On the complexity of range searching among curves

Peyman Afshani Anne Driemel

Abstract

Modern tracking technology has made the collection of large numbers of densely sampled trajectories of moving objects widely available. We consider a fundamental problem encountered when analysing such data: Given n polygonal curves S in $\mathbb{R}^d$, preprocess S into a data structure that answers queries with a query curve q and radius ρ for the curves of S that have Fréchet distance at most ρ to q.

We initiate a comprehensive analysis of the space/query-time trade-off for this data structuring problem. Our lower bounds imply that any data structure in the pointer model model that achieves $Q(n) + O(k)$ query time, where k is the output size, has to use roughly $\Omega\big((n/Q(n))^2\big)$ space in the worst case, even if queries are mere points (for the discrete Fréchet distance) or line segments (for the continuous Fréchet distance). More importantly, we show that more complex queries and input curves lead to additional logarithmic factors in the lower bound. Roughly speaking, the number of logarithmic factors added is linear in the number of edges added to the query and input curve complexity. This means that the space/query time trade-off worsens by an *exponential* factor of input and query complexity. This behaviour addresses an open question (see [1, 7]) in the range searching literature concerning multilevel partition trees which may be of independent interest, namely, whether it is possible to avoid the additional logarithmic factors in the space and query time of a multilevel partition tree. We answer this question negatively.

On the positive side, we show we can build data structures for the Fréchet distance by using semialgebraic range searching. The space/query-time trade-off of our data structure for the discrete Fréchet distance is in line with the lower bound, as the number of levels in the data structure is $O(t)$, where t denotes the maximal number of vertices of a curve. For the continuous Fréchet distance, the number of levels increases to $O(t^2)$.

1 Introduction

Recent technological advances have made it possible to collect trajectories of moving objects, indoors and outdoors, on a large scale using various technologies, such as GPS [16], WLAN, Bluetooth, RFID [18] or video analysis [12]. In this paper we study time-space trade-offs for data structures that store trajectory data and support similarity retrieval. In particular we focus on the case where the similarity or distance between two curves is measured by the Fréchet distance. This distance measure is widely studied in computational geometry and gives high-quality results for trajectory data. We focus on the case where the query should return all input curves in a specified *range*, given by a query curve q and a radius ρ. The range is defined as the set of curves that have Fréchet distance at most ρ to q, i.e., the metric ball of radius ρ centered at q. Our study is timely as it coincides with the 6th GIS-focussed algorithm competition hosted by ACM SIGSPATIAL[1] drawing attention to this very problem from the practical domain.

At the same time, our results address a broader question concerning multilevel partition trees, a very important classical tool from the range searching literature. See the following survey for more background [4], but briefly, in range searching the goal is to store a set of n points such that the points in a query region can be found efficiently. One of the most prominent problems is when the queries are simplicies in $\mathbb{R}^d$, a problem known as *simplex range searching*. Interestingly, the known solutions for simplex range searching can be easily repackaged into multilevel data structures that can even solve more difficult problems, such as simplex-simplex searching: store a set of n simplicies such that the simplicies that are entirely contained in a query simplex can be found efficiently. For some illustrative examples on the versatility and power of multilevel data structures see [7].

The concept of multilevel partition tree based data structures is broad and mathematically not well-defined. Roughly speaking, in a multilevel data structure, first a base data structure is built that defines some notion of first generation canonical sets. Then on the first generation canonical sets, a secondary set of data structures are built which in turn defines a second generation canonical sets. Continuing this "nested" structure for t levels would yield a multilevel data structure with t levels. This flexibility, allows more complex problems (such as simplex-simplex searching problem mentioned above) to be solved and with very little effort and by only degrading space or query time by small factors. It seems intuitively obvious that each additional level should blow up the space or the query time of the data structure and in fact all known data structures suffer an exponential factor in t (often a $\log^{O(t)} n$ factor). It has been conjectured that this should be the case but not even a polynomial blow up was proven before (see [1, 7]).

Exponential vs polynomial dependency. To better understand the situation, let us momentarily focus on planar data structures with linear or near-linear space. For the main problem of simplex range reporting, there exist data structures with $O(n)$ space and $O(n^{1-1/d} + k)$ query time where k is the output size. This query time is conjectured to be optimal and there exist lower bounds that almost match it up to a $2^{O(\sqrt{\log n})}$ factor ([1]) or $\log n$ factor ([8]). Thus, the base problem of simplex range reporting is well-understood. However, beyond this, things are less clear. In particular, we would like to know what happens if the query regions or the input are more complex objects. Assume the input is a set of n points but the query is a tuple of t hyperplanes that define a polytope. To report the set of points inside the query polytope, we can triangulate the query polytope into $O(t^d)$ simplices and then we can ask a simplex range searching query for each resulting

[1] 6th ACM SIGSPATIAL GISCUP 2017, see also `http://sigspatial2017.sigspatial.org/giscup2017/`

simplex. This will not alter the space consumption at all and it will only blow up the query time by a factor $O(t^d)$ but for a constant d, this factor is a fixed *polynomial* of the query complexity, t. This example shows that such "obviously more complex" queries can actually be handled very efficiently. Now consider what happens if both queries and input are complex objects. Consider a problem in which the input is a set containing n tuples where the i-th tuple $\mathbf{p}_i$ is composed of t points, i.e., $\mathbf{p}_i = (\mathbf{p}_i^{(1)}, \ldots, \mathbf{p}_i^{(t)})$ where each $\mathbf{p}_i^{(j)} \in \mathbb{R}^d$, and the query is a tuple of t simplices $(\sigma_1, \ldots, \sