). Analogously, we define v_b as the last intersection point of π_{st} with a grid line ℓ_{v_b} , lying before b . Let $v'_b \in \ell_{v_b}$ be a vertex that is directly dominated by v_b . Let $\tilde{\pi}_{v'_a v'_b} \subset G$ be the curve obtained by applying Lemma 15 to v_a, v_b, v'_a and v'_b . Recall that $\tilde{\pi}_{v'_a v'_b}$ is in W .

Let C_a be the grid cell containing a . Consider the part of π_{st} that lies inside C_a . Let w_a be the entry point of π_{st} into C_a and let w'_a be the left bottom corner vertex of C_a . Since $w'_a, v'_a \in C_a$, there exists an xy -increasing path $\tilde{\pi}_{w'_a v'_a}$ from w'_a to v'_a , such that

$$\|\tilde{\pi}_{w'_a v'_a}\| \leq 2 \frac{\delta}{n} \cdot \max\{|T_1|, |T_2|\}. \quad (4.4)$$

Consider the part of π_{st} from s to w_a . Since s and s' lie on the same side of a single grid cell, we have

$$\|ss'\| \leq \frac{\delta}{n} \cdot \max\{|T_1|, |T_2|\}. \quad (4.5)$$

Furthermore, w'_a is dominated by w_a and the weighted shortest xy -increasing path π_{sw_a} from s to w_a lies completely inside B . Since w_a and w'_a lie on the same side of a single grid cell, we have

$$\|w_a w'_a\| \leq \frac{\delta}{n} \cdot \max\{|T_1|, |T_2|\}. \quad (4.6)$$

Let $\tilde{\pi}_{s' w'_a} \subset G$ be an xy -increasing path from s' to w'_a . We have

$$\begin{aligned} \|\tilde{\pi}_{s' w'_a}\| &\leq \|s's\| + \|\pi_{sw_a}\| + \|w_a w'_a\| \\ &\leq \frac{\delta}{n} \cdot \max\{|T_1|, |T_2|\} + \|\pi_{sw_a}\| + \frac{\delta}{n} \cdot \max\{|T_1|, |T_2|\} \quad \text{by (4.5) and (4.6),} \\ &= \|\pi_{sw_a}\| + 2 \frac{\delta}{n} \cdot \max\{|T_1|, |T_2|\}. \end{aligned} \quad (4.7)$$

For the remaining part of π_{st} (from v_b to t), using a similar argument, we can construct

an xy -increasing path $\widetilde{\pi}_{v'_b w'_b}$ from v'_b to w'_b and an xy -increasing path $\widetilde{\pi}_{w'_b t'}$ from w'_b to t' such that

$$\|\widetilde{\pi}_{v'_b w'_b}\| \leq 2\frac{\delta}{n} \cdot \max\{|T_1|, |T_2|\}. \quad (4.8)$$

$$\|\widetilde{\pi}_{w'_b t'}\| \leq \|\pi_{w_b t}\| + 2\frac{\delta}{n} \cdot \max\{|T_1|, |T_2|\}. \quad (4.9)$$

By concatenating $\widetilde{\pi}_{s' w'_a}$, $\widetilde{\pi}_{w'_a v'_a}$, $\widetilde{\pi}_{v'_a v'_b}$, $\widetilde{\pi}_{v'_b w'_b}$ and $\widetilde{\pi}_{w'_b t'}$, we get an xy -increasing path $\widetilde{\pi}_{s' t'}$ from s' to t' in G such that

$$\|\widetilde{\pi}_{s' t'}\| \leq \|\pi_{st}\| + 8\frac{\delta}{n} \cdot \max\{|T_1|, |T_2|\}.$$

□

Lemma 17. *The graph G has a complexity of $O\left(\frac{n^4}{\delta^2}\right)$. If $\pi_{\mathfrak{s}\mathfrak{t}}$ (respectively, $\widetilde{\pi}_{\mathfrak{s}\mathfrak{t}}$) is a weighted shortest xy -increasing path in F (respectively, G) then $\|\pi_{\mathfrak{s}\mathfrak{t}}\| \leq \|\widetilde{\pi}_{\mathfrak{s}\mathfrak{t}}\| \leq \|\pi_{\mathfrak{s}\mathfrak{t}}\| + \delta \cdot \max\{|T_1|, |T_2|\}$.*

Proof. Denote the sequence of parameter cells that $\pi_{\mathfrak{s}\mathfrak{t}}$ intersects from $\mathfrak{s}$ to $\mathfrak{t}$ by $C_1, \dots, C_k$. Let $\widetilde{\pi}_{\mathfrak{s}\mathfrak{t}}$ be the concatenation of the paths obtained by applying Lemma 16 with $\delta := \frac{\delta}{16}$ on each C_i . The error made in each cell is therefore upper bounded by $\frac{\delta}{2n} \cdot \max\{|T_1|, |T_2|\}$. Since $\pi_{\mathfrak{s}\mathfrak{t}}$ passes through at most $2n$ parameter cells, it follows that $\|\pi_{\mathfrak{s}\mathfrak{t}}\| \leq \|\widetilde{\pi}_{\mathfrak{s}\mathfrak{t}}\| \leq \|\pi_{\mathfrak{s}\mathfrak{t}}\| + \delta \cdot \max\{|T_1|, |T_2|\}$.

Recall that there are at most $n + \frac{16n}{\delta}$ horizontal grid lines and at most as many vertical grid lines. Each of these grid lines can intersect at most $2n$ ellipses, thus requiring the addition of at most $4n\left(2n + \frac{32n}{\delta}\right) = O\left(\frac{n^2}{\delta}\right)$ additional intersection lines. Hence, the arrangement of all these lines has a complexity of $O\left(\frac{n^4}{\delta^2}\right)$. □

The above lemma shows that for the **wShortMP** problem, $\widetilde{\pi}_{\mathfrak{s}\mathfrak{t}} \subset G$ approximates $\pi_{\mathfrak{s}\mathfrak{t}} \subset F$. We now explain how to derive an approximate solution for the **MinEx** problem, given $\widetilde{\pi}_{\mathfrak{s}\mathfrak{t}}$. The path $\widetilde{\pi}_{\mathfrak{s}\mathfrak{t}}$ passes through a sequence of parameter cells. For each edge in $\widetilde{\pi}_{\mathfrak{s}\mathfrak{t}}$, find its embedding in F . Since each of these embedded edges are xy -increasing and the end vertex

of one edge is the start vertex of the next one, this results in an xy -increasing path from $\mathfrak{s}$ to $\mathfrak{t}$ in F . Let $\pi_{\mathfrak{s}\mathfrak{t}}$ be the weighted shortest xy -increasing path connecting $\mathfrak{s}$ to $\mathfrak{t}$. From Lemma 17 it follows that $\|\pi_{\mathfrak{s}\mathfrak{t}}\| \leq \|\tilde{\pi}_{\mathfrak{s}\mathfrak{t}}\| \leq \|\pi_{\mathfrak{s}\mathfrak{t}}\| + \delta \cdot \max\{|T_1|, |T_2|\}$. Since $\tilde{\pi}_{\mathfrak{s}\mathfrak{t}}$ is a concatenation of segments, it can be directly transformed into two corresponding parameterizations $\tilde{\alpha}_1$ and $\tilde{\alpha}_2$. Since $\|\tilde{\pi}_{\mathfrak{s}\mathfrak{t}}\| \leq \|\pi_{\mathfrak{s}\mathfrak{t}}\| + \delta \cdot \max\{|T_1|, |T_2|\}$, the accuracy of the solution $(\tilde{\alpha}_1, \tilde{\alpha}_2)$ follows from Observation 6. Let $\tilde{\pi}_{ab}$ be a maximal subpath of $\tilde{\pi}_{\mathfrak{s}\mathfrak{t}}$, passing through the forbidden-space and connecting the points a and b , both of which lie on the boundary of the free-space. These points correspond to two matchings (a_1, a_2) and (b_1, b_2) , where $a_1, b_1 \in T_1$ and $a_2, b_2 \in T_2$. Since both lie on the boundary of the free-space, it follows that $|a_1 a_2| = \varepsilon$ and $|b_1 b_2| = \varepsilon$. This implies that the Fréchet distance between the line segments $a_1 b_1$ and $a_2 b_2$ is at most ε . We replace the curve between a_1 and b_1 on T_1 with the segment $a_1 b_1$ and we replace the curve between a_2 and b_2 on T_2 with the segment $a_2 b_2$. With these substitutions, for all the maximal sub-curves of $\tilde{\pi}_{\mathfrak{s}\mathfrak{t}}$ passing through the forbidden-space, we obtain two new polygonal curves T'_1 and T'_2 such that $\delta_F(T'_1, T'_2) \leq \varepsilon$. Observation 6 implies that the sum of the lengths of the removed sub-curves of T_1 and T_2 is equal to $Q_{\tilde{\alpha}_1 \tilde{\alpha}_2}^B$. So, the sum of the lengths of the substituted curves does not exceed $Q^B(T_1, T_2) + \delta \cdot \max\{|T_1|, |T_2|\}$. The running time follows directly from Lemma 17 by running the linear time algorithm for finding a shortest path in a directed acyclic graph G [74]. We summarize our result for the MinEx problem in the following theorem.

Theorem 6. *Let T_1 and T_2 be two polygonal curves in the plane. Let $\varepsilon \geq 0$ and $\delta > 0$ be two real numbers. We can compute in $O(\frac{n^4}{\delta^2})$ time a pair of parameterizations $(\tilde{\alpha}_1, \tilde{\alpha}_2)$, and the quality $Q_{\tilde{\alpha}_1 \tilde{\alpha}_2}^B$ of T_1 and T_2 , such that $Q^B(T_1, T_2) \leq Q_{\tilde{\alpha}_1 \tilde{\alpha}_2}^B \leq Q^B(T_1, T_2) + \delta \cdot \max\{|T_1|, |T_2|\}$. Furthermore, if the distances between the starting and the ending points of T_1 and T_2 are at most ε , we can construct two polygonal curves T'_1 and T'_2 , realizing $Q_{\tilde{\alpha}_1 \tilde{\alpha}_2}^B$ and such that $\delta_F(T'_1, T'_2) \leq \varepsilon$.*

Since the length of an xy -increasing path in F is equal to the sum of the lengths of the

subpaths going through the free and the forbidden spaces, we derive the following corollary for the **MaxIn** problem.

Corollary 5. *Let T_1 and T_2 be two polygonal curves in the plane. Let $\varepsilon \geq 0$ and $\delta > 0$ be two real numbers. We can compute in $O(\frac{n^4}{\delta^2})$ time a pair of parameterizations $(\tilde{\alpha}_1, \tilde{\alpha}_2)$, and the quality $Q_{\tilde{\alpha}_1 \tilde{\alpha}_2}^W$ of T_1 and T_2 , such that $Q^W(T_1, T_2) - \delta \cdot \max\{|T_1|, |T_2|\} \leq Q_{\tilde{\alpha}_1 \tilde{\alpha}_2}^W \leq Q^W(T_1, T_2)$. Furthermore, if the distances between the starting and the ending points of T_1 and T_2 are at most ε , we can construct two polygonal curves T'_1 and T'_2 , realizing $Q_{\tilde{\alpha}_1 \tilde{\alpha}_2}^W$ and such that $\delta_F(T'_1, T'_2) \leq \varepsilon$.*

Proof. Let π_{st}^B (respectively, π_{st}^W) be the sum of the unweighted lengths of subpaths of π_{st} , going through the forbidden (respectively, free) space. Analogously, we define $\tilde{\pi}_{st}^B$ and $\tilde{\pi}_{st}^W$ for $\tilde{\pi}_{st}$. For a solution to the **wLongMP** problem to be optimal, the length of the path in the free space must be maximized. Hence, the quality of the solution to the **MaxIn** problem (provided by $\tilde{\pi}_{st}$) is $Q_{\tilde{\alpha}_1 \tilde{\alpha}_2}^W \leq Q^W(T_1, T_2)$. Moreover, the following equations hold:

$$\pi_{st}^B + \pi_{st}^W = |T_1| + |T_2|, \quad (4.10)$$

$$\tilde{\pi}_{st}^B + \tilde{\pi}_{st}^W = |T_1| + |T_2|, \quad (4.11)$$

where $|T_i|$ is the Euclidean length of T_i , for $i = 1, 2$. Lemma 17 gives us

$$\begin{aligned} \tilde{\pi}_{st}^B &\leq \pi_{st}^B + \delta \cdot \max\{|T_1|, |T_2|\} \\ |T_1| + |T_2| - \tilde{\pi}_{st}^W &\leq |T_1| + |T_2| - \pi_{st}^W + \delta \cdot \max\{|T_1|, |T_2|\} \end{aligned} \quad \text{by (4.11) and (4.10),}$$

from which we get $\pi_{st}^W - \delta \cdot \max\{|T_1|, |T_2|\} \leq \tilde{\pi}_{st}^W$. Since in the context of the **MaxIn** problem we have $Q_{\tilde{\alpha}_1 \tilde{\alpha}_2}^W = \tilde{\pi}_{st}^W$ and $Q^W(T_1, T_2) = \pi_{st}^W$, we obtain $Q^W(T_1, T_2) - \delta \cdot \max\{|T_1|, |T_2|\} \leq Q_{\tilde{\alpha}_1 \tilde{\alpha}_2}^W \leq Q^W(T_1, T_2)$. $\square$

4.3 Improvement

In this section, we improve the running times of the algorithms from the previous section. Recall that we computed an arrangement A induced by grid lines, intersection lines and ellipses, and then defined a graph G on top of A . Using G , we designed approximation algorithms to solve the **MinEx** and **MaxIn** problems. The running times of those algorithms are proportional to the size of G . Therefore, one way to improve the running time is to define a smaller graph G^* from which we can construct an approximate solution with the same accuracy. We will show how to construct such a graph G^* of size $O\left(\frac{n^3}{\delta} \log\left(\frac{n}{\delta}\right)\right)$.

4.3.1 An Auxiliary Lemma

We first present an abstract problem and explain how to solve it, see Lemma 18. In Section 4.3.2, we use Lemma 18 to construct G^* .

Suppose Ω is a convex region and P is a set of n points on $\partial\Omega$. Our goal is to construct a directed graph $G'(P)$ such that:

1. The vertices of $G'(P)$ are the points in P , together with some Steiner points.
2. For each pair of points $p_i, p_j \in P$ such that p_j dominates p_i , there exists a directed xy -increasing path $\tilde{\pi}_{ab}^*$ in $G'(P)$ such that:
 - (a) The path $\tilde{\pi}_{ab}^*$ goes from the vertex a , corresponding to p_i , to the vertex b , corresponding to p_j .
 - (b) The cost of $\tilde{\pi}_{ab}^*$ is $|p_i p_j|_1$, where $|p_i p_j|_1$ is the length of $p_i p_j$ with respect to the L_1 -metric.

We could construct such a graph $G'(P)$ by examining $\binom{n}{2}$ pairs of points from P . However, we show that $G'(P)$ can be constructed with the aid of Steiner points in $O(n \log n)$

time and using $O(n \log n)$ space. Our method is based on the following simple geometric observations [75].

Observation 7. *Let a (respectively, b) be a point in the South-West (respectively, North-East) quadrant of the Cartesian coordinate system. Then, there exists an L_1 -shortest path from a to b that passes through the origin.*

Observation 8. *Let a and b be two points such that b dominates a . Any xy -increasing path from a to b lies inside the bounding box of a and b . Furthermore, all xy -increasing paths from a to b , have the same length with respect to the L_1 -metric.*

We compute $G'(P)$ as follows. Initialize the vertex set of $G'(P)$ to be P . Then, add Steiner points in the following way. Compute the vertical median line m , splitting P into at least $\lfloor \frac{|P|}{2} - 1 \rfloor$ points to the left and at least $\lfloor \frac{|P|}{2} - 1 \rfloor$ points to the right of m , respectively (Figure 4.7(a)). Let m_1 be the upper and m_2 be the lower intersection points of m with $\partial\Omega$. Denote by ℓ_1 and ℓ_2 the horizontal lines containing m_1 and m_2 , respectively. Add m_1 and m_2 as Steiner points to the set of vertices of $G'(P)$. Partition P into three sets as follows. Let P_{above} be the set of points lying above ℓ_1 , P_{below} be the set of points lying below ℓ_2 and let $P_{middle} = P \setminus (P_{below} \cup P_{above})$. For each point $p_i \in P_{middle}$, compute its orthogonal projection p_i^m onto m and add p_i^m as a Steiner point. Moreover, add the edge (p_i, p_i^m) to $G'(P)$, directed with respect to the xy -increasing orientation. This projection onto m induces an ordering among the points in P_{middle} , with respect to increasing y -coordinate. Let $\langle p_1, \dots, p_k \rangle = P_{middle}$ be this ordering. For $i = 1, \dots, k-1$, add the edges (p_i^m, p_{i+1}^m) to $G'(P)$, directed with respect to the xy -increasing orientation. Also, add the edges (m_2, p_1^m) and (p_k^m, m_1) . For each vertex $p_i \in P_{above}$ (respectively, $p_j \in P_{below}$), add the edge (m_1, p_i) (respectively, (p_j, m_2)). The weight of each edge is its length with respect to the L_1 -metric. Let P_{left} (respectively, P_{right}) be the set of elements of P lying to the left (respectively, right) of m . Recursively apply the construction to P_{left} and P_{right} .

Figure 4.7: (a) The point set P is partitioned with respect to its median line m . Each $p_i \in P_{middle}$ is projected onto m (blue arrows and red points). The projections are ordered with respect to their y -coordinates (orange arrows). Each $p_j \in P_{above}$ (respectively, $p_j \in P_{below}$) is connected to m_1 (respectively, m_2) (dark green arrows). (b) Each $v \in V_{\partial\mathcal{E}}$ is connected by directed xy -increasing edges (light green edges) to all the sides of $\partial C^{i,j}$.

Lemma 18. *Let Ω be a convex region and P be a set of n points on its boundary. For each pair of points $p_i, p_j \in P$ such that p_j dominates p_i , there exists an xy -increasing path $\tilde{\pi}_{p_i p_j}$ in $G'(P)$ from the vertex corresponding to p_i to the vertex corresponding to p_j . Furthermore, the cost of $\tilde{\pi}_{p_i p_j}$ is $|p_i p_j|_1$ and the complexity of $G'(P)$ is $O(n \log n)$.*

Proof. The points p_i and p_j are separated by a median line in one of the recursive calls. Since p_j dominates p_i , the projection of p_i is below the projection of p_j on that median line. Thus, they are connected by an xy -increasing path $\tilde{\pi}_{p_i p_j}$ in $G'(P)$, since all the edges of $G'(P)$ are xy -increasing. The cost of $\tilde{\pi}_{p_i p_j}$ follows from Observation 8. The recursion depth is at most $O(\log n)$, since the problem is partitioned with respect to the median. In each call, a linear number of edges and Steiner points, with respect to the size of the input, are added. Thus $G'(P)$ has $O(n \log n)$ vertices and directed edges. $\square$

4.3.2 Construction of G^*

In Section 4.2, we showed how to find an approximate solution to **wShortMP** in $O\left(\frac{n^4}{\delta^2}\right)$ time. To do so, we defined a graph G of size $O\left(\frac{n^4}{\delta^2}\right)$, computed a shortest path $\tilde{\pi}$ in G and proved that $\tilde{\pi}$ approximates the solution to **wShortMP**. In this section, we show how to compute an approximate solution from a graph G^* of size $O\left(\frac{n^3}{\delta} \log\left(\frac{n}{\delta}\right)\right)$. The graph G^* is defined from $G = (V, E)$ as follows. For each parameter cell $C^{i,j}$, we restrict V to the boundaries of $C^{i,j}$ and $\mathcal{E}$, where $\mathcal{E}$ is the ellipse corresponding to $C^{i,j}$. Formally, let $V_{\partial C^{i,j}} = \partial C^{i,j} \cap V$ and $V_{\partial \mathcal{E}} = \partial \mathcal{E} \cap V$. Moreover, let V' be the set of vertices of $G'(V_{\partial \mathcal{E}})$, as defined in Section 4.3.1, where $\Omega := \mathcal{E}$. The vertex set of G^* for $C^{i,j}$ is $V_{\partial C^{i,j}} \cup V_{\partial \mathcal{E}} \cup V'$.

Lemma 19. *The number of vertices of G^* in the cell $C^{i,j}$ is $|V_{\partial C^{i,j}} \cup V_{\partial \mathcal{E}} \cup V'| = O\left(\frac{n}{\delta} \log\left(\frac{n}{\delta}\right)\right)$.*

Proof. Let us first consider the grid lines passing through the cell $C^{i,j}$. From the construction of G , there are at most $2\frac{n}{\delta}$ such grid lines and at most $\frac{n}{\delta}$ of them are vertical. All of these vertical grid lines may intersect $\partial C^{i,j}$ and $\partial \mathcal{E}$, creating at most $4\frac{n}{\delta}$ vertices. The same observation is valid for the horizontal grid lines.

We now consider the intersection lines passing through $C^{i,j}$. Notice that each horizontal intersection line is created by the intersection of a vertical grid line with an ellipse in the same row of the free-space diagram. And each vertical grid line may create at most two horizontal intersection lines in that row of the free-space diagram. Therefore, in each row of the free-space diagram, there are at most $2\frac{n}{\delta}$ horizontal intersection lines. Moreover, each horizontal intersection line creates at most four vertices in $C^{i,j}$, corresponding to the intersections with $\partial C^{i,j}$ and $\partial \mathcal{E}$. This sums up to $8\frac{n}{\delta}$ vertices in $C^{i,j}$ created by the horizontal intersection lines. The same argument applies to the vertical intersection lines.

Moreover, by Lemma 18, the size of V' is $O\left(\frac{n}{\delta} \log\left(\frac{n}{\delta}\right)\right)$. Consequently, the size of $V_{\partial C^{i,j}} \cup V_{\partial \mathcal{E}} \cup V'$ is $4\frac{n}{\delta} + 4\frac{n}{\delta} + 8\frac{n}{\delta} + 8\frac{n}{\delta} + O\left(\frac{n}{\delta} \log\left(\frac{n}{\delta}\right)\right) = O\left(\frac{n}{\delta} \log\left(\frac{n}{\delta}\right)\right)$. $\square$

For each cell $C^{i,j}$, we define the set of edges of G^* as follows. For each vertex $v \in V_{\partial \mathcal{E}}$,

there is an edge between v and each of its four projections on the sides of $\partial C^{i,j}$ (see Figure 4.7(b)), where $\mathcal{E}$ is the ellipse corresponding to $C^{i,j}$. Each of these edges are directed with respect to the xy -increasing orientation. Furthermore, every two consecutive vertices on each side of $C^{i,j}$ are connected by a directed xy -increasing edge. Moreover, each edge in $G'(V_{\partial\mathcal{E}})$ is an edge in $C^{i,j}$. By Lemma 19, G^* has $O\left(\frac{n}{\delta} \log\left(\frac{n}{\delta}\right)\right)$ vertices in $C^{i,j}$. From the above discussion, G^* has $O\left(\frac{n}{\delta} \log\left(\frac{n}{\delta}\right)\right)$ edges in $C^{i,j}$.

Let e be any edge of G^* . The weight of e is the length of the line segment corresponding to the intersection of e with the forbidden-space. In particular, each edge from $G'(V_{\partial\mathcal{E}})$ has a weight of zero.

Finally, G^* is the union of all the vertices and edges defined for each cell of the free-space diagram. Therefore, the size of G^* is given by the following lemma.

Lemma 20. *The size of G^* is $O\left(\frac{n^3}{\delta} \log\left(\frac{n}{\delta}\right)\right)$.*

4.3.3 Improved Algorithms for the MinEx and MaxIn Problems

We now explain how to compute an approximate solution to the wShortMP problem by using G^* . We start with the following lemma.

Lemma 21. *Let $\tilde{\pi}_{s't'}$ in G be a weighted shortest path connecting two points s' and t' on the boundary of $C^{i,j}$. Then, there exists a path $\tilde{\pi}_{s't'}^*$ connecting s' and t' in G^* , such that $\|\tilde{\pi}_{s't'}^*\| \leq \|\tilde{\pi}_{s't'}\|$.*

Proof. Let $\mathcal{E}$ be the ellipse corresponding to $C^{i,j}$. Since $\tilde{\pi}_{s't'}$ corresponds to an xy -increasing path, there exists at most one vertex a (respectively, b) where $\tilde{\pi}_{st}$ enters (respectively, exits) $\mathcal{E}$. So, $\tilde{\pi}_{s't'}$ consists of three sub-paths $\tilde{\pi}_{s'a} \subset B$, $\tilde{\pi}_{ab} \subset W$ and $\tilde{\pi}_{bt'} \subset B$, whose concatenation is $\tilde{\pi}_{s't'}$. As the notation suggests, $\tilde{\pi}_{s'a}$ (respectively, $\tilde{\pi}_{ab}$, $\tilde{\pi}_{bt'}$) is a path in G from s' to a (respectively, a to b , and b to t'). Since $\tilde{\pi}_{s'a}$ is xy -increasing, a dominates s' . Moreover, a dominates the orthogonal projection of a onto the side of $C^{i,j}$ on which s' lies. Thus, there

exists a path $\tilde{\pi}_{s'a}^*$ connecting s' and a in G^* . Since $\tilde{\pi}_{s'a} \subset B$, we have $\|\tilde{\pi}_{s'a}^*\| \leq \|\tilde{\pi}_{s'a}\|$. Since $\tilde{\pi}_{ab}$ is xy -increasing, b dominates a . It follows from Lemma 18 that there exists a path $\tilde{\pi}_{ab}^* \subset W$ such that $\|\tilde{\pi}_{ab}^*\| = 0$, since the weight in the free-space is zero. Analogous to the construction of $\tilde{\pi}_{s'a}^*$, there exists a path in G^* such that $\|\tilde{\pi}_{bt'}^*\| \leq \|\tilde{\pi}_{bt'}\|$. Therefore, $\tilde{\pi}_{s't'}^* = \tilde{\pi}_{s'a}^* \cup \tilde{\pi}_{ab}^* \cup \tilde{\pi}_{bt'}^*$ is the required path. $\square$

The following theorem is an improvement over Theorem 6.

Theorem 7. *Let T_1 and T_2 be two polygonal curves in the plane. Let $\varepsilon \geq 0$ and $\delta > 0$ be two real numbers. We can compute in $O\left(\frac{n^3}{\delta} \log\left(\frac{n}{\delta}\right)\right)$ time a pair of parameterizations $(\tilde{\alpha}_1, \tilde{\alpha}_2)$, and the quality $Q_{\tilde{\alpha}_1 \tilde{\alpha}_2}^B$ of T_1 and T_2 , such that $Q^B(T_1, T_2) \leq Q_{\tilde{\alpha}_1 \tilde{\alpha}_2}^B \leq Q^B(T_1, T_2) + \delta \cdot \max\{|T_1|, |T_2|\}$. Furthermore, if the distances between the starting and the ending points of T_1 and T_2 are at most ε , we can construct two polygonal curves T'_1 and T'_2 , realizing $Q_{\tilde{\alpha}_1 \tilde{\alpha}_2}^B$ and such that $\delta_F(T'_1, T'_2) \leq \varepsilon$.*

Proof. Let $\tilde{\pi}_{st} \subset G$ be a weighted shortest xy -increasing path from $\mathfrak{s}$ to $\mathfrak{t}$, as constructed in Section 4.2. Let $C_1, \dots, C_k$ be the parameter cells crossed by $\tilde{\pi}_{st}$. Furthermore, let s'_i (respectively, t'_i) be the entering (respectively, leaving) point of $\tilde{\pi}_{st}$ with respect to C_i , for $i \in \{1, \dots, k\}$. By construction of G^* , we know that these points are also vertices of G^* . In particular, $s'_1 = \mathfrak{s}$ and $t'_k = \mathfrak{t}$. Let $\tilde{\pi}_{s'_i t'_i}$ be the subpath of $\tilde{\pi}_{st}$ connecting s'_i and t'_i . Therefore, $\tilde{\pi}_{s'_i t'_i}$ is a weighted shortest xy -increasing path from s'_i to t'_i . By Lemma 21, there is an xy -increasing path $\tilde{\pi}_{s'_i t'_i}^*$ in G^* connecting s'_i and t'_i , and such that $\|\tilde{\pi}_{s'_i t'_i}^*\| \leq \|\tilde{\pi}_{s'_i t'_i}\|$. Let $\tilde{\pi}_{st}^*$ be the concatenation of the subpaths $\tilde{\pi}_{s'_i t'_i}^*$. Since $\|\tilde{\pi}_{s'_i t'_i}^*\| \leq \|\tilde{\pi}_{s'_i t'_i}\|$ for all $i \in \{1, \dots, k\}$, we have $\|\tilde{\pi}_{st}^*\| \leq \|\tilde{\pi}_{st}\|$.

By Lemma 17 and Theorem 6, we have $\|\tilde{\pi}_{st}^*\| \leq \|\pi_{st}\| + \delta \cdot \max\{|T_1|, |T_2|\}$. From Theorem 6, we obtain an approximate solution to the MinEx problem. That solution is given by $\tilde{\pi}_{st}^*$ and the corresponding modified polygonal curves T'_1 and T'_2 . Moreover, we have $\delta_F(T'_1, T'_2) \leq \varepsilon$ and the sum of the lengths of the substituted curves does not exceed $Q^B(T_1, T_2) + \delta \cdot \max\{|T_1|, |T_2|\}$.

The running time follows directly from Lemma 20. □

Corollary 6. *Let T_1 and T_2 be two polygonal curves in the plane. Let $\varepsilon \geq 0$ and $\delta > 0$ be two real numbers. We can compute in $O\left(\frac{n^3}{\delta} \log\left(\frac{n}{\delta}\right)\right)$ time a pair of parameterizations $(\tilde{\alpha}_1, \tilde{\alpha}_2)$, and the quality $Q_{\tilde{\alpha}_1 \tilde{\alpha}_2}^W$ of T_1 and T_2 , such that $Q^W(T_1, T_2) - \delta \cdot \max\{|T_1|, |T_2|\} \leq Q_{\tilde{\alpha}_1 \tilde{\alpha}_2}^W \leq Q^W(T_1, T_2)$. Furthermore, if the distances between the starting and the ending points of T_1 and T_2 are at most ε , we can construct two polygonal curves T'_1 and T'_2 , realizing $Q_{\tilde{\alpha}_1 \tilde{\alpha}_2}^W$ and such that $\delta_F(T'_1, T'_2) \leq \varepsilon$.*

4.3.4 Is FPTAS Achievable?

It remains to be discussed whether it is possible to design an FPTAS (Fully Polynomial-Time Approximation Scheme) for the **MaxIn** problem. As already discussed, asking for a polynomial time $(1 - \delta)$ -approximation for the **MaxIn** problem is equivalent to searching for a polynomial time $(1 - \delta)$ -approximation algorithm for the **wLongMP** problem. The common methodology to approximate longest paths in the presence of weighted regions is to approximate those areas by “accurate enough” structures that are more easily manageable than the original ones. A standard approach to implement this is to overlay the relevant weighted areas, in our case the connected components of W , by a mesh having a “small enough” grid size, related to δ . For example, this could be achieved by carefully connecting additionally positioned Steiner points on W to a dense enough neighborhood graph. To guarantee a $(1 - \delta)$ -approximation for the **MaxIn** problem, we have to upper bound the resulting error, i.e. the grid size, by $\delta \cdot Q^W(T_1, T_2)$. As illustrated via Figures 4.2(a) and 4.2(b), in Section 4.1.3, we can imagine configurations in which the size of the areas to be covered could be arbitrarily larger than the quality provided by an optimal solution. This implies an arbitrary large complexity of the applied grid-like structure. We can make similar observations for the **MinEx** problem, that is, the quality of the optimal solution $Q^B(T_1, T_2)$ could be arbitrary small (see Figure 4.8 for an illustration). One typical way

Figure 4.8: A free-space diagram in which the part of π_{st} in the forbidden-space is arbitrary small, compared to the length of T_1 and T_2 , and hence the quality of the optimal solution $Q^B(T_1, T_2)$ could be arbitrary small.

which leads to efficient approximation algorithms is to argue that approximating objects satisfy some “fatness” properties [125, 126, 127]. Unfortunately, there is no indication that such properties can be fulfilled in general in our framework.

As we discussed, it seems difficult to construct an FPTAS. However, we will design a polynomial time $(1 - \delta)$ -approximation algorithm that depends on factors other than the size of the input and the approximation factor δ . Based on Observation 6, any solution for the **wLongMP** problem can be translated into a solution for the **MaxIn** problem. We know that these problems are solvable exactly in the L_1 -metric (see [120]). We now use this fact to design a δ -approximation algorithm in the L_2 -metric. Let T_1 and T_2 be two trajectories. Let F_1 and F_2 be the free space diagrams of T_1 and T_2 , with Fréchet distance ε , for the L_1 -metric and the L_2 -metric, respectively.

Step 1: Compute the exact solution for the **wLongMP** problem, with respect to the L_1 -metric, using [120]. Let the length of an optimal solution be γ .

Step 2: Let $C^{i,j}$ be a parameter cell of F_2 and $\mathcal{E}$ be the ellipse corresponding to $C^{i,j}$. Insert equally-spaced Steiner points on $\partial\mathcal{E}$ so that the length of the part of $\partial\mathcal{E}$ between any two neighboring Steiner points is $\frac{\gamma\delta}{8n}$.

Step 3: Let $\tilde{\pi}_{st}$ be the path from s to t obtained by applying Corollary 6 to T_1 and T_2 .

Let π_{st} be an optimal solution for the **wLongMP** problem with respect to the L_2 -metric. We have the following lemma.

Lemma 22. $\gamma \leq \|\pi_{\text{st}}\|$.

Proof. Let W_1 and W_2 be the free-spaces of F_1 and F_2 , respectively. For any pair of points $(a, b) \in W_1$ we know that the L_1 -distance between a and b is at most ε . Moreover, for any two points in the plane, we know that the Euclidean distance between them is bounded by their L_1 -distance. Hence, for any pair of points, if $(a, b) \in W_1$ then $(a, b) \in W_2$. Therefore, $W_1 \subseteq W_2$. Thus, the length of an xy -increasing longest path in W_1 is at most the length of an xy -increasing longest path in W_2 . $\square$

We conclude with the following theorem.

Theorem 8. *Let T_1 and T_2 be two polygonal curves in the plane. Let $\varepsilon \geq 0$ and $\delta > 0$ be two real numbers. We can compute in $O\left(\frac{n^3}{\gamma\delta} \log\left(\frac{n}{\gamma\delta}\right)\right)$ time a pair of parameterizations $(\tilde{\alpha}_1, \tilde{\alpha}_2)$, and the quality $Q_{\tilde{\alpha}_1\tilde{\alpha}_2}^W$ of T_1 and T_2 , such that $(1 - \delta) \cdot Q^W(T_1, T_2) \leq Q_{\tilde{\alpha}_1\tilde{\alpha}_2}^W \leq Q^W(T_1, T_2)$, where γ is the length of an optimal solution with respect to the L_1 -metric. Furthermore, if the distances between the starting and the ending points of T_1 and T_2 are at most ε , we can construct two polygonal curves T'_1 and T'_2 , realizing $Q_{\tilde{\alpha}_1\tilde{\alpha}_2}^W$ and such that $\delta_F(T'_1, T'_2) \leq \varepsilon$.*

Proof. Since the path π_{st} is xy -increasing, it intersects no more than $2n$ cells. Let $C^{i,j}$ be any parameter cell intersected by π_{st} . We bound the maximum error made by $\tilde{\pi}_{\text{st}}$ inside $C^{i,j}$ (see Figure 4.9).

Let π_{ab} be the part of π_{st} lying in $\mathcal{E}$, where $\mathcal{E}$ is the ellipse corresponding to $C^{i,j}$ and $a, b \in \partial\mathcal{E}$. Let s_1 and s_2 be two neighboring Steiner points on $\partial\mathcal{E}$ such that a lies between s_1 and s_2 and s'_1 and s'_2 be two neighboring Steiner points on $\partial\mathcal{E}$ such that b lies between s'_1 and s'_2 . Consider the cell C_a containing a and having s_1 and s_2 on opposite corners and the cell C_b containing b and having s'_1 and s'_2 on opposite corners. Let us consider the paths

Figure 4.9: Illustration of the proof of Theorem 8. The path π_{st} intersecting the parameter cell $C^{i,j}$ is represented by the black curve. The L_1 -distance between a and b is represented by the dotted-dash line. The L_1 -distance between the Steiner points s_2 and s'_2 is represented by the dotted line.

from the Steiner points in $C_a \cap \mathcal{E}$ to the Steiner points in $C_b \cap \mathcal{E}$. We give an upper bound on the error made by any such path. Observe that the length of the elliptic arc in C_a is an upper bound on the length of any side of C_a . The same observation applies to C_b . By construction, the length of the elliptic arcs between any two neighboring Steiner points is $\frac{\gamma\delta}{8n}$. Therefore, the maximum error made by $\tilde{\pi}_{st}$ in each cell is at most $4\frac{\gamma\delta}{8n} = \frac{\gamma\delta}{2n}$.

Recall that we measure the length of the paths in the free-space diagram with respect to the L_1 -metric. Therefore, $\|\tilde{\pi}_{st}\| \geq \|\pi_{st}\| - 2n\frac{\gamma\delta}{2n} = \|\pi_{st}\| - \gamma\delta$. By Lemma 22, it follows that $\|\tilde{\pi}_{st}\| \geq (1 - \delta)\|\pi_{st}\|$.

From Theorem 4.1 in [120], the above discussion and Corollary 6, the time complexity of our algorithm is $O(n^3 \log n) + O\left(\frac{n^3}{\gamma\delta} \log\left(\frac{n}{\gamma\delta}\right)\right) = O\left(\frac{n^3}{\gamma\delta} \log\left(\frac{n}{\gamma\delta}\right)\right)$. $\square$

4.4 Conclusion

Our approach to measure the similarity between two polygonal curves in the presence of outliers is to minimize the portions of the curves where the matching is not possible (MinEx problem) or maximize the matched sub-curves (MaxIn problem). We reduced our problems to that of finding an xy -increasing weighted path in the deformed free-space diagram in

L_1 -metric. For the **MinEx** problem (**MaxIn** problem) the free-space is weighted by zero (one) and the forbidden-space is weighted by one (zero). Motivated by unsolvability proof, we designed approximation algorithms for the **MinEx** and **MaxIn** problems. We proposed an algorithm, running in $\mathcal{O}\left(\frac{n^3}{\delta} \log\left(\frac{n}{\delta}\right)\right)$ time with additive approximation error. It is still open if there exist a FPTAS for this problem. However, as we have shown, it is possible to design a $(1 - \delta)$ -approximation algorithm that its complexity depends on the input size n , given approximation factor δ and the length of the optimum solution in L_1 -metric γ .

Note that Driemel and Har-Peled [25] observed that computing the Fréchet distance under shortcuts, connecting arbitrarily chosen points on the polygonal input curves is NP-hard. Therefore, they considered a restricted version, where shortcuts are only allowed to connect vertices of the polygonal curves. An analogous modification of the **MinEx** (**MaxIn**) Problem would lead to a significant simplification of the problem setting as well. However, Buchin et al. [120] observed that the solution for these restricted problems may be far from any optimal solution for the unrestricted problems.

Chapter 5

Minimum Backward Fréchet Distance

5.1 Introduction

As we discussed in Chapter 1, a variant of the standard Fréchet distance is the weak Fréchet distance (or non-monotone Fréchet distance) [22]. Brakatsoulas et al. [18] pointed out that the choice between the weak Fréchet distance and the standard Fréchet distance depends on the application. Figure 5.1 illustrates an example of two polygonal curves that appear similar and, with a small amount of backward movement, the curves are indeed in weak ε Fréchet distance. While in the standard Fréchet distance no backtracking is allowed, in the weak Fréchet distance, the moving entities are allowed to backtrack of an arbitrary length. A natural question arises: what is the minimum length required for such backward movements to achieve a particular Fréchet distance? This optimization problem is solved here. Note that a simultaneous minimization of the length of backward movements and the Fréchet distance, are conflicting objectives. Increasing (resp., decreasing) the maximum leash length decreases (resp., increases) the minimum required length of all backward movement portions. Hence, we search for an optimal solution, i.e., walks that minimizes the backward movements, with a fixed input upper bound on the leash length.

Figure 5.1: Moving backward from a to b allows to walk on T_1 and T_2 and keeping the distance between moving objects less than ε during the walk.

In the language of the dog-leash metaphor, the considered variant of the Fréchet distance of this chapter is motivated as follows: imagine that an upper bound for the allowed leash length is given as an input parameter. This could be a maximum allowed distance between the two moving entities so that they can quickly reunite, in case of an emergency. Now, we are searching for walks on the given curves, such that the entire distances traveled are minimized. If the Fréchet distance between the input curves is not larger than the input threshold, then the optimal solution is zero (since no backward movement is necessary). In this situation, a corresponding optimal pair of walks can be computed via the approach from [22], i.e., a pair of walks, which realizes the Fréchet distance. If this is not the case, an optimal solution, or more precisely any valid solution, may force the man and/or his dog to move backwards, such that the sum of the traveled distances is larger than the lengths of the two input curves. Our objective is to minimize the length of the portions that are traveled backwards, while guaranteeing that the leash length stays below the input threshold.

The structure of the chapter is as follows. In Section 5.2, we discuss the preliminaries and define the problem formally. Then, in Section 5.3, we explain a polynomial time algorithm to solve the problem exactly. Afterward, in Section 5.4, we improve the time and space complexity of the algorithm to $\mathcal{O}(n^2 \log n)$ and $\mathcal{O}(n^2)$ respectively. At the end, we conclude the chapter.

5.2 Problem Definition

Let $T_1 : [0, n] \rightarrow \mathbb{R}^2$ and $T_2 : [0, m] \rightarrow \mathbb{R}^2$ be two polygonal curves of complexity (number of segments) n and m respectively. W.l.o.g., assume that $m \leq n$. For a parameterization, f , of T_1 , let $\mathcal{B}_f \subseteq [0, 1]$ be the closure of the set of times in which $f(t)$ is decreasing (i.e., the movement is backward). For a pair of parameterizations we define its quality by (5.1), where $\|\cdot\|$ is the Euclidean length.

$$Q_{f,g}(T_1, T_2) := \int_{t \in \mathcal{B}_f} \|(T_1(f(t)))'\| dt + \int_{t \in \mathcal{B}_g} \|(T_2(g(t)))'\| dt \quad (5.1)$$

We formally define the minimum backward Fréchet distance as follows. For a pair of polygonal curves, T_1 and T_2 , and a given leash length, ε , we are looking for a pair of optimal parameterizations, (f, g) , as defined in (5.2). We consider only matchings that induce matched pairs of points within distance less than or equal to ε .

$$Q^\varepsilon(T_1, T_2) = \inf_{f,g} Q_{f,g}(T_1, T_2) \quad (5.2)$$

In Section 4.1.1, we mentioned a useful structure, called deformed free-space diagram, denoted by $\mathcal{F}$. Recall that for two polygonal curves, $T_1 : [0, n] \rightarrow \mathbb{R}^2$ and $T_2 : [0, m] \rightarrow \mathbb{R}^2$, with n and m segments respectively, $\mathcal{F}$ consists of nm parameter cells $C^{i,j}$. The free-space of $\mathcal{F}$ is denoted by W and the forbidden-space of $\mathcal{F}$ is denoted by B .

An instance of our problem is drawn in Figure 5.2a. In this figure, two polygonal curves, a leash length, ε , and the corresponding deformed free-space diagram are shown. As the deformed free-space diagram illustrates, to be able to walk on T_1 and T_2 with a leash length less than ε , there must be a backward movement on the polygonal curves. However, the possible walk is not unique. We are looking for a walk that has the minimum backward movement. In this example, the value for the optimal walk is $\delta(t_{11}, t_{12}) + \delta(t_{21}, t_{22})$ where δ is the walking distance between the two points on the corresponding curve.

5.3 Algorithm

In order to solve the minimum backward Fréchet distance problem, we transform it to a shortest path problem on a graph, $\mathcal{G}_v = \langle V, E \rangle$. Consider the forbidden-space, B , as obstacle. We define the visibility inside the free-space, W , with respect to the obstacle. If it is possible to link two points inside W by a line segment such that it does not intersect B , then we

(a)

(b)

Figure 5.2: (a) Two polygonal curves, T_1 and T_2 , and the leash length, ε , are shown. Also the corresponding deformed free-space diagram is drawn. (b) Two paths in the free-space are drawn: an arbitrary path Π' (black dashed line) and an optimal path $\Pi \in \mathcal{G}_v$ (red solid line).

say they are visible. The set of vertices, V , of $\mathcal{G}_v$ is the set of all vertices of W (the free-space). The vertices are the intersection points of the ellipses with the boundary of the configuration space cells, in addition to two points, $\mathfrak{s}$ and $\mathfrak{t}$. The bottom-left corner of the free-space diagram $\mathfrak{s}$, called the source, corresponds to the starting points of the polygonal curves. Also, the top-right corner of the free-space diagram $\mathfrak{t}$, called the target, corresponds to the ending points of the polygonal curves. Each vertex, $v \in V$, has two coordinates, v_x and v_y . Every two visible vertices of W , v_1 and v_2 , are linked by two directed edges in E , from v_1 to v_2 , $\langle v_1, v_2 \rangle$, and vice versa, $\langle v_2, v_1 \rangle$. If a directed edge $e = \langle v_1, v_2 \rangle \in E$ is xy -increasing (i.e., it is non-decreasing from v_1 to v_2 in both x and y axes), its weight is zero. If e is only x -increasing, its weight is $|v_{1y} - v_{2y}|$. If e is only y -increasing, its weight is $|v_{1x} - v_{2x}|$. Otherwise, the weight of e is $|v_{1x} - v_{2x}| + |v_{1y} - v_{2y}|$, which is equal to the L_1 -distance between two vertices. If either $\mathfrak{s}$ or $\mathfrak{t}$ are not in W , then there is no solution for the given leash length. Otherwise, both $\mathfrak{s}$ and $\mathfrak{t}$ are in $\mathcal{G}_v$ as vertices and we prove that a shortest path from $\mathfrak{s}$ to $\mathfrak{t}$ gives an optimal walk. Note that in this chapter the vertices of the graph are also presented by a point in W . Therefore, a path in the visibility graph could be converted identically to a path in W by connecting its consecutive vertices by a line segment.

Observation 9. *Let $\Pi : [0, 1] \rightarrow [0, n] \times [0, m]$ be a path in the free-space W , from $\mathfrak{s}$ to $\mathfrak{t}$. Π is equivalent to a pair of parameterizations, $f : [0, 1] \rightarrow [0 : n]$ and $g : [0, 1] \rightarrow [0 : m]$, of the two polygonal curves, that maintains the leash length at most ε , for all $t \in [0, 1]$.*

In this chapter, a path from $\mathfrak{s}$ to $\mathfrak{t}$ is denoted by its vertices, $\Pi : \langle \mathfrak{s} = p_1, p_2, \dots, p_k = \mathfrak{t} \rangle$. All line segments in Π are directed, $\overrightarrow{p_i p_{i+1}}$, $i = 1, \dots, k-1$. The length (i.e., cost) of a segment is a function of its direction and is denoted by $|\overrightarrow{p_i p_{i+1}}|$ and defined as follows. If it is xy -increasing, its length is zero. If it is only x -increasing, its length is $|p_{iy} - p_{i+1y}|$. If it is only y -increasing, its length is $|p_{ix} - p_{i+1x}|$. Otherwise, its length is $|p_{ix} - p_{i+1x}| + |p_{iy} - p_{i+1y}|$, which is equal to the L_1 -distance between p_i and p_{i+1} . The length of a path, $|\Pi|$, is the sum of the length of its segments. In addition, the notation Π_i is used to denote the sub-path of Π from p_1 to p_i .

In this chapter, we use norms in two spaces: (1) the Euclidean space of the input polygonal curves, called the input space, (2) the deformed free-space diagram, called the configuration space. In the input space, we use the Euclidean length of a polygonal curve T and denote it by $\|T\|$. In configuration space, we use two norms to measure the length of a path Π : the length of a path in the L_1 metric, denoted by $|\Pi|_1$, and the direction-based norm, defined in the last paragraph, denoted by $|\Pi|$.

The corollary of the following lemma and Observation 9 is that the corresponding path of an optimal pair of parameterization, called optimal path, is a subset of $\mathcal{G}_v$.

Lemma 23. *For any path $\Pi : \langle \mathbf{s} = p_1, p_2, \dots, p_{k_1} = \mathbf{t} \rangle$ in W , there is a path $\Pi' : \langle \mathbf{s} = p'_1, p'_2, \dots, p'_{k_2} = \mathbf{t} \rangle$ in W such that $\Pi' \subset \mathcal{G}_v = \langle V, E \rangle$ and $|\Pi'| \leq |\Pi|$.*

In order to avoid repetition, we removed the proof of this lemma. The details of the inductive proof is provided in [50]. In Chapter 6, Lemma 27, we prove the lemma in a more general setting (i.e., weighted).

Corollary 7. *There is an optimal path which is a subset of the visibility graph of the free-space diagram, $\mathcal{G}_v$.*

Proof. Assume Π is an optimal path in the free-space W . If Π is not a subset of $\mathcal{G}_v$, then, by Lemma 23 there is a path, Π' in W such that $\Pi' \subset \mathcal{G}_v$ and $|\Pi'| \leq |\Pi|$. Since Π is an optimal path, $|\Pi'| = |\Pi|$. □

Corollary 8. *An optimal pair of parameterizations is achievable via a shortest path in the visibility graph, $\mathcal{G}_v$.*

Proof. Follows directly from Observation 9 and Corollary 7. □

Theorem 9. *Assume that T_1 and T_2 are two polygonal curves and a leash length, ε , is given. A parameterization of T_1 and T_2 that minimizes the backward movement during the walk can be found in polynomial time and space.*

Proof. The correctness follows directly from Corollary 8. For the space complexity, observe that the free-space diagram has a complexity of $\mathcal{O}(n^2)$. Therefore, the number of edges of $\mathcal{G}_v$ (and the total space complexity) is upper-bounded by $\mathcal{O}(n^4)$. It is possible to construct the topology of the graph in $\mathcal{O}(n^4 + n^2 \log n) = \mathcal{O}(n^4)$ time by the method proposed in [123]. Therefore, the construction of $\mathcal{G}_v$ takes $\mathcal{O}(n^4)$ time. It is possible to find a shortest path in the graph in $\mathcal{O}(|E| + |V| \log |V|) = \mathcal{O}(n^4)$. Therefore, the total time complexity is $\mathcal{O}(n^4)$. Note that if the representing nodes for $\mathfrak{s}$ and $\mathfrak{t}$ in $\mathcal{G}_v$ are not in a connected component of $\mathcal{G}_v$, then there is no feasible walk with the leash length of at most ε . Therefore, the algorithm halts with the answer of no feasible solution. $\square$

5.4 Improvement

In Corollary 7, in the previous section, we proved that the visibility graph of the free-space, $\mathcal{G}_v$, contains an optimal solution to our problem. As we will show next, it is not necessary to compute the complete visibility graph. In order to find an optimal path, we transform W to a polygonal domain, $\mathcal{D}$. The generated polygonal domain could have some holes. Then, we use the recent algorithm of Chen and Wang [76] to compute an L_1 shortest path in $\mathcal{D}$.

The process of building $\mathcal{D}$ from the free-space W is as follows. For each cell $C^{i,j}$, find the intersection of the boundary of $C^{i,j}$ with its corresponding ellipse $\phi^{i,j}$. Assume that the intersection points are sorted CCW on the boundary of $\phi^{i,j}$, $\langle v_1, v_2, \dots, v_\rho \rangle$. If the part of the boundary of $\phi^{i,j}$ from v_i to v_{i+1} , $i = 1, \dots, \rho - 1$, is inside $C^{i,j}$, connect v_i to v_{i+1} by a line segment (Figure 5.3), otherwise, ignore it. To complete the process of building $\mathcal{D}$ from W , add the line segments of the bounding polygon around the free-space diagram.

A known property of the free-space diagram states that every two adjacent cells in the free-space diagram, $C^{i,j}$ and $C^{i+1,j}$, have the same set of intersection points on the edge shared between them. Therefore, the mentioned process creates conforming polygonal chains. In addition, since $\phi^{i,j}$ is a convex object and the process of building $\mathcal{D}$ from W does not remove

any of the vertices of W , the visibility graph of $\mathcal{D}$ is identical to $\mathcal{G}_v$. By Corollary 7, there is an optimal path in $\mathcal{G}_v$. Therefore, there exists an optimal path in $\mathcal{D}$.

Lemma 24. *Let T_1 and T_2 be two polygonal curves, ε be the leash length, and W be the free-space of the deformed free-space diagram, $\mathcal{F}$. Suppose Π_1 and Π_2 are two paths in W from $\mathfrak{s}$ to $\mathfrak{t}$. If, in the L_1 norm, $|\Pi_1|_1 \leq |\Pi_2|_1$, then $|\Pi_1| \leq |\Pi_2|$.*

Proof. Suppose (f_1, g_1) and (f_2, g_2) are the corresponding parameterization pairs of Π_1 and Π_2 , respectively. We know that the length of any path in W from $\mathfrak{s}$ to $\mathfrak{t}$ is at least $\|T_1\| + \|T_2\|$, where $\|\cdot\|$ is the Euclidean length of a polygonal curve. Therefore, $|\Pi_1|_1 = \|T_1\| + \|T_2\| + 2 * \int_{t \in \mathcal{B}_{f_1}} \|(T_1(f_1(t)))'\| dt + 2 * \int_{t \in \mathcal{B}_{g_1}} \|(T_2(g_1(t)))'\| dt$. Thus:

$$\begin{aligned}
|\Pi_1|_1 &\leq |\Pi_2|_1 \\
\Rightarrow \|T_1\| + \|T_2\| + 2 * \int_{t \in \mathcal{B}_{f_1}} \|(T_1(f_1(t)))'\| dt + 2 * \int_{t \in \mathcal{B}_{g_1}} \|(T_2(g_1(t)))'\| dt \\
&\leq \|T_1\| + \|T_2\| + 2 * \int_{t \in \mathcal{B}_{f_2}} \|(T_1(f_2(t)))'\| dt + 2 * \int_{t \in \mathcal{B}_{g_2}} \|(T_2(g_2(t)))'\| dt \\
&\Rightarrow \int_{t \in \mathcal{B}_{f_1}} \|(T_1(f_1(t)))'\| dt + \int_{t \in \mathcal{B}_{g_1}} \|(T_2(g_1(t)))'\| dt \\
&\leq \int_{t \in \mathcal{B}_{f_2}} \|(T_1(f_2(t)))'\| dt + \int_{t \in \mathcal{B}_{g_2}} \|(T_2(g_2(t)))'\| dt \\
&\Rightarrow |\Pi_1| \leq |\Pi_2|
\end{aligned}$$

□

Theorem 10. *Assume we are given two polygonal curves, T_1 and T_2 , and a leash length, ε . A non-monotone parameterization of T_1 and T_2 that minimizes the backward movement during the walk can be found in $\mathcal{O}(n^2 \log n)$ time and $\mathcal{O}(n^2)$ space, where n is the number of segments in the input polygonal curves.*

Proof. As we discussed at the beginning of this section, the free-space W is transformed to a polygonal domain $\mathcal{D}$ (Figure 5.3). Based on Corollary 7, there is an optimal path in $\mathcal{D}$. By

Figure 5.3: A polygonal domain is constructed by replacing elliptic curves of the boundary of W by line segments.

Lemma 24, we know that a L_1 shortest path in $\mathcal{D}$ is also an optimal path for the minimum backward Fréchet distance problem. The algorithm by Chen and Wang [76] computes the L_1 shortest path in a polygonal domain, in $\mathcal{O}(k + h \log h)$ time and $\mathcal{O}(k)$ space, where k is the number of vertices of the input polygonal domain and h is the number of holes. In our problem, the size of the polygonal domain $\mathcal{D}$ and also the number of holes is $\mathcal{O}(n^2)$. Therefore, the total time complexity is $\mathcal{O}(n^2 \log n)$ and space complexity is $\mathcal{O}(n^2)$. $\square$

The implementation of our proposed algorithm is fairly straightforward. The followings are the main steps of the implementation of this algorithm:

1. The first step is to compute the free-space diagram. This diagram is explained in detail in [22]. Different implementations exist for this purpose. For example, we used the ipelet from [121] to produce the figures of this chapter. This implementation is open source.

2. The second step is converting the free-space diagram to a polygonal domain. At the beginning of Section 5.4 we explained how this conversion can be done by processing cells of free-space diagram one by one. The output of this step is a polygonal domain $\mathcal{D}$.
3. The last step is the implementation of the L_1 shortest path algorithm. Chen and Wang’s algorithm [76] is theoretically optimal. However, at this time we know of no implementation. There are alternatives for finding the L_1 shortest path in a polygonal domain (e.g., [64]). The easiest one to implement is the visibility graph based algorithm as explained in Section 5.3.

5.5 Conclusion

We studied a natural optimization problem for the weak Fréchet distance. In this measure, the union of backward movements on the two input curves T_1 and T_2 is minimized, to maintain a weak Fréchet distance of ε . We observed that this problem setting is dual to a weighted shortest path problem in a deformed free-space diagram, $\mathcal{F}$, of T_1 and T_2 in which only the portions of a path from $\mathcal{F}$ are measured if they are not xy -increasing. As a first approach, we showed that a corresponding optimal path in $\mathcal{F}$ is part of the visibility graph between the intersection points of the free-space ellipses with the boundary of the corresponding parameter cells and w.r.t. to the forbidden space of $\mathcal{F}$. This directly led to an algorithm which computes the minimum backward Fréchet distance in polynomial time. Then, as an improvement, we showed that it is not necessary to construct the entire visibility graph and by applying the very recent shortest path algorithm of [76] we obtained an improved running time of $\mathcal{O}(n^2 \log n)$ time to solve the minimum backward weak Fréchet problem exactly. For future work, other versions of the problem could be studied. The backward movement could be restricted to only one of the input curves. This restriction

may help to decrease the time complexity of the algorithm (See Chapter 8).

Chapter 6

Weighted Minimum Backward Fréchet Distance

6.1 Introduction

In Chapter 5, we introduced and solved an optimization problem on the weak Fréchet distance, called the *minimum backward Fréchet distance (MBFD)* problem. MBFD problem is to determine the minimum total length of backward movements on both input polygonal curves, required for the walks to achieve any given leash length. In that chapter, it is assumed that the cost (i.e., weight) of backward movement is uniform and depends only on the Euclidean distance traveled on each of the input polygonal curves. We proposed an algorithm with time complexity $\mathcal{O}(n^2 \log n)$ and space complexity $\mathcal{O}(n^2)$, to solve MBFD exactly. Here, in this chapter, we generalize this model to capture scenarios when the cost of backtracking on the input polygonal curves is not homogeneous. These weights could represent, for example, the cost of moving against a flow, or the cost for a moving entity (e.g., a human) to move backwards because of the entity's physiology [122]. Thus, in the new model, each edge of the input polygonal curves has an associated non-negative weight for backward movement. Then, the cost of backtracking on an edge is the Euclidean length of backward movement on that edge multiplied by the corresponding weight. The objective is to design an algorithm that a) finds a pair of walks that minimizes the sum of the costs on the edges of the curves, while guaranteeing that the leash length is at most ε , b) halts with the answer of no feasible solution if such a pair of walks does not exist (i.e., a longer leash length is required). We call this problem, the *weighted minimum backward Fréchet distance (WMBFD) problem*. Note that if the Fréchet distance between the input curves is already

at most ε , then no backtracking is necessary and the optimal solution is identical to a pair of walks that realizes the Fréchet distance.

Figure 6.1 shows an example. In this figure, two polygonal curves, T_1 and T_2 , and a length ε are drawn. The person walks on T_1 and the dog walks on T_2 . The weights, w_i , $i = 1, 2, 3$, for segments of T_1 are given. For this illustration, we let the cost of backward movement on T_2 be 1, on all segments of T_2 . In this example, it is impossible to walk from the starting point to the end and maintain the leash length at most ε , without moving backwards. Therefore, either the person or the dog or both must move backwards at some point during their respective walks. Six points, a_1, b_1, a_2, b_2, a_3 , and b_3 are specified on T_1 . If the person moves backwards, either from a_1 to b_1 , or from a_2 to b_2 , or from a_3 to b_3 , then the curves are at weak Fréchet distance ε . In this example, the Euclidean length of $\overline{a_1b_1}$ is less than the Euclidean length of $\overline{a_3b_3}$. However, the weight of moving backwards on the first segment is 10, while that on the third one is 1. Therefore, the pair of walks that minimizes the cost is as follows: the dog and the person move forwards together from the starting point, until the dog reaches the end of the third segment of T_2 and the person reaches the point a_3 on T_1 . Then, the dog keeps moving forwards until the end of the fourth segment of T_2 , while the person moves backwards from a_3 to b_3 . At the final step, they move forwards again together until the end of the respective curves. The cost of this pair of walks is the Euclidean length of $\overline{a_3b_3}$ multiplied by 1.

Figure 6.1: Moving backwards from a_3 to b_3 allows to walk on T_1 and T_2 and keeping the distance between moving objects at most ε during the walks while the cost is minimized.

The chapter is organized as follows. In Section 6.2, we discuss preliminaries and define the problem formally. In Section 6.3, we propose a polynomial time algorithm to solve the problem exactly. Then, in Section 6.4, we design an algorithm with improved time and space complexity. At the end, we conclude the chapter.

6.2 Preliminaries and Problem Definition

Let $T_1 : [0, n] \rightarrow \mathbb{R}^2$ and $T_2 : [0, m] \rightarrow \mathbb{R}^2$ be two polygonal curves of complexity (number of segments) n and m , respectively. W.l.o.g., assume that $m \leq n$. A vertex of T_1 is denoted by $T_1(i)$, $i = 0, \dots, n$. Analogously, a vertex of T_2 is denoted by $T_2(j)$, $j = 0, \dots, m$. An edge of T_1 between two vertices $T_1(i-1)$ and $T_1(i)$ is denoted by e_i , $i = 1, \dots, n$. Analogously, an edge of T_2 between two vertices $T_2(j-1)$ and $T_2(j)$ is denoted by e_j , $j = 1, \dots, m$. Furthermore, each edge, e_i (resp. e_j), has an associated non-negative weight (or cost) $w_i \in \mathbb{R}$ (resp. $w_j \in \mathbb{R}$). For a parameterization, f , of a polygonal curve, T_1 , let $\mathcal{B}_{f,i} \subseteq [0, 1]$ be the closure of the set of times in which $f(t)$ is decreasing (i.e., the movement is backward), and $f(t) \in [i-1, i]$ (i.e., the moving entity is on edge e_i at time t). The interval $\mathcal{B}_{g,j} \subseteq [0, 1]$ is defined analogously for a parameterization, g , of T_2 . For a pair of parameterizations, f and g , of two polygonal curves, T_1 and T_2 , we define the weighted quality by (6.1), where $\|T_1\|_{t \in \mathcal{B}_{f,i}} = \int_{t \in \mathcal{B}_{f,i}} \|(T_1(f(t)))'\| dt$, and $\|T_2\|_{t \in \mathcal{B}_{g,j}} = \int_{t \in \mathcal{B}_{g,j}} \|(T_2(g(t)))'\| dt$

$$\begin{aligned} \mathcal{WQ}_{f,g}(T_1, T_2) := & \sum_{i=1}^n \|T_1\|_{t \in \mathcal{B}_{f,i}} \cdot w_i \\ & + \sum_{j=1}^m \|T_2\|_{t \in \mathcal{B}_{g,j}} \cdot w_j \end{aligned} \tag{6.1}$$

Problem Definition. We define the WMBFD problem as follows. For a pair of weighted polygonal curves, T_1 and T_2 , and a given leash length, ε , we are looking for a pair of optimal parameterizations, (f, g) , as defined in (6.2). We consider only pairs of parameterizations

that guarantee to maintain the leash length at most ε , during the walks.

$$\mathcal{WQ}^\varepsilon(T_1, T_2) = \inf_{f,g} \mathcal{WQ}_{f,g}(T_1, T_2) \quad (6.2)$$

Weighted Deformed free-space diagram. In Section 4.1.1, we mentioned deformed free-space diagram, for two polygonal curves. Recall that for two polygonal curves, $T_1 : [0, n] \rightarrow \mathbb{R}^2$ and $T_2 : [0, m] \rightarrow \mathbb{R}^2$, with n and m segments respectively, the diagram consists of nm parameter cells $C^{i,j}$, for $i = 1, \dots, n$ and $j = 1, \dots, m$, whose interiors do not intersect with each other. The free-space of the diagram is denoted by W and the forbidden-space of the diagram is denoted by B . In this chapter, each cell, $C^{i,j}$, has two associated weights, w_x^i and w_y^j . The weight w_x^i is the weight of the edge between vertices $T_1(i-1)$ and $T_1(i)$ and the weight w_y^j is the weight of the edge between vertices $T_2(j-1)$ and $T_2(j)$. The resulting diagram is called the *weighted deformed free-space diagram* and is denoted by $\mathcal{F}$. The bottom left corner of $\mathcal{F}$ represents the starting points of T_1 and T_2 and is denoted by $\mathfrak{s}$. The top right corner of $\mathcal{F}$ represents the ending points of T_1 and T_2 and is denoted by $\mathfrak{t}$. For the given polygonal curves and ε in Figure 6.1, the corresponding weighted deformed free-space diagram is shown in Figure 6.2. As the diagram illustrates, to be able to walk on T_1 and T_2 with a leash length less than ε , there must be a backward movement on the polygonal curves (since there is no xy -monotone path from $\mathfrak{s}$ to $\mathfrak{t}$ in W). However, the possible walks are not unique. We are looking for a pair of walks that has the minimum backward movement cost, as we discussed in Section 6.1. In Figure 6.2, the red solid polygonal chain, called Π' , is a path in W that realizes an optimal pair of walks on T_1 and T_2 .

6.3 Algorithm

In this section, we propose a polynomial time algorithm, by transforming the WMBFD problem to a shortest path problem on a weighted directed graph, $\mathcal{G}_w = \langle V, E \rangle$.

Figure 6.2: The corresponding weighted deformed free-space diagram of the given polygonal curves in Figure 6.1 is drawn. Π (the black dashed path) is an arbitrary path in W . $\Pi' \subset \mathcal{G}_w$ (the red solid path) is a path in W that realizes a pair of parameterizations which gives an optimal solution for WMBFD. Π'' (the blue dashed path) is a path in W that realizes the optimal solution for MBFD.

Constructing Graph. Let $\mathcal{F}$ be the weighted deformed free-space diagram and B (resp. W) be the corresponding forbidden-space (resp. free-space) of $\mathcal{F}$. The vertices of W are the end points of the intervals on the boundary of the cells in $\mathcal{F}$ (i.e., at most 4 intervals per cell). The set of vertices, V , of $\mathcal{G}_w$, is the set of all vertices of W . Each vertex, v , has a x -coordinate (resp. y -coordinate), denoted by v_x (resp. v_y). Also, V contains $\mathfrak{s}$ and $\mathfrak{t}$. We say two points are visible if it is possible to link them by a line segment in W . Every two visible vertices, v_1 and v_2 , are linked by two directed edges in E , from v_1 to v_2 , $\langle v_1, v_2 \rangle$, and vice versa, $\langle v_2, v_1 \rangle$. The weight of a directed edge $e = \langle v_1, v_2 \rangle \in E$ is a function of its direction, the x - and y -coordinates of v_1 and v_2 , and the associated weights of the cells that e intersects. It is defined as follows: suppose e intersects a sequence of k cells, $\langle C^{\sigma(1),\sigma'(1)}, C^{\sigma(2),\sigma'(2)}, \dots, C^{\sigma(k),\sigma'(k)} \rangle$, of $\mathcal{F}$, where σ (resp. σ') is a function that maps the set $\{1, 2, \dots, k\}$ to a sub-sequence, or a reversed sub-sequence, of the index sequence $\langle 1, 2, \dots, n \rangle$ (resp. $\langle 1, 2, \dots, m \rangle$). The line segment e enters a cell, $C^{\sigma(i),\sigma'(i)}$, $i = 1, \dots, k$, at point $a^{\sigma(i),\sigma'(i)}$ and exits that cell at point $b^{\sigma(i),\sigma'(i)}$. Note that $a^{\sigma(1),\sigma'(1)}$ (resp. $b^{\sigma(k),\sigma'(k)}$) is identical to v_1 (resp. v_2). The x -coordinate (resp. y -coordinate) of a point, a , is denoted by a_x (resp. a_y). Note that each cell, $C^{\sigma(i),\sigma'(i)}$, has two associated weights, $w_x^{\sigma(i)}$ and $w_y^{\sigma'(i)}$. Let $|e|_{w_x} = \sum_{i=1}^k |a_x^{\sigma(i),\sigma'(i)} - b_x^{\sigma(i),\sigma'(i)}| \cdot w_x^{\sigma(i)}$ and $|e|_{w_y} = \sum_{i=1}^k |a_y^{\sigma(i),\sigma'(i)} - b_y^{\sigma(i),\sigma'(i)}| \cdot w_y^{\sigma'(i)}$. The weight of e , $|e|_w$, is calculated by the following function.

- If e is xy -increasing (i.e., it is non-decreasing from v_1 to v_2 in both x and y axes), then $|e|_w = 0$.
- If e is only x -increasing (resp. y -increasing), then $|e|_w = |e|_{w_y}$ (resp. $|e|_w = |e|_{w_x}$).
- Otherwise, $|e|_w = |e|_{w_x} + |e|_{w_y}$.

Finding an Optimal Solution. Based on the construction of the visibility graph, $\mathcal{G}_w$, both $\mathfrak{s}$ and $\mathfrak{t}$ are vertices in V . If either $\mathfrak{s}$ or $\mathfrak{t}$ is not in W , or there is no path from $\mathfrak{s}$ to $\mathfrak{t}$ in $\mathcal{G}_w$, then there is no solution for the given leash length. Otherwise, we prove that a shortest

path from $\mathfrak{s}$ to $\mathfrak{t}$, in $\mathcal{G}_w$, gives an optimal solution. Note that, in this chapter, a vertex of the graph is also presented by a point in W . Therefore, the geometric embedding of a path in $\mathcal{G}_w$ is constructed by connecting the consecutive vertices of the path in W by line segments.

In this chapter, we use norms in two spaces: (1) the Euclidean space of the input polygonal curves, called the input space, and (2) the weighted deformed free-space diagram, called the configuration space. In the input space, we use the Euclidean length of a polygonal curve T and denote it by $\|T\|$. In configuration space, a path from $\mathfrak{s}$ to $\mathfrak{t}$ in W , is denoted by its vertices, $\Pi : \langle \mathfrak{s} = p_1, p_2, \dots, p_k = \mathfrak{t} \rangle$. All segments in Π are directed, $\overrightarrow{p_i p_{i+1}}$, $i = 1, \dots, k-1$. The weighted length (or simply length), $|\cdot|_w$, of each segment of Π is calculated by the weight function that we explained earlier in this section, for computing the weight of a directed edge in the graph. The weighted length (or simply length) of a path, $|\Pi|_w$, is the sum of the length of its segments. In addition, the notation Π_i is used to denote the sub-path of Π from p_1 to p_i .

Correctness. Lemma 27 is at the heart of the correctness proof. In order to prove that lemma, we need Lemmas 25 and 26 and some definitions. This section is concluded by a corollary to Lemma 27 and Observation 9 (in Chapter 5), that is, in order to find an optimal pair of parameterizations in our problem setting, it suffices to find a shortest path from $\mathfrak{s}$ to $\mathfrak{t}$ in $\mathcal{G}_w$.

Definition 22. A path $\Pi \in W$ is *x-monotone* (resp. *y-monotone*), if and only if, any vertical (resp. horizontal) line intersects it at most once. Π is *xy-monotone*, if and only if, it is both *x-* and *y-monotone*.

Observation 10. Let a and b be two points in W such that $a_x \neq b_x$ and $a_y \neq b_y$. Suppose Π is a *xy-monotone* path from a to b . In addition, let $R(a, b)$ be the axes-aligned rectangle uniquely determined by a and b as corners. Π lies inside $R(a, b)$.

Lemma 25. Let Π_1 and Π_2 be two *xy-monotone* paths from a to b , in W , where $a, b \in W$.

Then, $|\Pi_1|_w = |\Pi_2|_w$. Furthermore, if Π_3 is an arbitrary path in W from a to b , then $|\Pi_1|_w \leq |\Pi_3|_w$.

Proof. If $a_x = b_x$ or $a_y = b_y$, then Π_1 and Π_2 are identical and the proof is trivial. Otherwise, by Observation 10, Π_1 and Π_2 lie in $R(a, b)$. Since Π_1 (also Π_2) is xy -monotone, its orthogonal projection onto the x and y axes is not overlapping and equal to the width and height of $R(a, b)$, respectively. Because Π_1 and Π_2 have identical projections onto the x and y axes and the weighted length of a path is defined based on its projection, then $|\Pi_1|_w = |\Pi_2|_w$. Also, any xy -monotone path from a to b has minimum weighted length among all paths from a to b , because its orthogonal projection onto x - and y -axis is non-overlapping. $\square$

Definition 23. Let a, b , and c be three distinct non-collinear points in W such that $\vec{ab} \in W$, $\vec{bc} \in W$ and $\vec{ac} \notin W$. We define the visibility chain from a to c , denoted by CC_a^c , as follows (see Figure 6.3a). Let B_{abc} be the portion of B (the black-space) that is inside the triangle $\triangle abc$. Suppose c is on the right side of $\vec{ab}$. The case that c is on the left is analogous. The gift-wrapping algorithm starts at a . Let $q_0 = a$. At the i -th iteration of the gift-wrapping algorithm, the point $q_i \in B_{abc}$, is selected such that B_{abc} lies on the right side of $\overrightarrow{q_{i-1}q_i}$. If c is visible to q_i , the gift-wrapping algorithm stops. Then, the visibility chain, directed from a to c , is $CC_a^c = \langle a, q_1, \dots, q_{last}, c \rangle$.

It is easy to prove that the following properties hold. First, CC_a^c exists and lies inside $\triangle abc$ and W . Second, CC_a^c and the line segment $\vec{ca}$ form a convex polygon. Thus, we say CC_a^c is a convex chain. Also, note that each q_i , $i = 1, \dots, last$, is represented by a node in the graph, $\mathcal{G}_w$, because it is a vertex of W .

Lemma 26. Let $a, b, c \in W$ be three distinct non-collinear points that $\vec{ab}, \vec{bc} \in W$ and $\vec{ac} \notin W$. If $\triangle abc$ lies in $R(a, c)$, then CC_a^c is xy -monotone and $|CC_a^c|_w = |\vec{ab}|_w + |\vec{bc}|_w$ (Figure 6.3a).

Proof. Since $\triangle abc$ lies in $R(a, c)$, the path that consists of $\vec{ab}$ and $\vec{bc}$ is a xy -monotone path

Figure 6.3: a) The visibility chain from a to c (the blue solid polygonal chain), $CC_a^c : \langle a, q_1, q_2, c \rangle$. b) The visibility chain from p'_z to p_{i+1} , $CC_{p'_z}^{p_{i+1}}$ (see Algorithm 4).

from a to c . If we show that CC_a^c is also a xy -monotone path from a to c , then by Lemma 25, $|CC_a^c|_w = |\vec{ab}|_w + |\vec{bc}|_w$.

To prove this, we need to define the angle of a vector. Suppose a directed segment in the free-space is a vector from the origin of the Cartesian coordinate system, when the angle of the vector is defined as the angle between that vector and the positive direction of the x -axis. Let α (resp., β) be the angle of $\vec{ab}$ (resp., $\vec{bc}$). Since $\triangle abc$ lies in $R(a, c)$, $|\alpha - \beta| = \pi/2$. In addition, since $R(a, c)$ is axes-aligned, precisely one of the four following cases is true: $0 \leq \alpha, \beta \leq \pi/2$, $\pi/2 \leq \alpha, \beta \leq \pi$, $\pi \leq \alpha, \beta \leq 3\pi/2$, $3\pi/2 \leq \alpha, \beta \leq 2\pi$. We denote the angles of segments, $\vec{aq_1}, \vec{q_1q_2}, \dots, \vec{q_{last}c}$, of $CC_a^c = \langle a, q_1, \dots, q_{last}, c \rangle$ by θ_μ , $\mu = 1, \dots, last + 1$. Since CC_a^c is a convex chain, the sequence of θ_μ , $\mu = 1, \dots, last + 1$, is in a sorted order (either increasing or decreasing), between α and β . Therefore, all θ_μ , $\mu = 1, \dots, last + 1$, are in one of the four mentioned quadrants and CC_a^c is thus xy -monotone. This proves the lemma. $\square$

Lemma 27. *For any path $\Pi : \langle \mathfrak{s} = p_1, p_2, \dots, p_{k_1} = \mathfrak{t} \rangle$ in W , there is a path $\Pi' : \langle \mathfrak{s} = p'_1, p'_2, \dots, p'_{k_2} = \mathfrak{t} \rangle$ in W such that $\Pi' \subset \mathcal{G}_w = \langle V, E \rangle$ and $|\Pi'|_w \leq |\Pi|_w$.*

Proof. We prove this lemma by designing an algorithm that constructs the path $\Pi' \subset \mathcal{G}_w$, through a transformation of path Π . Initially, Π' contains only $\mathfrak{s} = p'_1 = p_1$ and $p'_z = \mathfrak{s}$.

In this algorithm, p'_z is the latest vertex appended to the tail of Π' . Π' is constructed as follows. When considering the i -th vertex of Π , p_i , the algorithm tests if $\overrightarrow{p'_z p_{i+1}} \in W$. If so, p_i is skipped and Π' remains unchanged. Otherwise, the visibility chain from p'_z to p_{i+1} is constructed in $\Delta p'_z p_i p_{i+1}$ (Figure 6.3b) and its vertices from q_1 to q_c are appended to the tail of Π' . The algorithm for constructing Π' is stated in Algorithm 4. The correctness of this algorithm is given in Lemma 28. The output, Π' , of this algorithm is a path from $\mathfrak{s}$ to $\mathfrak{t}$, such that $\Pi' \subset \mathcal{G}_w$ and $|\Pi'|_w \leq |\Pi|_w$.

Algorithm 4 Constructing $\Pi' \in \mathcal{G}_w$

Input: The free-space W , A path $\Pi : \langle p_1, p_2, \dots, p_{k_1} \rangle$, where $\mathfrak{s} = p_1$ and $p_{k_1} = \mathfrak{t}$.

Output: A path $\Pi' : \langle p'_1, p'_2, \dots, p'_{k_2} \rangle$, where $\mathfrak{s} = p'_1$ and $p'_{k_2} = \mathfrak{t}$, such that $\Pi' \subset \mathcal{G}_w$ and $|\Pi'|_w \leq |\Pi|_w$.

```

1:  $\Pi' := \langle \mathfrak{s} \rangle$ ;
2:  $p'_z = \mathfrak{s}$ ;
3: for  $i=2 : k_1 - 1$  do
4:   if  $\overrightarrow{p'_z p_{i+1}} \notin W$  then
5:     Compute the visibility chain from  $p'_z$  to  $p_{i+1}$ ,  $CC_{p'_z}^{p_{i+1}} : \langle p'_z, q_1, \dots, q_c, p_{i+1} \rangle$ , in  $\Delta p'_z p_i p_{i+1}$ ;
6:     Append  $q_j$ ,  $j = 1, \dots, c$ , to  $\Pi'$ ;
7:      $p'_z = q_c$ ;
8: Append  $\mathfrak{t}$  to  $\Pi'$ ;
9: return  $\Pi'$ ;
```

□

Lemma 28. *Algorithm 4 is correct.*

Proof. We show that prior to the execution of the i -th iteration, $i = 2, \dots, k_1 - 1$, of the **for**-loop in Algorithm 4 the following invariant holds:

- I1. $\overrightarrow{p'_z p_i} \in W$
- I2. $\Pi'_z \subset \mathcal{G}_w$ and
- I3. $|\Pi'_z|_w + |\overrightarrow{p'_z p_i}|_w \leq |\Pi_i|_w$.

We prove this by induction on i , the index of the vertices of Π (i.e., the index of the **for**-loop in Algorithm 4). The base case is $i = 2$. In this case, p'_z is equal to $p'_1 = \mathfrak{s}$. Clearly, $\overrightarrow{p'_1 p_2} \in W$, $\Pi'_1 \subset \mathcal{G}_w$ and $|\Pi'_1|_w + |\overrightarrow{p'_1 p_2}|_w = |\Pi_2|_w$, because $\overrightarrow{p'_1 p_2} = \overrightarrow{p_1 p_2}$. The induction hypothesis is that the invariant holds for all loop iterations before the i -th iteration of the **for**-loop. In the following, we prove that it also holds before $(i + 1)$ -st iteration of the **for**-loop.

We distinguish between the two cases: a) $\overrightarrow{p'_z p_{i+1}} \in W$, b) $\overrightarrow{p'_z p_{i+1}} \notin W$.

Case a) Here, p_i is skipped and thus Π' remains unchanged. Therefore, I1 and I2 remain true (due to the induction hypothesis). In addition, since $\overrightarrow{p'_z p_{i+1}}$ is a segment in W , it is xy -monotone, and by Lemma 25, $|\overrightarrow{p'_z p_{i+1}}|_w \leq |\overrightarrow{p'_z p_i}|_w + |\overrightarrow{p_i p_{i+1}}|_w$. By induction hypothesis, we have $|\Pi'_z|_w + |\overrightarrow{p'_z p_i}|_w \leq |\Pi_i|_w$. By adding $|\overrightarrow{p_i p_{i+1}}|_w$ to both sides of the inequality, we obtain $|\Pi'_z|_w + |\overrightarrow{p'_z p_{i+1}}|_w \leq |\Pi_i|_w + |\overrightarrow{p_i p_{i+1}}|_w = |\Pi_{i+1}|_w$. Therefore, I3 remains true after i -th iteration (i.e., before $(i + 1)$ -st iteration).

Case b) Here, the **then** part of the **if** statement is entered and the visibility chain from p'_z to p_{i+1} is constructed. It is denoted by $CC_{p'_z}^{p_{i+1}} : \langle p'_z, q_1, \dots, q_c, p_{i+1} \rangle \in W$, where $q_j \in V$, $j = 1, \dots, c$. Then, q_j , from $j = 1$ to $j = c$, is appended to Π' . Finally, p'_z is updated to q_c . In the remainder, it is proved that I1, I2, and I3 hold.

Since $CC_{p'_z}^{p_{i+1}}$ is the visibility chain, I1 and I2 hold. In order to check if I3 holds, we need to analyze different cases. Each directed segment in W is of one of the following types: 1. xy -increasing 2. x -increasing and y -decreasing 3. y -increasing and x -decreasing 4. xy -decreasing. Therefore, there are 16 cases for the combination of two segments, $\overrightarrow{p'_z p_i}$ and $\overrightarrow{p_i p_{i+1}}$ (Figure 6.4). In all 16 cases, the orthogonal projection of $CC_{p'_z}^{p_{i+1}}$ onto the x -axis (resp., y -axis) is not longer than the sum of the orthogonal projections of $\overrightarrow{p'_z p_i}$ and $\overrightarrow{p_i p_{i+1}}$ onto the x -axis (resp., y -axis). Therefore, $|CC_{p'_z}^{p_{i+1}}|_w \leq |\overrightarrow{p'_z p_i}|_w + |\overrightarrow{p_i p_{i+1}}|_w$. Here, we only show the proofs for two cases of Figure 6.4 as the proofs for the other cases are analogous.

Consider the case when both $\overrightarrow{p'_z p_i}$ and $\overrightarrow{p_i p_{i+1}}$ are y -increasing and x -decreasing (see Case 6 in Figure 6.4). In this case, since $\Delta p'_z p_i p_{i+1}$ lies in $R(p'_z, p_{i+1})$, by Lemma 26, $|CC_{p'_z}^{p_{i+1}}|_w =$

Figure 6.4: There are 16 cases for the combination of two directed segments.

$|\overrightarrow{p'_z p_i}|_w + |\overrightarrow{p_i p_{i+1}}|_w$. By inductive hypothesis we have $|\Pi'_z|_w + |\overrightarrow{p'_z p_i}|_w \leq |\Pi_i|_w$. Add now $|\overrightarrow{p_i p_{i+1}}|_w$ to both sides of the inequality. We obtain $|\Pi'_z|_w + |CC_{p'_z}^{p_{i+1}}|_w \leq |\Pi_i|_w + |\overrightarrow{p_i p_{i+1}}|_w$. It follows that $|\Pi'_{z+c}|_w + |\overrightarrow{q_c p_{i+1}}|_w \leq |\Pi_{i+1}|_w$, where q_c is the latest vertex appended to Π' and Π'_{z+c} is the sub-path of Π' from index 1 to index $z+c$. Therefore, I3 holds. The proofs for cases 1, 11 and 16 are similar.

Now consider Case 9, when $\overrightarrow{p'_z p_i}$ is xy -increasing and $\overrightarrow{p_i p_{i+1}}$ decreases in both x and y axes (illustrated in Figure 6.5). In this case, $|\overrightarrow{p'_z p_i}|_w + |\overrightarrow{p_i p_{i+1}}|_w = 0 + |\overrightarrow{p_i p_{i+1}}|_w$.

The vertical line passing through p_{i+1} is denoted by $L_x^\perp$. The horizontal line passing through p_{i+1} is denoted by $L_y^\perp$. Suppose these lines are directed toward $+\infty$. The following two properties hold. First, any directed segment of $CC_{p'_z}^{p_{i+1}}$ that lies to the left of $L_x^\perp$ is increasing in x . Therefore, they have a weighted length of zero in the x -dimension. Second, any directed segment of $CC_{p'_z}^{p_{i+1}}$ that lies below $L_y^\perp$ is increasing in y . Therefore, they have a

weighted length of zero in the y -dimension. By these two properties, any directed segment of $CC_{p'_z}^{p_{i+1}} : \langle p'_z, q_1, \dots, q_c, p_{i+1} \rangle$ that lies to the left of $L_x^\perp$ and below $L_y^\perp$ is xy -increasing and has weighted length of zero.

Suppose $\overrightarrow{q_r q_{r+1}}$ is the first line segment in $CC_{p'_z}^{p_{i+1}}$ on the right side of $L_x^\perp$ that is x -decreasing. Since $CC_{p'_z}^{p_{i+1}}$ is a convex chain and is inside the triangle $\Delta p'_z p_i p_{i+1}$, the sub-chain $\langle q_r, \dots, q_c, p_{i+1} \rangle$ is x -monotone and its weighted length in the x -dimension is less than or equal to the weighted length of $\overrightarrow{p_i p_{i+1}}$ in the x -dimension. Therefore, the weighted length of $CC_{p'_z}^{p_{i+1}}$ in the x -dimension is less than or equal to the weighted length of $\overrightarrow{p_i p_{i+1}}$ in the x -dimension.

Now, suppose $\overrightarrow{q_u q_{u+1}}$ is the first line segment in $CC_{p'_z}^{p_{i+1}}$ above $L_y^\perp$ that is y -decreasing. Since $CC_{p'_z}^{p_{i+1}}$ is a convex chain and is inside the triangle $\Delta p'_z p_i p_{i+1}$, the sub-chain $\langle q_u, \dots, q_c, p_{i+1} \rangle$ is y -monotone and its weighted length in the y -dimension is less than or equal to the weighted length of $\overrightarrow{p_i p_{i+1}}$ in the y -dimension. Therefore, the weighted length of $CC_{p'_z}^{p_{i+1}}$ in the y -dimension is less than or equal to the weighted length of $\overrightarrow{p_i p_{i+1}}$ in the y -dimension.

To conclude, the weighted length of $CC_{p'_z}^{p_{i+1}}$, which is the sum of the weighted lengths of $CC_{p'_z}^{p_{i+1}}$ in the x - and y -dimensions, is less than or equal to the weighted length of $\overrightarrow{p_i p_{i+1}}$, which is the sum of the weighted lengths of $\overrightarrow{p_i p_{i+1}}$ in the x - and y -dimensions. Thus, $|CC_{p'_z}^{p_{i+1}}|_w \leq |\overrightarrow{p_i p_{i+1}}|_w = |p'_z p_i|_w + |\overrightarrow{p_i p_{i+1}}|_w$. By inductive hypothesis, $|\Pi'_z|_w + |p'_z p_i|_w \leq |\Pi_i|_w$. By adding $|\overrightarrow{p_i p_{i+1}}|_w$ to both sides of the inequality, we conclude $|\Pi'_z|_w + |CC_{p'_z}^{p_{i+1}}|_w \leq |\Pi'_z|_w + |p'_z p_i|_w + |\overrightarrow{p_i p_{i+1}}|_w \leq |\Pi_i|_w + |\overrightarrow{p_i p_{i+1}}|_w$. It follows that $|\Pi'_{z+c}|_w + |\overrightarrow{q_c p_{i+1}}|_w \leq |\Pi_{i+1}|_w$. Therefore, I3 also holds for this case. The proofs of the remaining cases are similar. $\square$

Corollary 9. *There is a path from $\mathfrak{s}$ to $\mathfrak{t}$ in $\mathcal{G}_w$ with minimum weighted length that lies in the free-space W .*

Proof. Assume Π is a path from $\mathfrak{s}$ to $\mathfrak{t}$ with minimum weighted length in the free-space W . If Π is not a subset of $\mathcal{G}_w$, then, by Lemma 27 there is a path from $\mathfrak{s}$ to $\mathfrak{t}$, Π' in W such that $\Pi' \subset \mathcal{G}_w$ and $|\Pi'|_w \leq |\Pi|_w$. Since Π has minimum weighted length, $|\Pi'|_w = |\Pi|_w$. $\square$

Figure 6.5: The segment from q_r to q_{r+1} is the first line segment in $CC_{p'_z}^{p_{i+1}}$ on the right side of $L_x^\perp$ that is x -decreasing. The segment from q_u to q_{u+1} is the first line segment in $CC_{p'_z}^{p_{i+1}}$ above of $L_y^\perp$ that is y -decreasing.

Corollary 10. *A shortest path in $\mathcal{G}_w$ yields an optimal pair of parameterizations for the WMBFD problem.*

Proof. Follows directly from Observation 9 (in Chapter 5), and Corollary 9. $\square$

Theorem 11. *Let T_1 and T_2 be two polygonal curves and ε be a given leash length. Each segment of T_1 and T_2 has an associated weight, corresponding to the backward movement on that segment. A pair of parameterizations of T_1 and T_2 that minimizes the weighted sum of the backward movements during the walks can be found in polynomial time and space.*

Proof. The correctness follows directly from Corollary 10. Since the weighted free-space diagram has a complexity of $\mathcal{O}(n^2)$, the number of edges of $\mathcal{G}_w$ (and the total space complexity) is upper-bounded by $\mathcal{O}(n^4)$. To find an edge of $\mathcal{G}_w$, a brute force algorithm checks the intersection of a line segment between two vertices with all $\mathcal{O}(n^2)$ ellipses. Therefore, the construction of $\mathcal{G}_w$ takes $\mathcal{O}(n^6)$ time. However, it is possible to construct the graph in $\mathcal{O}(n^4 + n^2 \log n) = \mathcal{O}(n^4)$ time by the method proposed in [123]. Then, compute the weight of each edge of $\mathcal{G}_w$ in constant time using prefix sums in linear space (see [131]). Therefore, the total time for constructing $\mathcal{G}_w$ is $\mathcal{O}(n^4)$. It is possible to find a shortest path in the graph

in $\mathcal{O}(n^4)$ time by Dijkstra's algorithm. Therefore, the total time complexity is $\mathcal{O}(n^4)$. Note that if the representing nodes for $\mathfrak{s}$ and $\mathfrak{t}$ in $\mathcal{G}_w$ are not in a connected component of $\mathcal{G}_w$, then there is no feasible walk with the leash length of at most ε . Therefore, the algorithm halts with the answer of “no feasible solution”. $\square$

6.4 Improvement

In Section 6.3, we showed that the weighted graph $\mathcal{G}_w = \langle V, E \rangle$ contains a path, from $\mathfrak{s}$ to $\mathfrak{t}$, that yields an optimal pair of parameterizations for solving the WMBFD problem. In this section, we will discuss that it is not necessary to compute the complete visibility graph, $\mathcal{G}_w$. As a result we will obtain an improved algorithm. There are two steps to obtain this improvement. In the first step, we discuss how to reduce the number of edges in E by a linear factor. Then, in the second step, we use Steiner points, to reduce the size of graph ensuring that it still contains a path, from $\mathfrak{s}$ to $\mathfrak{t}$, with minimum weighted length.

6.4.1 First Step

Let $\mathcal{G}_w' = \langle V, E' \rangle$ be a sub-graph of $\mathcal{G}_w$ such that $E' = \{e' \in E \mid e' \text{ lies in a row or column of } \mathcal{F}\}$, where $\mathcal{F}$ is the weighted free-space diagram.

Lemma 29. *There is a path in $\mathcal{G}_w'$ that realizes an optimal pair of parameterizations for the WMBFD problem.*

Proof. We will show that, for any directed edge $e = \langle u_1, u_2 \rangle \in E$ that is not in E' , we can construct a xy -monotone path, π_{u_1, u_2} , in $\mathcal{G}_w'$ (see Figure 6.6). Then, by Lemma 25, $|\pi_{u_1, u_2}|_w = |e|_w$. By Corollary 10, a shortest path, Π' in $\mathcal{G}_w$ yields an optimal solution. Therefore, if for any directed edge, $e = \langle u_1, u_2 \rangle$, of Π' , π_{u_1, u_2} exists in $\mathcal{G}_w'$, then there is a path in $\mathcal{G}_w'$ that realizes an optimal pair of parameterizations.

Now, we prove that π_{u_1, u_2} exists in $\mathcal{G}_w'$, for any directed edge $e = \langle u_1, u_2 \rangle \in E$. If e stays

Figure 6.6: a) The directed edge $e = \langle u_1 u_2 \rangle \in \mathcal{G}_w$ intersects a sequence of intervals on the boundary of the cells (red line segments). The edge is partitioned to three sub-edges, $\langle u_1 p_1, p_1 p_2, p_2 u_2 \rangle$. Each sub-edge is in a row or a column. b) The green dashed polygonal chain shows the xy -monotone path that is constructed in the first phase. The blue dotted polygonal chain shows the xy -monotone path that is in $\mathcal{G}_w'$.

in a row or column of $\mathcal{F}$, then $\pi_{u_1, u_2} = e$. Otherwise, e crosses several rows and columns. There are four cases, depending on the orientation of e : a) xy -increasing b) x -increasing and y -decreasing c) y -increasing and x -decreasing d) xy -decreasing. We prove this lemma for the last case. The proofs for the other 3 cases are analogous. Assume that e is xy -decreasing. The edge e intersects a sequence of intervals on the boundary of the cells of $\mathcal{F}$. We partition e into sub-edges so that each sub-edge is contained within a row or a column of $\mathcal{F}$, as follows (see Figure 6.6).

We traverse e from u_1 to u_2 . The point $p_1 \in e$ is the point where we exit the row and the column that contain u_1 . Therefore, any point after p_1 on e during the traversal is not in the row or the column that contains u_1 . We continue the traversal from p_1 to u_2 . The point $p_2 \in e$ is defined analogously. It is the point where we exit the row and the column that contain p_1 . We define p_i , $i = 3, \dots, z$, analogously with respect to p_{i-1} . Then, the sequence of sub-edges of e is $\langle \overrightarrow{u_1 p_1}, \overrightarrow{p_1 p_2}, \dots, \overrightarrow{p_z u_2} \rangle$.

Denote the interval that contains p_i , $i = 1, \dots, z$, by I_i . Let $u_1 = p_0 \in I_0$ and $u_2 = p_{z+1} \in I_{z+1}$. Note that I_i and I_{i+1} , $i = 0, \dots, z$, are on the boundary of a row or a column.

We say a point $q = (q_x, q_y)$ dominates a point $p = (p_x, p_y)$, if $p_x \leq q_x$ and $p_y \leq q_y$. In this proof, the endpoint of I_i that dominates p_i is denoted by q_i . We construct π_{u_1, u_2} , in two phases. In the first phase, we construct an intermediate xy -monotone path, π'_{u_1, u_2} , from u_1 to u_2 (the green dashed polygonal chain in Figure 6.6). Then, in the second phase, we transform it to a path, π_{u_1, u_2} , in $\mathcal{G}_w'$ (the blue dotted polygonal chain in Figure 6.6).

In the first phase, we start from $p_0 = u_1$. For each sub-edge of e , $\overrightarrow{p_i p_{i+1}}$, $i = 0, \dots, z$, we construct a xy -monotone path, $\pi'_{p_i, q_{i+1}}$, from p_i to q_{i+1} . Then, the concatenation of $\pi'_{p_i, q_{i+1}}$ and $\overrightarrow{q_{i+1} p_{i+1}}$ is a xy -monotone path, $\pi'_{p_i, p_{i+1}}$, from p_i to p_{i+1} , because q_{i+1} dominates p_{i+1} . Then, the concatenation of $\pi'_{p_i, p_{i+1}}$, $i = 0, \dots, z$ is a xy -monotone path, π'_{u_1, u_2} , from u_1 to u_2 . Now, we explain how to construct $\pi'_{p_i, q_{i+1}}$. If $\overrightarrow{p_i q_{i+1}} \in W$, then $\pi'_{p_i, q_{i+1}} = \overrightarrow{p_i q_{i+1}}$. It is obviously xy -monotone. If $\overrightarrow{p_i q_{i+1}} \notin W$, then $\pi'_{p_i, q_{i+1}}$ is the visibility chain, $CC_{p_i}^{q_{i+1}}$, from p_i to q_{i+1} , in $\Delta p_i p'_{i+1} q_{i+1}$, where p'_{i+1} is a point, defined as follows.. Let I'_{i+1} be the last interval that $\overrightarrow{p_i p_{i+1}}$ intersects before intersecting I_{i+1} and p'_{i+1} be the intersection point of I'_{i+1} and $\overrightarrow{p_i p_{i+1}}$. The point p'_{i+1} dominates q_{i+1} . Therefore, $\Delta p_i p'_{i+1} q_{i+1}$ lies in $R(p_i, q_{i+1})$, the axes-aligned rectangle that is determined by p_i and q_{i+1} as opposite corners. Thus, by Lemma 26, $CC_{p_i}^{q_{i+1}}$ is xy -monotone.

In the second phase, we transform π'_{u_1, u_2} to a xy -monotone path, π_{u_1, u_2} , in $\mathcal{G}_w'$. This transformation is done by replacing the edges in π'_{u_1, u_2} that are not in E' . These edges are $\overrightarrow{q_i p_i}$ and $\overrightarrow{p_i \mathcal{S}(p_i)}$, $i = 1, \dots, z$, where $\mathcal{S}(\cdot)$ is the successor operation and $\mathcal{S}(p_i)$ is the vertex after p_i in π'_{u_1, u_2} . These are the edges that connect p_i , $i = 1, \dots, z$, to the previous and next vertex of p_i in π'_{u_1, u_2} . Note that $\mathcal{S}(p_i)$ is a vertex in V and it may be q_{i+1} . If $\overrightarrow{q_i \mathcal{S}(p_i)} \in W$, then the two edges, $\overrightarrow{q_i p_i}$ and $\overrightarrow{p_i \mathcal{S}(p_i)}$, are replaced by $\overrightarrow{q_i \mathcal{S}(p_i)} \in E'$. Note that it is obviously xy -monotone. If $\overrightarrow{q_i \mathcal{S}(p_i)} \notin W$, then the two edges, $\overrightarrow{q_i p_i}$ and $\overrightarrow{p_i \mathcal{S}(p_i)}$, are replaced by the visibility chain, $CC_{q_i}^{\mathcal{S}(p_i)}$, from q_i to $\mathcal{S}(p_i)$, in $\Delta q_i p_i \mathcal{S}(p_i)$. Since π'_{u_1, u_2} is a xy -monotone path,

the concatenation of $\overrightarrow{q_i p_i}$ and $\overrightarrow{p_i \mathcal{S}(p_i)}$ is also a xy -monotone path. Therefore, $\Delta q_i p_i \mathcal{S}(p_i)$ lies in $R(q_i, \mathcal{S}(p_i))$. Thus, by Lemma 26, $CC_{q_i}^{\mathcal{S}(p_i)}$ is xy -monotone. Also, $CC_{q_i}^{\mathcal{S}(p_i)} \subset \mathcal{G}_w'$ since the vertices of this visibility chain belong to one column or one row of $\mathcal{F}$. By repeating this process for every p_i , $i = 1, \dots, z$, the resulting path, denoted by π_{u_1, u_2} , is in $\mathcal{G}_w'$. Since all sub-paths of π_{u_1, u_2} are xy -monotone, π_{u_1, u_2} is also xy -monotone. $\square$

Theorem 12. *Assume we are given two polygonal curves, T_1 and T_2 , and a leash length, ε . Each segment of T_1 and T_2 has an associated weight, corresponding to the backward movement on that segment. A pair of parameterizations of T_1 and T_2 that minimizes the weighted sum of the backward movements during the walks can be found in $\mathcal{O}(n^3)$ time and space, where n is the number of segments in the input polygonal curves.*

Proof. The correctness follows directly from Lemma 29. The weighted free-space diagram, $\mathcal{F}$, has $\mathcal{O}(n^2)$ cells and each vertex of $\mathcal{G}_w'$ on the boundary of a cell is connected to at most $\mathcal{O}(n)$ vertices of $\mathcal{G}_w'$ that are in the same row or column. Therefore, the number of edges of $\mathcal{G}_w'$ (and the total space complexity) is upper-bounded by $\mathcal{O}(n^3)$. It is possible to find all the edges of $\mathcal{G}_w'$ that lie in a column or row of $\mathcal{F}$ in $\mathcal{O}(n^2 + n \log n) = \mathcal{O}(n^2)$ time by the method proposed in [123]. Since there are $\mathcal{O}(n)$ rows and columns in $\mathcal{F}$, the total time complexity to find the topology of $\mathcal{G}_w'$ is $\mathcal{O}(n^3)$. In addition, to compute the weight of the edges that are within one row or column, $\mathcal{O}(n^2)$ time and $\mathcal{O}(n)$ space suffice (by using prefix sums, see [131]). Therefore, we can compute the weight of the edges of $\mathcal{G}_w'$ in $\mathcal{O}(n^3)$ time and $\mathcal{O}(n)$ space.

Using Dijkstra's algorithm, we find a shortest path in the graph in $\mathcal{O}(n^3 + n^2 \log n)$ time. Therefore, both time and space complexities of our algorithm are $\mathcal{O}(n^3)$. Note that if the nodes representing $\mathfrak{s}$ and $\mathfrak{t}$ in $\mathcal{G}_w'$ are not in a connected component of $\mathcal{G}_w'$, then there is no feasible walk with the leash length of at most ε . Therefore, the algorithm halts with the answer “no feasible solution”. $\square$

Figure 6.7: A row of a weighted free-space diagram is drawn. The boundary of the free-space of the row is highlighted by a red curve.

6.4.2 Second Step

As we discussed in Section 6.4.1, $|E'| = \mathcal{O}(n^3)$. It leads to an algorithm with $\mathcal{O}(n^3)$ time and space complexity. In this section, we decrease the number of edges in E' by using some Steiner points. More specifically, we construct a new graph $\mathcal{G}_w'' = \langle V'', E'' \rangle$ such that $V \subset V''$ and for any two vertices $u, v \in V$, $|\Pi'_{uv}|_w = |\Pi''_{uv}|_w$ where Π'_{uv} is the shortest path in $\mathcal{G}_w'$ from u to v and Π''_{uv} is the shortest path in $\mathcal{G}_w''$ from u to v . Recall that we denote the weighted free-space diagram by $\mathcal{F}$ and its free-space by W . $\mathcal{F}$ has m rows, $\mathcal{R}_1, \dots, \mathcal{R}_m$, and n columns, $\mathcal{C}_1, \dots, \mathcal{C}_n$. We denote the free-space of a row $\mathcal{R}_i$ by $W_{\mathcal{R}_i}$ and its boundary by $\partial W_{\mathcal{R}_i}$. Also, we denote the free-space of a column $\mathcal{C}_j$ by $W_{\mathcal{C}_j}$ and its boundary by $\partial W_{\mathcal{C}_j}$.

Observation 11. *Any vertical line intersects the boundary of $W_{\mathcal{R}_i}$ at most twice and any horizontal line intersects the boundary of $W_{\mathcal{C}_j}$ at most twice (see Figure 6.7 for an illustration).*

Lemma 30 characterizes a property of the weighted metric $|\cdot|_w$ in a row $\mathcal{R}_i$. Analogously, this property also holds in a column.

Lemma 30. *For two points $a, b \in W_{\mathcal{R}_i}$, let Π_1 be a L_1 shortest path, from a to b , in $W_{\mathcal{R}_i}$ and Π_w be a path, from a to b , in $W_{\mathcal{R}_i}$ that has the minimum weighted length $|\cdot|_w$. Then, $|\Pi_1|_w = |\Pi_w|_w$.*

Proof. Let Π_1^x be the orthogonal projection of Π_1 onto the x -axis and Π_1^y be the orthogonal projection of Π_1 onto the y -axis. Also, let Π_w^x be the orthogonal projection of Π_w onto the

x -axis and Π_w^y be the orthogonal projection of Π_w onto the y -axis. We denote the Euclidean length of the portion of Π_1^x that lies in j -th column by $|\Pi_1^x|^j$ and the Euclidean length of the portion of Π_w^x that lies in j -th column by $|\Pi_w^x|^j$. Also, the Euclidean length of Π_1^y is denoted by $|\Pi_1^y|$ and the Euclidean length of Π_w^y is denoted by $|\Pi_w^y|$.

Note that, by Observation 11, we can infer that Π_1 and Π_w are x -monotone (i.e., any vertical line intersects it at most once). Otherwise, we could reduce its length by a vertical shortcut, which is a contradiction. Therefore, Equation 6.3 holds.

$$|\Pi_1^x|^j = |\Pi_w^x|^j, j = 1, \dots, n \quad (6.3)$$

Note that some $|\Pi_1^x|^j$ (resp., $|\Pi_w^x|^j$) could be zero. More specifically, w.l.o.g., assume a is in k -th column and b is in ℓ -th column, with $k < \ell$. Then, $|\Pi_1^x|^j = |\Pi_w^x|^j = \text{width of the } j\text{-th column}$, $j = k + 1, \dots, \ell - 1$, and $|\Pi_1^x|^j = |\Pi_w^x|^j = 0$, $j = 1, \dots, k - 1, \ell + 1, \dots, n$.

By definition, $|\Pi_1|_w$ and $|\Pi_w|_w$ are given by Equation (6.4) and (6.5) respectively, where w_x^j is the weight of the x -axis in j -th column, and w_y^i is the weight of the y -axis in i -th row. In the remainder, we prove $|\Pi_1^y| = |\Pi_w^y|$, by contradiction. Combining this with Equation (6.3) proves the lemma.

$$|\Pi_1|_w = |\Pi_1^y| \cdot w_y^i + \sum_{j=1}^n |\Pi_1^x|^j \cdot w_x^j \quad (6.4)$$

$$|\Pi_w|_w = |\Pi_w^y| \cdot w_y^i + \sum_{j=1}^n |\Pi_w^x|^j \cdot w_x^j \quad (6.5)$$

Suppose $|\Pi_1^y| < |\Pi_w^y|$. Combining this with Equation (6.3) follows that $|\Pi_1|_w < |\Pi_w|_w$. Since we assumed Π_w has the minimum weighted length, from a to b , in $W_{\mathcal{R}_i}$, it is a contradiction. Now, assume $|\Pi_1^y| > |\Pi_w^y|$. Combining this with Equation (6.3) leads to $|\Pi_1|_1 > |\Pi_w|_1$, where $|\cdot|_1$ is the L_1 length of the path. This is a contradiction, because we assumed Π_1 is the L_1 shortest path, from a to b , in $W_{\mathcal{R}_i}$. Therefore, $|\Pi_1^y| = |\Pi_w^y|$ which proves $|\Pi_1|_w = |\Pi_w|_w$. $\square$

As a corollary to Lemma 30, it suffices to connect every two vertices $u, v \in V$ that are in

$W_{\mathcal{R}_i}$, by a L_1 shortest path from u to v . This provides a path with minimum weighted length, from u to v , in $W_{\mathcal{R}_i}$. Now, we explain how to construct the new graph $\mathcal{G}_w'' = \langle V'', E'' \rangle$. Let $V_{\mathcal{R}_i}$ be a subset of V that contains all vertices in $W_{\mathcal{R}_i}$ and $V_{\mathcal{C}_j}$ be a subset of V that contains all vertices in $W_{\mathcal{C}_j}$. We use the proposed construction in [73] to build a graph, $G_{\mathcal{R}_i} = \langle S_{\mathcal{R}_i}, E_{\mathcal{R}_i} \rangle$ for each $W_{\mathcal{R}_i}$, and a graph $G_{\mathcal{C}_j} = \langle S_{\mathcal{C}_j}, E_{\mathcal{C}_j} \rangle$ for each $W_{\mathcal{C}_j}$. Note that $V_{\mathcal{R}_i} \subset S_{\mathcal{R}_i}$ and for any two vertices in $V_{\mathcal{R}_i}$ there is a L_1 shortest path between them in $G_{\mathcal{R}_i}$. Also, $V_{\mathcal{C}_j} \subset S_{\mathcal{C}_j}$ and for any two vertices in $V_{\mathcal{C}_j}$ there is a L_1 shortest path between them in $G_{\mathcal{C}_j}$. At final step, we define V'' and E'' as follows:

$$V'' = \left(\bigcup_i S_{\mathcal{R}_i} \right) \cup \left(\bigcup_j S_{\mathcal{C}_j} \right) \quad (6.6)$$

$$E'' = \left(\bigcup_i E_{\mathcal{R}_i} \right) \cup \left(\bigcup_j E_{\mathcal{C}_j} \right) \quad (6.7)$$

Theorem 13. *Assume we are given two polygonal curves, T_1 and T_2 , and a leash length, ε . Each segment of T_1 and T_2 has an associated weight, corresponding to the backward movement on that segment. A pair of parameterizations of T_1 and T_2 that minimizes the weighted sum of the backward movements during the walks can be found in $\mathcal{O}(n^2 \log^{3/2} n)$ time and space complexity, where n is the number of segments in the input polygonal curves.*

Proof. The correctness follows directly from lemmas 29 and 30. To establish the time and space complexities, we first recall that $|V_{\mathcal{R}_i}| = \mathcal{O}(n)$. Therefore, $|S_{\mathcal{R}_i}| = \mathcal{O}(n \log^{1/2} n)$ and $|E_{\mathcal{R}_i}| = \mathcal{O}(n \log^{3/2} n)$ and the time complexity of the construction of $G_{\mathcal{R}_i}$ is $\mathcal{O}(n \log^{3/2} n)$ (see [73]). Since we have a linear number of rows and columns, $|V''| = \mathcal{O}(n^2 \log^{1/2} n)$ and $|E''| = \mathcal{O}(n^2 \log^{3/2} n)$, and the time complexity of the construction of $\mathcal{G}_w''$ is $\mathcal{O}(n^2 \log^{3/2} n)$. Using Dijkstra's algorithm, we can find a shortest path in $\mathcal{G}_w''$, from $\mathfrak{s}$ to $\mathfrak{t}$, in $\mathcal{O}(n^2 \log^{3/2} n)$ time. Therefore, the total time and space complexities of the proposed algorithm are $\mathcal{O}(n^2 \log^{3/2} n)$. $\square$

6.5 Conclusion

In this chapter, we generalized the MBFD problem to WMBFD so that one can capture weighted scenarios. We established that this problem corresponds to a weighted shortest path problem in a weighted deformed free-space diagram, $\mathcal{F}$. As a first approach, we showed that there is a path in the weighted visibility graph of the free-space that realizes an optimal pair of walks for the WMBFD problem. This led to an algorithm to solve the problem in polynomial time and space. Then, as an improvement, we showed that it is not necessary to construct the entire weighted visibility graph and we proposed an exact algorithm to solve the problem in $\mathcal{O}(n^3)$ time and space complexity. Then, we used Steiner points to reduce the size of the graph further that led to an exact algorithm with $\mathcal{O}(n^2 \log^{3/2} n)$ time and space complexity.

In the WMBFD problem, each edge of the input polygonal curves has an associated non-negative weight for backward movement. Now, consider a version of the problem that both moving forwards and backwards have weights. Then, the cost of moving on an edge is the Euclidean length of movement on that edge multiplied by the corresponding weight, which is a function of the movement direction (either forward or backward). The objective is to find a pair of walks that minimizes the sum of the costs on the edges of the curves, while guaranteeing that the leash length is at most ε . In this version, the total cost, for each polygonal curve T , is the summation of three values: (1) the total cost of moving forwards from the beginning to the end of T , (2) the total cost of moving backwards on T , (3) the total cost of moving forwards for covering the distance that was traveled backwards (and was counted in (2)). For any pair of walks, the value (1) is fixed. Therefore, this new problem can be straightforwardly reduced to the WMBFD problem, in linear time, as follows: for each edge of the input polygonal curves, set the weight of backtracking on that edge to the sum of its corresponding forward and backward weights. Then, solve the WMBFD problem. The solution of the WMBFD problem will be the solution to the new version of the problem.

Chapter 7

Minimizing Walking Length in Map Matching

7.1 Introduction

Trajectory data are often obtained from global positioning system (GPS) devices. Such devices have accuracy limitations due to noise, sampling intervals, or poor signals (e.g., inside buildings) thus raw spatial trajectories tend not to be accurate. Under the assumption that the travel captured by the trajectory was following edges of a map (stored as a graph) the map matching problem arises. It asks to find a path on the map that “corresponds well” to the given trajectory. Map matching arises in different contexts and is a necessary step in preprocessing raw data before data mining [17]. A variety of approaches have been used to solve the map matching problem (e.g. geometric, probabilistic methods, fuzzy logic, neural networks). In [132], Chen et al. discussed recent map matching algorithms when a trajectory is obtained from low-frequency GPS data of vehicles driving on a road network. Ruan et al. [133] studied indoor map matching technology based on personal motion states. In [134], Asakura et al. proposed a pedestrian-oriented map matching algorithm in the context of disasters. In this context, refugees have battery-driven mobile GPS terminals and move to shelters at walking speed. They stated that in order to reduce battery consumption (which is vital in this context), they chose a geometric approach in which computation resources are less utilized when compared e.g., with probabilistic methods.

In this chapter, we focus on geometric approaches. We assume that a map is given as a planar graph via a straight-line embedding in a plane. Therefore, a path in the graph corresponds to a polygonal curve in the plane. A trajectory is given as a directed polygonal

curve from a starting point to an ending point. The objective is to find a path in the map which is most “similar” to the given trajectory.

In [135], Alt et al. discussed the map matching problem set in the context of the standard Fréchet distance. Their algorithm finds a path in the planar graph with minimum Fréchet distance to the given trajectory. The time complexity of their algorithm is $\mathcal{O}(mn \log^2 mn)$ where m is the number of edges in the input planar graph and n is the number of edges in the input polygonal curve. Brakatsoulas et al. [18], extended the map matching algorithm of [135] for the weak Fréchet distance. The time complexity of their algorithm is $\mathcal{O}(mn \log mn)$. In [136], Chen et al. proposed a $(1 + \varepsilon)$ -approximation algorithm for the map matching problem when the similarity measure is the standard Fréchet distance and input model is more “realistic”. They assumed that the input polygonal curve is c -packed and the input graph is ϕ -low density in $\mathbb{R}^d$.

In this chapter, we study the map matching problem when the similarity measure is the MBFD. More specifically, as input, we are given: a planar graph, H , with a straight-line embedding in a plane, a directed polygonal curve, T , and a distance $\varepsilon > 0$. Both forward and backward motions along T and the edges of H are allowed. The objective is to find a path, P , in H , and a parameterization of T , that minimize the sum of the walk lengths along T and P while keeping a leash length of at most ε . We restrict the start and end point of P to be at a vertex of H . However, P may partially contain an edge of H . The difference between this problem setting and that optimization setting of [135] and [18], is that here the total walking length along T and P is minimized while in the other settings the leash length is minimized. The optimization problem introduced in this chapter, can also be used to track objects moving on road networks. To ensure high-quality tracking, the mobile tracker must remain within a distance of ε to the moving object, at all time. To minimize energy consumption, the tracker wants to minimize walking distance. This type of scenario has been discussed in the context of wireless networks (see [137] and [138]).

Figure 7.1 shows an example of an embedding of a planar graph, H , in $\mathbb{R}^2$, a polygonal curve, T , and a length ε . The dog walks on T from $T(0)$ to $T(4)$ and the person chooses a path in H , from one vertex of H to another vertex. Two points, a and b , are determined on $\langle v_4, v_5 \rangle$ and $\langle v_3, v_4 \rangle$ respectively. A path in H , and a walk on T , that minimize the walking lengths on H and T , are as follows: the dog starts at $T(0)$ and continues on T . The person starts at v_1 and walks on the edges $\langle v_1, v_3 \rangle$, $\langle v_3, v_4 \rangle$, and $\langle v_4, v_5 \rangle$. They move together in a forward direction until the dog reaches the end of the second segment of T and the person reaches the point a on $\langle v_4, v_5 \rangle$. Then, the dog continues to move forwards until the end of the third segment of T is reached, while the person moves backwards from a to v_4 and then to b . At the final step, they move forwards again together until the dog reaches the end of T and the person reaches v_5 . We show the path in the graph by the sequence of its vertices, $P^* = [v_1, v_3, v_4, a, v_4, b, v_4, v_5]$.

Figure 7.1: An embedding of a planar graph, H , a polygonal curve, T , and a length ε are given. The path $P^* = [v_1, v_3, v_4, a, v_4, b, v_4, v_5]$, in H , is a part of a solution to the map matching problem instance. The edges of H that P^* lies on, are illustrated in bold.

The structure of this chapter is as follows. In Section 7.2, we discuss preliminaries and define the problem formally. In Section 7.3, we propose a polynomial time algorithm for

the map matching problem introduced. Then, in Section 7.4, we develop an algorithm with improved time and space complexities. In Section 7.5, we sketch a solution to a weighted problem variant. Finally, in Section 7.6, we conclude the chapter.

7.2 Preliminaries and Definitions

Let $T : [0, n] \rightarrow \mathbb{R}^2$ be a polygonal curve with n segments. A vertex of T is denoted by $T(i)$, $i = 0, \dots, n$. Let $H = \langle V_H, E_H \rangle$ be a planar graph with a straight-line embedding in $\mathbb{R}^2$ where V_H (E_H , respectively) is the set of vertices (edges, respectively) of H . In this chapter, the geometric embedding of H is crucial and we simply refer to the straight-line embedding of the graph in $\mathbb{R}^2$ as H . A path, P , in H , is a polygonal curve $P : [0, 1] \rightarrow H$, such that $P \subset H$ and $P(0), P(1) \in V_H$.

Walking Length. Let $f : [0, 1] \rightarrow [0, n]$ be a parameterization of T and $\mathcal{B}_f \subseteq [0, 1]$ be the closure of the set of times in which $f(t)$ is decreasing (i.e., the movement is backward). The walking length of T is defined by (7.1), where $\|\cdot\|$ is the Euclidean length.

$$\mathcal{L}_f(T) := \|T\| + 2 \int_{t \in \mathcal{B}_f} \|(T(f(t)))'\| dt \quad (7.1)$$

Note that if f is monotone (i.e., there is no backward movements on T), then $\mathcal{L}_f(T) = \|T\|$.

Problem Definition. Suppose H , T and a length, $\varepsilon > 0$, are given. The objective is to find a path in H and a parameterization of T such that sum of the length of P and the walking length of T is minimized (see (7.2)). We consider only paths in the graph and parameterizations of T that guarantee to maintain the leash length at most ε , during the walks.

$$\mathcal{M}^\varepsilon(H, T) := \inf_{P \subset H, f} \{\|P\| + \mathcal{L}_f(T)\} \quad (7.2)$$

Deformed free-space surface. In [135], Alt et al. introduced a 3D structure, called free-space surface, to solve the decision version of their map matching problem. Here, we utilize this structure to explain our algorithm. However, we modify it slightly, to fit our problem setting.

Let P be a path in H with $k + 1$ vertices, $[p_0, p_1, \dots, p_k]$. Consider the deformed free-space diagram, for T and P (see Section 4.1.1). We denote it by $\mathcal{F}_\varepsilon(T, P)$. It consists of nk parameter cells $C^{x,y}$, for $x = 1, \dots, n$ and $y = 1, \dots, k$. Cell $C^{x,y}$ is the result of the product of two sub-intervals of $[0, 1]$ that are mapped to edge $\overrightarrow{T(x-1)T(x)}$ of T and edge $\overrightarrow{p_{y-1}p_y}$ of P , respectively. We denote the free-space of $\mathcal{F}_\varepsilon(T, P)$ by W_P and the forbidden-space of $\mathcal{F}_\varepsilon(T, P)$ by B_P . In Figure 7.2, the free-space diagram $\mathcal{F}_\varepsilon(T, P^*)$ is drawn, where $P^* = [v_1, v_3, v_4, a, v_4, b, v_4, v_5]$ is a path in H , denoted as a sequence of its vertices.

Figure 7.2: The free-space diagram $\mathcal{F}_\varepsilon(T, P^*)$ is drawn. W_P is the white area and B_P is the gray area.

Note that if the path P contains only a single vertex of H , $v_i \in V_H$, then $\mathcal{F}_\varepsilon(T, P)$ is a line segment and its length is equal to the Euclidean length of T . We call this 1D free-space diagram, $\mathcal{F}_i$, the *deformed free-space line* of v_i . We denote the left endpoint of $\mathcal{F}_i$ (i.e.,

the endpoint corresponding to $T(0)$ by s_i and the right endpoint of $\mathcal{F}_i$ (i.e., the endpoint corresponding to $T(n)$) by t_i . If P contains only an edge, $\langle v_i, v_j \rangle \in E_H$, of H , then $\mathcal{F}_\varepsilon(T, P)$ has only one row. We call this row the *deformed free-space face* of $\langle v_i, v_j \rangle$, and denote it by $\mathcal{F}_i^j$.

Note that $\mathcal{F}_i^j$ and $\mathcal{F}_j^k$ have $\mathcal{F}_j$ in common. Therefore, gluing $\mathcal{F}_i^j$ and $\mathcal{F}_j^k$ along $\mathcal{F}_j$ produces a conforming surface. Thus, we can construct the *deformed free-space surface* as follows. We first lay out the straight-line embedding of H in the xy -plane. For each edge $\langle v_i, v_j \rangle \in E_H$, we lay out $\mathcal{F}_i^j$, orthogonal to the xy -plane, along z axis, such that $\mathcal{F}_i$ ($\mathcal{F}_j$, respectively) is on top of v_i (v_j , respectively) and s_i (s_j , respectively) is in the xy -plane. Note that $\mathcal{F}_i^j$ is stretched along z axis from the plane $z = 0$ to the plane $z = \|T\|$. Suppose $\text{Adj}(v_j)$ is the set of all vertices $v_k \in V_H$ such that $\langle v_j, v_k \rangle \in E_H$. We glue $\mathcal{F}_i^j$ to $\mathcal{F}_j^k$ along $\mathcal{F}_j$, where $v_k \in \text{Adj}(v_j)$. Also, we glue $\mathcal{F}_i^j$ to $\mathcal{F}_h^i$ along $\mathcal{F}_i$, where $v_i \in \text{Adj}(v_h)$. The result is a conforming 3D surface between two planes, $z = 0$ and $z = \|T\|$, called deformed free-space surface and is denoted by $\mathcal{S} = H \times [0, \|T\|]$. Note that s_i is on the plane $z = 0$ and t_i is on the plane $z = \|T\|$, $i = 1, \dots, |V_H|$. The union of the white-space of all faces of $\mathcal{S}$ is called the *white-surface* and is denoted by $\mathcal{W}$. Analogously, the union of the black-space of all faces of $\mathcal{S}$ is called the *black-surface* and is denoted by $\mathcal{B}$. For the given planar graph H , the polygonal curve T , and the length ε in Figure 7.1, the corresponding deformed free-space surface is shown in Figure 7.3, from two points of views. Since the white-space of each cell of any $\mathcal{F}_i^j$ is convex, for simplicity, we just draw the white-space intervals on the boundary of the cells. In this figure, the red dashed polygonal curve is a path on the white-surface $\mathcal{W}$, from s_1 to t_5 , that realizes $P^* = [v_1, v_3, v_4, a, v_4, b, v_4, v_5]$, in H , and a parameterization of T , that is an optimal solution to our problem setting. It intersects the following free-space faces sequentially: $\mathcal{F}_1^3$, $\mathcal{F}_3^4$, $\mathcal{F}_4^5$, $\mathcal{F}_3^4$, $\mathcal{F}_4^5$.

Figure 7.3: The free-space surface for the example of Figure 7.1 is drawn from two different viewpoints in $3D$. The yellow line segments show the intervals on the cell boundaries. The red dashed polygonal curve is a path on the white-surface that realizes an optimal solution to our problem setting.

7.3 Algorithm

In this section, we first transform the map matching problem to a shortest path problem on a weighted graph, $\mathcal{G} = \langle V, E \rangle$; this yields a polynomial time algorithm. Before discussing the construction of $\mathcal{G}$, we introduce a set of Steiner points on the boundary of the cells of $\mathcal{S}$.

Steiner Points. We position Steiner points so as to create intervals on the boundary of the cells of $\mathcal{S}$. There are two types of intervals, Type 1 and Type 2. We classify the Steiner points based on the type of the intervals that they belong to. We denote the set of Type 1 (Type 2, respectively) Steiner points by S_1 (S_2 , respectively).

Type 1. We say an interval is Type 1, if it lies completely in a plane, $z = c$, parallel to the xy -plane, where c is a constant. Each deformed free-space face, $\mathcal{F}_i^j$, may have $n + 1$ Type 1 intervals, $\mathcal{IT}_i^j(\ell)$, $\ell = 0, \dots, n$, shared between its cells (where n is the number of edges in T). For each interval $\mathcal{IT}_i^j(\ell)$, we project the endpoints of $\mathcal{IT}_i^j(\ell)$ orthogonally to all $\mathcal{IT}_i^j(k)$, $k \neq \ell$. If the line segment from an endpoint of $\mathcal{IT}_i^j(\ell)$ to its projection on $\mathcal{IT}_i^j(k)$ lies in the free-space of $\mathcal{F}_i^j$ and the projection point is not identical with an endpoint of $\mathcal{IT}_i^j(k)$, then we take the projection point as a Type 1 Steiner point (see Figure 7.4). The set of all Steiner points, obtained by the projections on $\mathcal{F}_i^j$, for all $\langle v_i, v_j \rangle \in E_H$, is denoted by S_1 .

Figure 7.4: The free-space face $\mathcal{F}_i^j$ is drawn. The endpoints of the intervals, $\mathcal{IT}_i^j(\ell)$, are shown by black points and the Type 1 Steiner points are shown by squares.

Type 2. We say an interval is Type 2 if it lies completely on a deformed free-space line. As we mentioned in Section 7.2, a plane $z = c$ corresponds to a point on the given trajectory T . The intersection of $z = c$ and $\mathcal{S}$ is an instance of H , denoted by H_c . Note that some part (possibly empty) of H_c is in $\mathcal{W}$. Let $z = h_j$ be the corresponding plane of $T(j)$, a vertex of T . The part of the deformed free-space surface, $\mathcal{S}$, between the two parallel planes, $z = h_{j-1}$

and $z = h_j$, $j = 1, \dots, n$, corresponds to edge $\overrightarrow{T(j-1)T(j)}$ of T . We denote this part of $\mathcal{S}$ by $\mathcal{T}_{j-1}^j = H \times [h_{j-1}h_j]$. In $\mathcal{T}_{j-1}^j$, $j = 1, \dots, n$, there is at most one Type 2 interval per vertex $v_i \in V_H$. We denote these Type 2 intervals by $\mathcal{TI}_{j-1}^j(i)$, $i = 1, \dots, |V_H|$. Suppose $z = c$ is the plane that is passing through an endpoint, p , of $\mathcal{TI}_{j-1}^j(i)$. Let the intersection of $z = c$ with $\mathcal{TI}_{j-1}^j(k)$, $k \neq i$, be q_k . Note that both p and q_k are on the graph H_c . Then, if q_k is not an endpoint of $\mathcal{TI}_{j-1}^j(k)$ and there is a path, from p to q_k , in H_c , that is in $\mathcal{W}$, then q_k is a Type 2 Steiner Point. The set of all Type 2 Steiner points is denoted by S_2 . An example is given in Figure 7.5. Suppose it is $\mathcal{T}_{j-1}^j$, for $j = 1$. In this example, there are four yellow intervals, $\mathcal{TI}_{j-1}^j(1)$, $\mathcal{TI}_{j-1}^j(3)$, $\mathcal{TI}_{j-1}^j(5)$, and $\mathcal{TI}_{j-1}^j(6)$. The black points show the interval endpoints and red balls show the Type 2 Steiner points. For simplicity, only two, out of eight planes, are drawn. The plane z_3 is passing through an endpoint of $\mathcal{TI}_{j-1}^j(3)$. Also, the plane z_6 is passing through an endpoint of $\mathcal{TI}_{j-1}^j(6)$. The intersections of z_3 with $\mathcal{TI}_{j-1}^j(5)$ and $\mathcal{TI}_{j-1}^j(6)$ are Steiner points. However, the intersection of z_6 with $\mathcal{TI}_{j-1}^j(3)$ is not a Steiner point.

Constructing Graph. Now, we explain the construction of $\mathcal{G} = \langle V, E \rangle$. Recall that the white-surface (the white-space of $\mathcal{S}$) is denoted by $\mathcal{W}$. The vertices of $\mathcal{W}$ are the end points of the intervals on the boundary of the cells in $\mathcal{S}$ (at most 4 intervals may exist per cell). We denote the set of vertices of $\mathcal{W}$ by $V_{\mathcal{W}}$. The set of vertices, V , of $\mathcal{G}$, is $V = V_{\mathcal{W}} \cup S_1 \cup S_2$. Note that V contains all s_i and t_i if they are in $\mathcal{W}$. Every two vertices, $v_1, v_2 \in V$, that are on the boundary of a cell, are linked by two directed edges in E , from v_1 to v_2 , $\langle v_1, v_2 \rangle$, and vice versa, $\langle v_2, v_1 \rangle$. The weight of an edge $e = \langle v_1, v_2 \rangle \in E$, is its length in the L_1 metric, $|e|_1$.

Obtain an Optimal Solution. In order to have an optimal solution, at least one s_i and one t_j , $i, j = 1, \dots, |V_H|$, must be in $\mathcal{W}$. The main steps of the algorithm are as follows:

- Find all the vertices in V_H that are in ε distance of $T(0)$, $v_{i_1}, \dots, v_{i_{k_1}}$, and in ε distance of $T(n)$, $v_{j_1}, \dots, v_{j_{k_2}}$.

$\mathcal{S}$, from one s_i to one t_j . Π realizes a path, $P : [0, 1] \rightarrow H$, in H , and a parameterization, $f : [0, 1] \rightarrow [0, n]$, of T , that maintain the leash length at most ε , for all $t \in [0, 1]$.

Constructing a path in H . We can construct a path P in H , from the given path Π in $\mathcal{W}$, as follows. As we mentioned earlier in this section, we have two types of intervals on the boundary of the cells, Type 1 and Type 2. A Type 1 interval lies completely on a plane, $z = c$, parallel to the xy -plane. A Type 2 interval lies completely on a deformed free-space line, $\mathcal{F}_i$. The path Π intersects a sequence of intervals (of both types). The path P in H is constructed by processing the intervals in this sequence. For each interval in this sequence, if it is Type 2 interval, on $\mathcal{F}_i$, then we append v_i to the tail of P . If it is Type 1, then the intersection point, q , of Π and that interval, is appended to the tail of P , as a vertex of P . Note that q may not be a vertex of H . However, it is a point on an edge of H . At the end, we connect the consecutive vertices in P by straight line segments.

Correctness. To establish the correctness, we use norms in two spaces: (1) the Euclidean space of the embedding of the input graph and the polygonal curve, called the input space, (2) the deformed free-space surface, called the configuration space. In the input space, we denote the Euclidean length of a polygonal curve T by $\|T\|$. We also defined walking length of T , $\mathcal{L}_f(T)$, based on a parameterization f . Note that if f is a monotone parameterization, then $\mathcal{L}_f(T) = \|T\|$. In configuration space, a path from an s_i to a t_j in $\mathcal{W}$, is also denoted by its vertices, $\Pi : \langle s_i = p_1, p_2, \dots, p_k = t_j \rangle$. The length, $|\cdot|_1$, of each segment of Π is calculated by the L_1 metric. The length of a path, $|\Pi|_1$, is the sum of the length of its segments.

This section is concluded by a corollary to Lemma 31 and Observation 12, that is, in order to find a solution for our problem setting, it suffices to find a shortest path from s' to t' in $\mathcal{G}$.

Lemma 31. *For any path $\Pi : \langle s_i = p_1, p_2, \dots, p_{k_1} = t_j \rangle$ in $\mathcal{W}$, there is a path $\Pi' : \langle s_i = p'_1, p'_2, \dots, p'_{k_2} = t_j \rangle$ in $\mathcal{W}$ such that $\Pi' \subset \mathcal{G} = \langle V, E \rangle$ and $|\Pi'|_1 \leq |\Pi|_1$.*

Proof. The path Π intersects a sequence, SF , of deformed free-space faces. Every two consecutive faces in SF share a deformed free-space line. Therefore, we can unfold the free-space faces in the sequence, along the shared free-space lines. The result is a 2D free-space diagram, denoted by $\mathcal{F}_\varepsilon(SF)$. The path Π is also unfolded into a 2D path in the white-space, $\mathcal{W}_{SF}$, of $\mathcal{F}_\varepsilon(SF)$. Note that unfolding does not change the length of a path. As an example, in Figure 7.6a, the result of unfolding the faces that are intersected by the red dashed polygonal curve in Figure 7.3, is shown.

Figure 7.6: a) The result of unfolding the sequence of deformed free-space faces that are intersected by the red dashed polygonal curve in Figure 7.3. It is a 2D free-space diagram, $\mathcal{F}_\varepsilon(SF)$. The red dashed polygonal curve is shown after unfolding. b) Illustration of case 1 in the proof of Lemma 31.

Let $\Pi_{opt} : \langle s_i = q_1, q_2, \dots, q_{k_3} = t_j \rangle$ be a L_1 shortest path, from s_i to t_j , in $\mathcal{W}_{SF}$. Then, $|\Pi_{opt}|_1 \leq |\Pi|_1$. To prove the lemma, it suffices to show that there is a path $\Pi' \subset \mathcal{G}$, from s_i to t_j , in $\mathcal{W}_{SF}$, such that $|\Pi'|_1 = |\Pi_{opt}|_1$.

We know that the vertices of Π_{opt} are endpoints of some intervals on the boundary of the cells of $\mathcal{F}_\varepsilon(SF)$ [123] (the well known rubber band property of shortest paths). Therefore, the vertices of Π_{opt} are in V (i.e., the set of vertices of $\mathcal{G}$). Thus, it is sufficient to show that for

each edge, $\overrightarrow{q_i q_{i+1}}$, of Π_{opt} , there is a path, $\pi_{q_i q_{i+1}}$, from q_i to q_{i+1} , in $\mathcal{G}$, that $|\overrightarrow{q_i q_{i+1}}|_1 = |\pi_{q_i q_{i+1}}|_1$. Two cases arise depending on whether $\overrightarrow{q_i q_{i+1}}$ lies completely within a row (or a column) of $\mathcal{F}_\varepsilon(SF)$, or not. We need a definition before discussing these cases. We assume that $\mathcal{F}_\varepsilon(SF)$ is an axis-aligned rectangle in a 2D Cartesian coordinate system, where the x -axis corresponds to T (Figure 7.6a). We say a path $\Pi \in \mathcal{W}_{SF}$ is x -monotone (y -monotone, respectively), if any vertical (horizontal, respectively) line intersects it at most ones. Π is said to be xy -monotone, if it is both x - and y -monotone.

Case 1. In this case, $\overrightarrow{q_i q_{i+1}}$ lies completely within a column (or a row) of $\mathcal{F}_\varepsilon(SF)$. Here, we discuss the case when it lies within a column (see Figure 7.6b); the arguments are analogous for the case of a row (we just need to apply a rotation of 90 degrees). W.l.o.g we assume that $\overrightarrow{q_i q_{i+1}}$ is xy -increasing. The other cases are symmetric. Edge $\overrightarrow{q_i q_{i+1}}$ intersects a sequence of horizontal intervals, I_z , within a column. They are sorted based on their y coordinates. We construct $\pi_{q_i q_{i+1}}$ sequentially and always denote the last vertex appended to $\pi_{q_i q_{i+1}}$ by π_{last} . Initially, $\pi_{q_i q_{i+1}}$ contains only q_i and $\pi_{last} = q_i$. The sequence of intervals are processed sequentially. Suppose we processed interval I_z and now we want to process I_{z+1} . We project orthogonally from π_{last} to I_{z+1} . If the projection point exists (i.e., the perpendicular line from π_{last} to I_{z+1} intersects I_{z+1}), then append the projection point on I_{z+1} to $\pi_{q_i q_{i+1}}$ and update π_{last} . Otherwise, the closest endpoint of I_{z+1} to π_{last} is appended to $\pi_{q_i q_{i+1}}$ and we update π_{last} . When all intervals have been processed, q_{i+1} is appended to $\pi_{q_i q_{i+1}}$.

Since the sorted list of intervals within a column are traversed by $\pi_{q_i q_{i+1}}$ sequentially, the path $\pi_{q_i q_{i+1}}$ is y -monotone. Also, by construction, each vertex of $\pi_{q_i q_{i+1}}$ either has the same x as its preceding vertex in $\pi_{q_i q_{i+1}}$ (i.e., it is the result of the orthogonal projection) or its x is greater than its preceding vertex's x (since the orthogonal projection does not exist and $\overrightarrow{q_i q_{i+1}}$ is xy -increasing inside the white-space). Therefore, the path $\pi_{q_i q_{i+1}}$ is x -monotone. Thus, the path $\pi_{q_i q_{i+1}}$ is xy -monotone. We know that the L_1 length of two xy -monotone paths that

have the same starting and ending points, are equal. Therefore, $|\overrightarrow{q_i q_{i+1}}|_1 = |\pi_{q_i q_{i+1}}|_1$.

Now, we prove that $\pi_{q_i q_{i+1}} \subset \mathcal{G} = \langle V, E \rangle$. It suffices to show that each vertex of $\pi_{q_i q_{i+1}}$ is in V and between every two consecutive vertices of $\pi_{q_i q_{i+1}}$ there is an edge in E . Each vertex of $\pi_{q_i q_{i+1}}$ is either the result of the orthogonal projection or an endpoint of an interval. Therefore, each vertex is either a Steiner point or a vertex of the white-surface. In both cases, the vertex is in V . In addition, between every two consecutive vertices of $\pi_{q_i q_{i+1}}$ there is an edge in E because every two consecutive vertices of $\pi_{q_i q_{i+1}}$ lie on the boundary of a cell and, by the construction of $\mathcal{G}$, all members of V that lie on the boundary of a cell are linked by edges in E .

Case 2. In the proof of Lemma 29, in Chapter 6, we showed that if $\overrightarrow{q_i q_{i+1}}$ does not lie completely within a row and within a column of $\mathcal{F}_\varepsilon(SF)$, then there is a xy -monotone path $\pi'_{q_i q_{i+1}}$, from q_i to q_{i+1} , such that its edges lie completely within a row or within a column of $\mathcal{F}_\varepsilon(SF)$. For each edge of $\pi'_{q_i q_{i+1}}$, we apply Case 1. Then, we concatenate the resulting xy -monotone paths for edges of $\pi'_{q_i q_{i+1}}$, to obtain $\pi_{q_i q_{i+1}}$. Since, xy -monotone paths for edges of $\pi'_{q_i q_{i+1}}$ are in $\mathcal{G}$ (as we proved in Case 1), the resulting path, $\pi_{q_i q_{i+1}}$, is a xy -monotone path in $\mathcal{G}$. Therefore, $|\overrightarrow{q_i q_{i+1}}|_1 = |\pi_{q_i q_{i+1}}|_1$. $\square$

Corollary 11. *For any pair of s_i and t_j , if t_j is reachable from s_i by a path in $\mathcal{W}$, then there is a path from s_i to t_j , in $\mathcal{G}$, that is a L_1 shortest path in $\mathcal{W}$.*

Corollary 12. *A shortest path in $\mathcal{G}$, from s' to t' , yields an optimal solution for our problem setting.*

Proof. Let Π'_{opt} be a shortest path in $\mathcal{G}$, from s' to t' . We remove s' and t' from the head and tail of Π'_{opt} . The result, Π_{ij} , is a shortest path from s_i to t_j . Therefore, among all possible shortest paths $\Pi_{k\ell}$, for s_k and t_ℓ , $k, \ell = 1, \dots, |V_H|$, the pair (s_i, t_j) has a shortest L_1 shortest path, Π_{ij} . By Corollary 11, $\Pi_{ij} \subset \mathcal{G}$ is a L_1 shortest path in $\mathcal{W}$. Each point on Π_{ij} is corresponding to a point, p , on H and a point, q , on T , such that the Euclidean distance

of p and q is less than ε . By Observation 12, Π_{ij} , is corresponding to a path, P , in H , from $v_i \in V_H$ to $v_j \in V_H$, and a parameterization, f , of T . The summation of the Euclidean length of P , $\|P\|$, and the walking length of T , $\mathcal{L}_f(T)$, is equal to the L_1 length of Π_{ij} . Since Π_{ij} is a shortest among all possible shortest paths $\Pi_{k\ell}$, P and f minimize the matching cost, $\mathcal{M}^\varepsilon(H, T)$ (Equation 7.2). $\square$

Theorem 14. *Let H be a planar graph with a straight-line embedding in a plane, T be a directed polygonal curve, and $\varepsilon > 0$ be a distance. A path, $P : [0, 1] \rightarrow H$, between two vertices of H , and the parameterization, f , of T , that minimizes the sum of the walking length of T and P , can be found in polynomial time and space. It is guaranteed that at any time $t \in [0, 1]$, the Euclidean distance between $P(t)$ and $T(f(t))$ is at most ε .*

Proof. The correctness follows directly from Corollary 12. The deformed free-space surface, $\mathcal{S}$, has $\mathcal{O}(mn)$ cells, where m (n , respectively) is the number of edges of H (T , respectively). Each cell of $\mathcal{S}$ has at most four intervals and at most $\mathcal{O}(m+n)$ Steiner points on its intervals. Therefore, the graph $\mathcal{G}$ has $\mathcal{O}(mn(m+n))$ vertices and $\mathcal{O}(mn(m+n)^2)$ edges (including the extra edges that connect s' and t' to the graph). In addition, it takes $\mathcal{O}(n^2)$ time to compute all Type 1 Steiner points for each free-space face. Therefore, computing S_1 takes $\mathcal{O}(mn^2)$ time. In order to compute Type 2 Steiner points, we use breadth first search for each interval endpoint to propagate the projection on the instance of the graph, H_c , in the plane $z = c$. Therefore, computing S_2 takes $\mathcal{O}(nm^2)$ time. At the end, it is possible to find a shortest path in $\mathcal{G}$, from s' to t' , in $\mathcal{O}(mn(m+n)^2)$ time, by using Dijkstra's algorithm. Therefore, both the total time and space complexities are $\mathcal{O}(mn(m+n)^2)$. $\square$

7.4 Improvement

In Section 7.3, we showed that the graph $\mathcal{G} = \langle V, E \rangle$ contains a path that yields an optimal solution for our problem setting. The bottleneck in the time complexity of the algorithm in

Section 7.3 is due to the number of edges of $\mathcal{G}$. In this section, we construct a new graph $\mathcal{G}' = \langle V, E' \rangle$, such that $|E'| < |E|$ and it preserves the connectivity information of $\mathcal{G}$. More precisely, if there is a path, from $v_i \in V$ to $v_j \in V$, in $\mathcal{G}$, then there is a path, from v_i to v_j , in $\mathcal{G}'$, with the same L_1 length.

Based on the construction of $\mathcal{G}$, there are at most $\mathcal{O}(m + n)$ vertices in V (including the interval endpoints and Steiner points) on the boundary of each cell, C , of $\mathcal{S}$. We connect these $\mathcal{O}(m + n)$ vertices by a linear number of edges, in E' , as follows. The weight of each edge in E' is equal to its L_1 -length. Let T , B , L , and R be the intervals on the top, bottom, left and right side of C , respectively. Suppose cell C is in a 2D Cartesian coordinate system and the vertices on each interval $I \in \{T, B, L, R\}$ are sorted by x and y . Every two adjacent vertices, v_i and v_{i+1} , on I , are linked by two directed edges, $\langle v_i, v_{i+1} \rangle$ and $\langle v_{i+1}, v_i \rangle$ (Figure 7.7). Every two of the eight interval endpoints are linked by two directed edges assuming they are not identical. A vertex v_i on interval L (T , respectively), is linked by two directed edges to another vertex v'_i on R (B , respectively) if v'_i has the same y (x , respectively) coordinate as v_i ; the two edges are denoted by $\langle v_i, v'_i \rangle$ and $\langle v'_i, v_i \rangle$, respectively. By this approach, each vertex of $\mathcal{G}'$ on the boundary of C is connected to a constant number of vertices of $\mathcal{G}'$ on the boundary of C . It is now straightforward to prove the following lemma.

Lemma 32. *Let v_i and v_j be two vertices, in V , on the boundary of a cell, C , of $\mathcal{S}$. There is a path, from v_i to v_j , in $\mathcal{G}'$, that has the same L_1 length as the direct line segment between them.*

Proof. We prove this lemma by case analysis. There are three cases: (1) v_i and v_j lie on a same side of C ; (2) v_i and v_j lie on opposite sides of C ; (3) v_i and v_j lie on two adjacent sides of C . In each case, it suffices to show that there is a xy -monotone path from v_i to v_j in $\mathcal{G}'$.

Case (1). Since there is an edge between every two adjacent members of V , there is a path from v_i to v_j and it is on the straight line from v_i to v_j .

Figure 7.7: A cell of the free-space surface is drawn. The red solid line segments show the four intervals on the boundary of the cell. The arcs show the edges in E' that connect every two adjacent vertices of $\mathcal{G}'$, on each interval. The dashed black line segments show the edges in E' that connect a vertex with its orthogonal projection on the opposite side of the cell. The dash dotted blue line segments show some of the edges that connect endpoints of the intervals. For simplicity, we did not draw all of them.

Case (2). W.l.o.g. we assume that $v_i \in L$ and $v_j \in R$, where L is the interval on the left side of C and R is the interval on the right side of C . The other cases are symmetric. In this case, there are three sub-cases (see Figure 7.8):

- Either the orthogonal projection of v_i onto R , or the orthogonal projection of v_j onto L , exists. In this sub-case, by following the orthogonal projection we can construct a xy -monotone path from v_i to v_j , that proves the lemma.
- There is at least one endpoint of either L or R that has a y -coordinate between the y -coordinates of v_i and v_j and its orthogonal projection to the opposite side exists. In this sub-case, by following the orthogonal projection of the end point, we can construct a xy -monotone path from v_i to v_j , that proves the lemma.
- None of the previous cases hold. In this sub-case, by following the edge between the endpoints of L and R , we can construct a xy -monotone path from v_i to v_j , that proves the lemma.

Case (3). W.l.o.g. we assume that $v_i \in L$ and $v_j \in B$, where B is the interval on

Figure 7.8: The three sub-cases in the proof of Case (2), Lemma 32.

the bottom side of C . The other cases are symmetric. In this case we can construct a xy -monotone path from v_i to v_j as follows. This xy -monotone path is the result of the concatenation of three straight sub-paths: one from v_i to the endpoint, a , of L that is closer to B ; one from v_j to the endpoint, b , of B that is closer to L ; and the direct edge between the two endpoints, a and b .

□

Corollary 13. *There is a path in $\mathcal{G}'$ that realizes an optimal solution for our problem setting.*

Theorem 15. *Let H be a planar graph with a straight-line embedding in a plane, T be a directed polygonal curve, and $\varepsilon > 0$ be a distance. A path, $P : [0, 1] \rightarrow H$, between two vertices of H , and a parameterization, f , of T , that minimizes the sum of the walking length of T and P , can be found in $\mathcal{O}(nm(n+m)\log(nm))$ time and $\mathcal{O}(nm(n+m))$ space, where n (m , respectively) is the number of edges of T (H , respectively). It is guaranteed that at any time $t \in [0, 1]$, the Euclidean distance between $P(t)$ and $T(f(t))$ is at most ε .*

Proof. The correctness follows directly from Corollary 13. The number of vertices and edges of $\mathcal{G}'$ (and the total space complexity) is upper-bounded by $\mathcal{O}(nm(n+m))$. Using Dijkstra's algorithm, we find a shortest path in $\mathcal{G}'$, from s' to t' . Therefore, the time complexity of our algorithm is $\mathcal{O}(nm(n+m)\log(nm))$. Note that if there is no pair of (s_k, t_ℓ) , $k, \ell = 1, \dots, |V_H|$, in a connected component of $\mathcal{G}'$, then there is no feasible walk, on H and T , with a leash length of at most ε . Therefore, the algorithm halts with answer “no feasible solution”. □

7.5 Weighted Non-planar Graphs

In this chapter, we assumed that the input graph H is planar. That made the illustration of the algorithm easier since the faces of the free-space surface do not intersect except at the boundary of the faces. However, all lemmas and theorems, derived in Section 7.3 and 7.4, are proved without making an assumption regarding the planarity of H . Therefore, the algorithm proposed in this paper remains correct for any graph for which a straight-line embedding in a plane is provided (see [135], Section 2.7). In the embedding, the edges of the graph may intersect. Transition from one edge to another is allowed only at a vertex.

We also assumed that H is unweighted. Here, we sketch how the proposed algorithm can be generalized to also handle the problem instance, when H is weighted. Suppose that each edge of H has a non-negative, real weight. A weight could represent the cost of moving on the edge of the graph. The edges of the input polygonal curve T could also have weights capturing the costs of moving forwards and backwards. The objective is to find a path in H whose weighted walking length is minimized. In the weighted problem setting, inside each cell of the free-space surface, there are two weights, one corresponding to an edge of T and one corresponding to an edge of H . These weights are fixed inside the cell and do not change. Therefore, in the construction of $\mathcal{G}$ or $\mathcal{G}'$, instead of computing the L_1 length for each edge, e , we compute the orthogonal projections of e onto H and T . Then, we multiply the projection lengths with the corresponding weights, and the sum of these multiplications is the weight that we assign to e . The remaining parts of the algorithm remain the same and the time and space complexities do not change.

7.6 Conclusion

In this chapter, we discussed a geometric algorithm for the map matching problem that minimizes the walking length. We established that this problem setting is dual to a weighted

shortest path problem. Then, we proposed an algorithm with $\mathcal{O}(mn(m+n)\log(mn))$ time and $\mathcal{O}(mn(m+n))$ space complexity, where m (n , respectively) is the number of edges of H (T , respectively). At the end, we discussed that the proposed algorithm is easily adaptable to handle weighted non-planar graphs.

Note that the white-space, $\mathcal{W}$, of a deformed free-space surface, $\mathcal{S}$, is a 3D surface with the complexity of $\mathcal{K} = \mathcal{O}(mn)$ that could have some holes. It is possible to convert $\mathcal{W}$ to a polyhedral surface, $\mathcal{W}'$, with possibly some holes, in $\mathcal{O}(\mathcal{K})$ time, using the proposed technique in Section 5.4. This conversion preserves the length of L_1 shortest path from any s_i to any t_j , $i, j = 1, \dots, m$. Therefore, the existing algorithms for finding a L_1 shortest path on polyhedral surfaces can be used to find L_1 shortest paths in $\mathcal{W}'$, from s_i to t_j . The best known algorithm to find a L_1 shortest path on polyhedral surfaces has been proposed by Cheng and Jin [139] with time complexity of $\mathcal{O}(\mathcal{K}^2 \log^4(\mathcal{K})) = \mathcal{O}((mn)^2 \log^4(mn))$. This time complexity is much higher than the complexity of the proposed algorithm in this chapter.

Chapter 8

Open Problems and Future Work

In Chapter 2, we discussed the weighted region problem for an arrangement $\mathcal{A}$ of lines. We proposed a minimal region that contains shortest paths from s to t . It is a closed polygonal region that is independent of the weights assignment. In the following, we provide a list of the open problems related to Chapter 2:

- A question arises regarding the existence of such bounded region for arrangements of line segments in an appropriately defined weighted region problem. The formal problem statement is as follows: let s and t be two points in the plane $\mathbb{R}^2$ and let S be a set of n line segments. The subdivision of the plane induced by members of S is called the arrangement $\mathcal{A}(S)$ of S . It consists of vertices, edges, and faces. A positive weight w_i is assigned to each face of $\mathcal{A}(S)$. The task is to find a minimal closed region in $\mathbb{R}^2$ that contains a weighted shortest path from s to t .
- We proposed an algorithm to compute the SP-Hull in the absence of any assumption about the weights of faces of $\mathcal{A}$. An open problem is if we can improve (i.e., reduce the area of) the SP-Hull when the weights are known.
- An interesting extension of this problem is the question of existence of such a bound for a given arrangement of curves (e.g., algebraic curves). Also, we can discuss the problem in higher dimensions (e.g., 3D).

In the following, we list possible extensions for the work discussed in Chapter 3:

- Designing a post-processing refinement algorithm for approximation algorithms in 3D.

- Designing an algorithm to refine the approximation results for other types of the shortest path problem (e.g., the shortest anisotropic path problem).

In Chapter 4, a variant of the Fréchet distance was discussed. Using the common dog-leash metaphor for Fréchet distance, this variant is outlined as follows. Assume that a person wants to walk along one curve and his/her dog on another one. For a given constant leash length $\varepsilon \geq 0$, we want to determine a walk that minimizes the total length of all parts of the curves for which a leash length bigger than ε is needed. In this setting, the following is a list of related open problems:

- We know that the **MaxIn** problem is solvable exactly in the L_1 metric (see [120]). Is it possible to achieve a constant factor (i.e., $\sqrt{2}$) approximation for the **MaxIn** problem when the metric is L_2 , by using the solution for the L_1 metric?
- In Chapter 4, we assumed that a fixed leash distance is given and we wanted to minimize the total length of all parts of the curves for which a leash length bigger than ε is needed. However, in the standard Fréchet distance problem the objective is to find a walk that minimizes the leash length. Note that a simultaneous minimization of these two objectives may conflict. A variant of the problem which may have a polynomial-time solution, is to minimize the leash length under a given bound for the other criterion. Therefore, we look for an exact algorithm for this variant of the problem.
- It is possible to discuss the new variant of the Fréchet distance (considered in Chapter 4) when the input curves are weighted. For example, some parts of the curves could be more important and there is a priority to have a leash length less than ε in these parts. Furthermore, in the case that backward movement is allowed, an open problem is to find a walk that minimizes both the weighted length of the ignored parts and the distance traveled.
- We know that for surfaces, in general, the decision problem of the Fréchet distance is

NP-hard. It is hard even if the surfaces are “well-behaved” terrains [37]. The semi-computability of the Fréchet distance between surfaces is discussed in [142]. The Fréchet distance for two homeomorphic orientable surfaces, P and Q , is defined as follows:

$$\delta_F(P, Q) = \inf_{\sigma} \max_{t \in P} L_2(t, \sigma(t)) \quad (8.1)$$

where $L_2(.,.)$ is the Euclidean metric and $\sigma : P \rightarrow Q$ ranges over all orientation-preserving homeomorphisms between P and Q [140]. It is still open to design an approximation algorithm, similar to the variant of the Fréchet distance introduced in Chapter 4, to measure the similarity of two polyhedral surfaces in the presence of outliers.

- In order to measure the similarity in a cluster of curves in the presence of outliers, we may use the proposed variant of the Fréchet distance considered in Chapter 4, for every two curves inside the cluster and then aggregate the results. This solution could be expensive and needs $\mathcal{O}(m^2)$ measurements where m is the number of curves in a cluster. It is an open problem to design a more efficient algorithm to measure the similarity in a cluster of curves in the presence of outliers.
- It is also open if there exist a FPTAS for this problem.

In Chapter 5 and 6, we discussed MBFD and WMBFD problems. We showed that MBFD can be reduced to a L_1 shortest path problem in the white-space of the free-space diagram. This reduction leads to an algorithm with $\mathcal{O}(n^2 \log n)$ time complexity. In addition, we proposed a graph-based algorithm that solves WMBFD in $\mathcal{O}(n^2 \log^{3/2} n)$ time. Obviously, this algorithm can be used for the unweighted scenario. Here, is the list of related open problems. These problems could be discussed in both unweighted and weighted scenarios. The graph-based algorithm can be adapted for these problems. It is open if we can solve them more efficiently, i.e., in less time.

- In a same setting, assume that the backward movement could be allowed only in some parts of the curves and the remaining parts are one-way. This is the case in some applications. For example, suppose one of the curves is the path that a robot is moving on. Additional information about each segment of the curve (e.g., the slope, the road material) is given to the robot. Based on this information and other factors (e.g., engine power, type of wheels, etc.), the robot will determine that it is not be able to move backward in some parts of the curve. Therefore, the robot searches for a walk that minimizes the total cost of the backward movements only on feasible segments (see Figure 8.1 for an example).
- Furthermore, in a same setting, we could assume that there is a limit on the length of backward movement. This limit could be different for each segment of the input polygonal curves, or it could be a global constraint on the total length of backward movement.
- It is open if we can improve the time complexity for WMBFD further to match the time complexity for the unweighted version of the problem. One approach for improving the time complexity of our algorithm is to reduce the size of the graph, for each row and column, that is preserving rectilinear shortest paths (see [141]). We conjecture that, for our problem setting, the size of the graph could be linear.

In Chapter 7, we discussed a geometric algorithm for the map matching problem that minimizes the walking length. It remains open if we can improve the proposed algorithm further, if the graph is known to be planar. The main challenge here is the existence of cycles in the input graph and propagation through the cycles.

Figure 8.1: Suppose it is not allowed to move backward on the first and second segment of T_2 . In this example the weight of moving backward is 1 everywhere. The path Π_1 in the free-space is the L_1 shortest path from s to t . However, it is not a feasible solution to this instance of the problem since there is a backward movement on the first segment of T_2 . The path Π_2 is a solution.

Bibliography

- [1] Branko Grünbaum. *Are your polyhedra the same as my polyhedra?*, Discrete and Computational Geometry, 25:461-488, 2003.
- [2] Joseph O'Rourke, Subhash Suri. *Polygons*, Handbook of Discrete Computational Geometry Edited by J. E. Goodman and J. O'Rourke., Chapman and Hall/CRC, 2004.
- [3] Wolfram MathWorld. <http://mathworld.wolfram.com/SimplyConnected.html>
- [4] Joseph S. B. Mitchell. *Shortest path and networks*, Handbook of Discrete Computational Geometry Edited by J. E. Goodman and J. O'Rourke., Chapman and Hall/CRC, 2004.
- [5] Stefan Hertel, Kurt Mehlhorn. *Fast triangulation of the plane with respect to simple polygons*, Information and Control, 64(1):52-76, 1985.
- [6] Mark de Berg, Otfried Cheong, Marc van Kreveld, Mark Overmars. *Computational geometry: algorithms and applications*, Springer, 3rd edition, 2008.
- [7] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. *Introduction to algorithms*, The MIT Press, 3rd edition, 2009.
- [8] Minghui Jiang, Ying Xu, Binhai Zhu. *Protein structure-structure alignment with discrete Fréchet distance*, Journal of Bioinformatics and Computational Biology, 6(1):51-64, 2008.
- [9] Man-Soon Kim, Sang-Wook Kim, Miyoung Shin. *Optimization of subsequence matching under time warping in time-series databases*, In Proceedings of ACM symposium on Applied computing (SAC '05), pages 581-586, 2005.
- [10] Remco C. Veltkamp. *Shape matching: similarity measures and algorithms*, In Proceedings of International Conference on Shape Modeling and Applications, pages 188-197, 2001.
- [11] Helmut Alt, Bernd Behrends, Johannes Blömer. *Approximate matching of polygonal shapes*, Annals of Mathematics and Artificial Intelligence, 13(3-4): 251-265, 1995.
- [12] R. Sriraghavendra, K. Karthik, Chiranjib Bhattacharyya. *Fréchet distance based approach for searching online handwritten documents*, Document Analysis and Recognition, In Proceedings of 9th ICDAR, pages 461-465, 2007.
- [13] Vafa Khoshaein. *Trajectory clustering using a variation of Fréchet distance*, M.Sc. thesis, Carleton University, 2014.
- [14] Joachim Gudmundsson, Patrick Laube, Thomas Wolle. *Movement patterns in spatio-temporal data*, In S. Shekhar and H. Xiong, editors, Encyclopedia of GIS, pages 726-732, 2008.

- [15] Joachim Gudmundsson, Nacho Valladares. *A GPU approach to subtrajectory clustering using the Fréchet distance*, In Proceeding of ACM SIGSPATIAL, pages 259-268, 2012.
- [16] Joachim Gudmundsson, Thomas Wolle. *Football analysis using spatio-temporal tools*, In Proceeding of ACM SIGSPATIAL, pages 566-569, 2012.
- [17] Yu Zheng. *Trajectory data mining: an overview*, ACM Trans. Intelligent Systems and Technology, 6(3), Article 1, 2015.
- [18] Sotiris Brakatsoulas, Dieter Pfoser, Randall Salas, Carola Wenk. *On map-matching vehicle tracking data*, In Proceeding of ACM VLDB, pages 853-864, 2005.
- [19] Yin Lou, Chengyang Zhang, Yu Zhen, Xing Xi, Wei Wang, Yan Huang. *Map-matching for low-sampling-rate GPS trajectories*, In Proceeding of ACM SIGSPATIAL, pages 352-361, 2009.
- [20] Hector Gonzalez, Jiawei Han, Xiaolei Li, Margaret Myslinska, John Paul Sondag. *Adaptive fastest path computation on a road network: a traffic mining approach*, In Proceeding of ACM VLDB, pages 794-805, 2007.
- [21] Kaveh Shahbaz. *Applied similarity problems using Fréchet distance*, PhD Thesis, Carleton University, 2013.
- [22] Helmut Alt, Michael Godau. *Computing the Fréchet distance between two polygonal curves*, International Journal of Computational Geometry, 5:75-91, 1995.
- [23] Kevin Buchin, Maike Buchin, Joachim Gudmundsson, Maarten Löffler, Jun Luo. *Detecting commuting patterns by clustering subtrajectories*, International Journal of Computational Geometry, 21(3):253-282, 2011.
- [24] Kevin Buchin, Maike Buchin, Wouter Meulemans, Wolfgang Mulzer. *Four soviets walk the dog-with an application to Alts conjecture*, In Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 1399-1413, 2014.
- [25] Anne Driemel, Sarel Har-Peled. *Jaywalking your dog: computing the Fréchet distance with shortcuts*, In Proceedings of 23rd SODA, SIAM, pages 318-337, 2012.
- [26] Maike Buchin, Anne Driemel, Bettina Speckmann. *Computing the Fréchet distance with shortcuts is NP-hard*, In Proceeding of ACM Symposium on Computational Geometry (SoCG'14), pages 367-376, 2014.
- [27] Maike Buchin. *On the computability of the Fréchet distance between triangulated surfaces*, PhD thesis, Freie Universität Berlin, 2007.
- [28] Alon Efrat, Quanfu Fan, Suresh Venkatasubramanian. *Curve matching, time warping, and light fields: new algorithms for computing similarity between curves*, Journal of Mathematical Imaging and Vision, 27:203-216, 2007.

- [29] Helmut Alt, Christian Knauer, Carola Wenk. *Matching polygonal curves with respect to the Fréchet distance*, In Proceedings of 18th STACS, volume 2010 of Lecture Notes in Computer Science, Springer, pages 63-74, 2001.
- [30] Helmut Alt, Christian Knauer, Carola Wenk. *Comparison of distance measures for planar curves*, Algorithmica, 38(1):45-58, 2003.
- [31] Anne Driemel, Sariel Har-Peled, Carola Wenk. *Approximating the Fréchet distance for realistic curves in near linear time*, Discrete and Computational Geometry, 48(1):94-127, 2012.
- [32] Alon Efrat, Leonidas J. Guibas, Sariel Har-Peled, Joseph S. B. Mitchell, T. M. Murali. *New similarity measures between polylines with applications to morphing and polygon sweeping*, Discrete and Computational Geometry, 28(4):535-569, 2002.
- [33] Sergei Bespamyatnikh. *An optimal morphing between polylines*, International Journal of Computational Geometry and Applications, 12(3):217-228, 2002.
- [34] Atlas F. Cook IV and Carola Wenk. *Geodesic Fréchet distance inside a simple polygon*, ACM Trans. Algorithms, 7(1), Article 9, 19 pages, 2010.
- [35] Anil Maheshwari, Jörg-Rüdiger Sack, Kaveh Shahbaz, Hamid Zarrabi-Zadeh. *Fréchet distance with speed limits*. Computational Geometry: Theory and Applications, 44(2):110-120, 2011.
- [36] Yam Ki Cheung and Ovidiu Daescu. *Fréchet distance problems in weighted regions*, Discrete Mathematics, Algorithms and Applications, 2(2):161-180, 2010.
- [37] Sariel Har-Peled, Benjamin Raichel. *The frechet distance revisited and extended*, In Proceedings of the twenty-seventh annual symposium on computational geometry, pages 448-457, 2011.
- [38] Jacob E. Goodman, Janos Pach, Chee-Keng Yap. *Mountain climbing, ladder moving, and the ring-width of a polygon*, The American Mathematical Monthly, 96(6):494-510, 1989.
- [39] Thomas Eiter, Heikki Mannila. *Computing discrete Fréchet distance*. Technical report CD-TR 94/64, Christian Doppler Laboratory for Expert Systems, TU Vienna, Austria, 1994.
- [40] Pankaj K. Agarwal, Rinat Ben Avraham, Haim Kaplan, Micha Sharir. *Computing the discrete Fréchet distance in subquadratic time*, In Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 156-168, 2013.
- [41] Boris Aronov, Sariel Har-Peled, Christian Knauer, Yusu Wang, Carola Wenk. *Fréchet distance for curves, revisited*, In Proceedings of 14th ESA, Lecture Notes in Computer Science, volume 4168, pages 52-63, 2006.

- [42] Rinat Ben Avraham, Omrit Filtser, Haim Kaplan, Matthew J. Katz, Micha Sharir. *The discrete and semicontinuous Fréchet distance with shortcuts via approximate distance counting and selection*, ACM Trans. Algorithms, 11(4), Article 29, 29 pages, 2015.
- [43] Kevin Buchin, Maike Buchin, Christian Knauer, Günter Rote, Carola Wenk. *How difficult is it to walk the dog*, In Proceedings of 23rd European Workshop on Computational Geometry, pages 170-173, 2007.
- [44] Anka Gajentaan, Mark H. Overmars. *On a class of $\mathcal{O}(n^2)$ problems in computational geometry*, Computational Geometry: Theory and Applications, 45(4):140-152, 2012.
- [45] Allan Gronlund, Seth Pettie. *Threesomes, Degenerates, and Love Triangles*, In Proceedings of IEEE 55th Annual Symposium on Foundations of Computer Science (FOCS), pages 621-630, 2014.
- [46] Karl Bringmann. *Why walking the dog takes time: Fréchet distance has no strongly subquadratic algorithms unless SETH fails*, In Proceedings of 55th IEEE FOCS, pages 661-670, 2014.
- [47] Karl Bringmann, Wolfgang Mulzer. *Approximability of the discrete Fréchet distance*, In Proceedings of 31st International Symposium on Computational Geometry (SoCG), pages 739-753, 2015.
- [48] Amin Gheibi, Anil Maheshwari, and Jörg-Rüdiger Sack. *Weighted region problem in arrangement of lines*, In proceeding of 25th CCCG, 2013.
- [49] Jean-Lou De Carufel, Amin Gheibi, Anil Maheshwari, Jörg-Rüdiger Sack, Christian Scheffer. *Similarity of polygonal curves in the presence of outliers*, Computational Geometry: Theory and Applications, 47(5):625-641, 2014.
- [50] Amin Gheibi, Anil Maheshwari, Jörg-Rüdiger Sack, Christian Scheffer. *Minimum backward Fréchet distance*, In Proceedings of the 22nd ACM SIGSPATIAL, pages 381-388, 2014.
- [51] Amin Gheibi, Anil Maheshwari, Jörg-Rüdiger Sack. *Weighted minimum backward Fréchet distance*, accepted to 27th CCCG, Kingston, 2015.
- [52] Amin Gheibi, Anil Maheshwari, Jörg-Rüdiger Sack. *Minimizing walking length in map matching*, accepted to Topics in Theoretical Computer Science, Tehran, 2015.
- [53] Bernard Chazelle. *Triangulating a simple polygon in linear time*, Discrete Computational Geometry, 6:485-524, 1991.
- [54] Bernard Chazelle. *A theorem on polygon cutting with applications*, In Proceedings of 23rd Annu. IEEE Sympos. Found. Comput. Sci., pages 339-349, 1982.

- [55] Leonidas Guibas, John Hershberger, Daniel Leven, Micha Sharir, Robert E. Tarjan. *Lineartime algorithms for visibility and shortest path problems inside triangulated simple polygons*, Algorithmica, 2:209-233, 1987.
- [56] John Hershberger and Subhash Suri. *An optimal algorithm for Euclidean shortest paths in the plane*, Manuscript, Washington University, 1995.
- [57] John Hershberger. *A new data structure for shortest path queries in a simple polygon*, Inform. Process. Lett., 38:231–235, 1991.
- [58] John Hershberger and Subhash Suri. *Matrix searching with the shortest path metric*, SIAM J. Comput., 26(6):1612-1634, 1997.
- [59] Joseph S. B. Mitchell, David M. Mount, and Christos H. Papadimitriou. *The discrete geodesic problem*, SIAM J. Comput., 16(4):647-668, 1987.
- [60] Emo Welzl. *Constructing the visibility graph for n line segments in $O(n^2)$ time*, Inform. Process. Lett., 20:167-171, 1985.
- [61] M. Pocchiola and G. Vegter. *Topologically sweeping visibility complexes via pseudotriangulations*, Discrete Comput. Geom., 16:419-453, 1996.
- [62] Edsger W. Dijkstra. *A note on two problems in connexion with graphs*, Numerische Mathematik, 1:269-271, 1959.
- [63] Michael L. Fredman, Robert E. Tarjan. *Fibonacci heaps and their uses in improved network optimization algorithms*, J. ACM, 34:596-615, 1987.
- [64] Joseph S. B. Mitchell. *L_1 shortest paths among polygonal obstacles in the plane*, Algorithmica, 8:55-88, 1992.
- [65] Joseph S. B. Mitchell. *A new algorithm for shortest paths among obstacles in the plane*, Ann. Math. Artif. Intell., 3:83-106, 1991.
- [66] Subhash Suri. *A linear time algorithm with minimum link paths inside a simple polygon*, Comput. Vision Graph. Image Process., 35(1):99-110, 1986.
- [67] Joseph S. B. Mitchell, Günter Rote, and Gerhard J. Woeginger. *Minimum-link paths among obstacles in the plane*, Algorithmica, 8:431-459, 1992.
- [68] Joseph S.B. Mitchell, Valentin Polishchuk, Mikko Sysikaski. *Minimum-link paths revisited*, Computational Geometry: Theory and Applications, 47(6):651-667, 2014.
- [69] Gautam Das, Giri Narasimhan. *Geometric searching and link distance*, In Proceedings of 2nd WADS, Lecture notes in Computer Science, volume 591, pages 261-272, 1991.

- [70] Robert L. Drysdale III, Clifford Stein, and David P. Wagner. *An $O(n^{5/2} \log n)$ algorithm for the rectilinear minimum link-distance problem in three dimensions*. In Proceedings of the 17th Canadian Conference on Computational Geometry, pages 94-97, 2005.
- [71] Anil Maheshwari, Jörg-Rüdiger Sack, and Hristo N. Djidjev. *Link distance problems*, In Jörg-Rüdiger Sack and J. Urrutia, editors, Handbook of Computational Geometry, chapter 12, Elsevier, 2000.
- [72] Kenneth L. Clarkson, Sanjiv Kapoor, Pravin M. Vaidya. *Rectilinear shortest paths through polygonal obstacles in $O(n(\log n)^2)$ time*, In Proceedings of 3rd ACM SoCG, pages 251-257, 1987.
- [73] Kenneth L. Clarkson, Sanjiv Kapoor, and Parvin M. Vaidya. *Rectilinear shortest paths through polygonal obstacles in $O(n(\log n)^{3/2})$ time*, *Manuscript*, 1988.
- [74] David Eppstein. *Finding the k shortest paths*, In Proceedings of Foundations of Computer Science, pages 154-165, 1994.
- [75] D. T. Lee, Chung-Do Yang, and T. H. Chen. *Shortest rectilinear paths among weighted obstacles*, International Journal of Computational Geometry, 1(2): 109-124, 1991.
- [76] Danny Z. Chen, Haitao Wang. *L_1 shortest path queries among polygonal obstacles in the plane*, In Proceedings of 30th STACS, pages 293-304, 2013.
- [77] Danny Z. Chen, Kevin S. Klenk, Hung-Yi Tu. *Shortest path queries among weighted obstacles in the rectilinear plane*, In Proceedings of 11th Annu. ACM Sympos. Comput. Geom., pages 370-379, 1995.
- [78] Andrzej Lingas, Anil Maheshwari, Jörg-Rüdiger Sack. *Parallel algorithms for rectilinear link distance problems*, Algorithmica, 14:261-289, 1995.
- [79] Srinivasa Arikati, Danny Z. Chen, L. Paul Chew, Gautam Das, Michiel Smid, Christos D. Zaroliagis. *Planar spanners and approximate shortest path queries among obstacles in the plane*, In Proceedings of ESA, Lecture Notes in Computer Science, volume 1136, pages 514-528, 1996.
- [80] Joseph S. B. Mitchell and Christos H. Papadimitriou. *The weighted region problem: finding shortest paths through a weighted planar subdivision*, J. ACM 38(1):18-73, 1991.
- [81] Jean-Lou De Carufel, Carsten Grimm, Anil Maheshwari, Megan Owen, and Michiel Smid. *A note on the unsolvability of the weighted region shortest path problem*, Computational Geometry: Theory and Applications, 47(7):724-727, 2014.
- [82] Lyudmil Aleksandrov, Anil Maheshwari, and Jörg-Rüdiger Sack. *Determining approximate shortest paths on weighted polyhedral surfaces*, J. ACM 52(1):25-53, 2005.

- [83] D. Nieuwenhuisen, A. Kamphuis, M. Mooijekind, and M. Overmars. *Automatic construction of roadmaps for path planning in games*, In proceeding of International Conference on Computer Games: Artificial Intelligence, Design and Education, pages 285-292, 2004.
- [84] Wonshik Chee, Masayoshi Tomizuka. *Lane change maneuver of automobiles for the intelligent vehicle and highway system (IVHS)*, In proceedings of American Control Conference, pages 3586-3587, 1994.
- [85] Winston Nelson. *Continuous-curvature paths for autonomous vehicles*, In proceeding of IEEE International Conference on Robotics and Automation, pages 1260-1264, 1989.
- [86] Peter A. Burrough, Rachael A. McDonnell, Christopher D. Lloyd. *Principles of Geographical Information Systems*, OUP Oxford, ISBN: 0198742843, 2013.
- [87] <http://people.scs.carleton.ca/~agheibi/tins/tins.html>
- [88] NIST/SEMATECH e-Handbook of Statistical Methods, <http://www.itl.nist.gov/div898/handbook/>, 2012.
- [89] B.L.S. Prakasa Rao. *Nonparametric functional estimation*, Academic Press, 1983.
- [90] Noel A. C. Cressie. *Statistics for spatial data*, revised edition, John Wiley and Sons Inc., New York, 1993.
- [91] Thomas D. Gauthier. *Detecting trends using Spearman's rank correlation coefficient*, Environmental Forensics, 2(4):359-362, 2001.
- [92] Eric Gervais, Hongsheng Liu, Doron Nussbaum, Young-Soo Roh, Jörg-Rüdiger Sack, Jiehua Yi. *Intelligent map agents - an ubiquitous personalized GIS*, Journal of Photogrammetry and Remote Sensing 62(5):347-365, 2007.
- [93] Kiho Im, Hang Joon Jo, Jean-Francois Mangin, Alan C. Evans, Sun I. Kim, Jong-Min Lee. *Spatial distribution of deep sulcal landmarks and hemispherical asymmetry on the cortical surface*, Cerebral Cortex 20(3):602-611, 2010.
- [94] Ovidiu Daescu, James D. Palmer. *Finding optimal weighted bridges with applications*, In Proceedings of the 44th Annual Southeast Regional Conference, pages 12-17, 2006.
- [95] Mark Lanthier, Anil Maheshwari, and Jörg-Rüdiger Sack. *Approximating shortest paths on weighted polyhedral surfaces*, Algorithmica 30, 4:527-562, 2001.
- [96] Zheng Sun, John H. Reif. *On finding approximate optimal paths in weighted regions*, J. Algorithms 58, 132, 2006.
- [97] Siu-Wing Cheng, Jiongxin Jin, Anotoine Vigneron, Yajun Wang. *Approximate shortest homotopic paths in weighted regions*, International Journal of Computational Geometry and Applications, 22 (1): 83-102, 2012.

- [98] Norman Jaklin, Mark Tibboel, Roland Geraerts. *Computing high-quality paths in weighted regions*, In Proceedings of the Seventh International Conference on Motion in Games (MIG '14), ACM, New York, NY, USA, pages 77-86, 2014.
- [99] Hristo Djidjev, Christian Sommer. *Approximate distance queries for weighted polyhedral surfaces*, In Proceedings of 19th ESA, pages 579-590, 2011.
- [100] John Canny, John Reif. *New lower bound techniques for robot motion planning problems*, In Proceedings of 28th Annu. IEEE Sympos. Found. Comput. Sci., pages 49-60, 1987.
- [101] Lyudmil Aleksandrov, Anil Maheshwari, and Jörg-Rüdiger Sack. *Approximation algorithms for geometric shortest path problems*, In Proceedings of 32nd Annual ACM Symposium on Theory of Computing, pages 286-295, 2000.
- [102] Ovidiu Daescu, Joseph S. B. Mitchell, Simeon Ntafos, James D. Palmer, Chee K. Yap. *An experimental study of weighted k -link shortest path algorithms*, Algorithmic Foundation of Robotics VII, Springer Tracts in Advanced Robotics Volume 47, pages 187-202, 2008.
- [103] Lyudmil Aleksandrov, Hristo Djidjev, Anil Maheshwari, Jörg-Rüdiger Sack. *An approximation algorithm for computing shortest paths in weighted 3-d domains*, Discrete and Computational Geometry, 50(1):124-184, 2013.
- [104] Michiel Hagedoorn. *Pattern matching using similarity measures*, PhD thesis, Utrecht University, 2000.
- [105] Daniel P. Huttenlocher, Klara Kedem, Micha Sharir. *The upper envelope of Voronoi surfaces and its applications*, Discrete and Computational Geometry, 9:267-291, 1993.
- [106] L. Paul Chew, Klara Kedem. *Improvements on approximate pattern matching*, In 3rd Scandinavian Workshop on Algorithm Theory, Lecture Notes in Computer Science 621, pages 318-325, 1992.
- [107] Daniel P. Huttenlocher, Klara Kedem, Jon M. Kleinberg. *On dynamic Voronoi diagrams and the minimum Hausdorff distance for point sets under Euclidean motion in the plane*, In Proceedings of the eighth annual symposium on Computational geometry (SCG '92), pages 110-119, 1992.
- [108] Peter Bras, Christian Knauer. *Testing the congruence of d -dimensional point sets*, In Proceedings of the sixteenth annual symposium on Computational geometry (SCG '00), pages 310-314, 2000.
- [109] Piotr Indyk, Suresh Venkatasubramanian. *Approximate congruence in nearly linear time*, Computational Geometry: Theory and Applications, 24:115-128, 2003.

- [110] Pankaj K. Agarwal, Sarel Har-Peled, Micha Sharir, Yusu Wang. *Hausdorff distance under translation for points and balls*, ACM Trans. Algorithms 6(4), Article 71, 26 pages, 2010.
- [111] Sarana Nutanong, Edwin H. Jacox, Hanan Samet. *An incremental Hausdorff distance calculation algorithm*, In Proceedings of VLDB Endowment, 4(8):506-517, 2011.
- [112] Joseph S. B. Mitchell, Micha Sharir. *New results on shortest paths in three dimensions*, In: Snoeyink, J., Boissonnat, J.-D. (eds.) ACM Symposium on Computational Geometry, pages 124-133, 2004.
- [113] Hamid Zarrabi-Zadeh. *Flying over a polyhedral terrain*, Information Processing Letters, 105(3):103-107, 2008.
- [114] Christos H. Papadimitriou. *An algorithm for shortest path motion in three dimensions*, Inform. Process. Lett., 20:259-263, 1985.
- [115] Tetsuo Asano, David Kirkpatrick, Chee Yap. *Pseudo approximation algorithms with applications to optimal motion planning*, Discrete and Computational Geometry, 31(1):139-171, 2004.
- [116] Pankaj K. Agarwal, R. Sharathkumar, Hai Yu. *Approximate Euclidean shortest paths amid convex obstacles*, In Proceedings of SODA, pages 283-292, SIAM, 2009.
- [117] Mikhail Atallah. *Computing the convex hull of line intersections*. J. Algorithms 7:285-288, 1986.
- [118] Marc J. van Kreveld and Lionov Wiratma. *Median trajectories using well-visited regions and shortest paths*, In Proceedings of 19th ACM SIGSPATIAL (GIS '11), pages 241-250, 2011.
- [119] Kevin Buchin, Maike Buchin, Marc J. van Kreveld, Maarten Löffler, Rodrigo I. Silveira, Carola Wenk, Lionov Wiratma. *Median trajectories*, ESA 2010, Part I, LNCS 6346, pages 463-474, 2010.
- [120] Kevin Buchin, Maike Buchin, Yusu Wang. *Exact algorithms for partial curve matching via the Fréchet distance*, In Proceedings of 20th SODA, pages 645-654, 2009.
- [121] Free-space diagram ipelet from Günter Rote's homepage: <http://www.inf.fu-berlin.de/inst/ag-ti/people/rote/Software/ipelets.html#fsd>
- [122] Timothy William Flynn, Sean M. Connery, Michael A. Smutok, Jorge Zeballos, Idelle M. Weisman. *Comparison of cardiopulmonary responses to forward and backward walking and running*, Medicine and science in sports and exercise, 26(1):89-94, 1994.
- [123] Subir K. Ghosh, David M. Mount. *An output-sensitive algorithm for computing visibility graphs*, SIAM Journal on Computing, 20(5): 888-910, 1991.

- [124] Sarel Har-Peled and Yusu Wang, *Partial curve matching under the Fréchet distance*, Technical Report, Ohio State University, <ftp://ftp.cse.ohio-state.edu/pub/tech-report/2007/TR73.pdf>, 2007.
- [125] Lyudmil Aleksandrov, Hristo Djidjev, Hua Guo, Anil Maheshwari, Doron Nussbaum, Jörg-Rüdiger Sack. *Algorithms for approximate shortest path queries on weighted polyhedral surfaces*, Discrete and Computational Geometry, 44:762-801, 2010.
- [126] Mark de Berg. *Better bounds on the union complexity of locally fat objects*, In Proceedings of SoCG, pages 39-47, 2010.
- [127] Frank van der Stappen, Dan Halperin, Mark H. Overmars. *The complexity of the free space for a robot moving amidst fat obstacles*, Comput. Geom. Theory Appl., 3:353-373, 1993.
- [128] Nancy M. Amato, Michael T. Goodrich, Edgar A. Ramos. *A randomized algorithm for triangulating a simple polygon in linear time*, Discrete and Computational Geometry 26 (2): 245-265, 2001.
- [129] Sang Won Bae, Matias Korman, Yoshio Okamoto. *The Geodesic diameter of polygonal domains*, ESA 2010, Lecture Notes in Computer Science, Volume 6346, pages 500-511, 2010.
- [130] Personal communication with Dr. André Pugin, Natural Resources of Canada, and Dr. Dariush Motazedian, Carleton University, Department of Earth Sciences, February 2012.
- [131] G. E. Blelloch. *Prefix sums and their applications*, Synthesis of Parallel Algorithms, Morgan Kaufmann, 1990.
- [132] Bi Y. Chen, Hui Yuan, Qingquan Li, William H.K. Lam, Shih-Lung Shaw, Ke Yan. *Map-matching algorithm for large-scale low-frequency floating car data*, International Journal of Geographical Information Science, 28(1):22-38, 2014.
- [133] Fengli Ruan, Zhongliang Deng, Qian An, Keji Wang, Xiaoyang Li. *A method of map matching in indoor positioning*, In Proceedings of China Satellite Navigation Conference (CSNC): Volume III, Lecture Notes in Electrical Engineering, 305, pages 669-679, 2014.
- [134] Koichi Asakura, Masayoshi Takeuchi, Toyohide Watanabe. *A pedestrian-oriented map matching algorithm for map information sharing systems in disaster areas*, International Journal of Knowledge and Web Intelligence, 3(4):328-342, 2012.
- [135] Helmut Alt, Alon Efrat, Günter Rote, Carola Wenk. *Matching planar maps*, In Proceedings of the fourteenth annual ACM-SIAM symposium on discrete algorithms (SODA), pages 589-598, 2003.
- [136] Daniel Chen, Anne Driemel, Leonidas J. Guibas, Andy Nguyen, Carola Wenk. *Approximate map matching with respect to the Fréchet distance*, In Proceedings of the Thirteenth Workshop on Algorithm Engineering and Experiments (ALENEX), pages 75-83, 2011.

- [137] Md Zakirul Alam Bhuiyan, Guojun Wang, Athanasios V. Vasilakos. *Local area prediction-based mobile target tracking in wireless sensor networks*, IEEE Transactions on Computers, 64(7): 1968 - 1982, 2015.
- [138] Harsh Vachhani. *Continuous spatio temporal tracking of mobile targets*, Master's Thesis, Arizona State University, 2014.
- [139] Siu-Wing Cheng, Jiongxin Jin. *Shortest Paths on Polyhedral Surfaces and Terrains*, In Proceedings of ACM Symposium on Theory of Computing, pages 373-382, 2014.
- [140] Kevin Buchin, Maike Buchin, Andre Schulz. *Fréchet distance of surfaces: some simple hard cases*, In Proceedings of ESA, Lecture Notes in Computer Science, Volume 6347, pages 63-74, 2010.
- [141] Peter Widmayer. *On graphs preserving rectilinear shortest paths in the presence of obstacles*, Annals of Operations Research, 33(7):557-575, 1991.
- [142] Helmut Alt, Maike Buchin. *Semi-computability of the Fréchet distance between surfaces*, In proceedings of EWCG, pages 45-48, 2005.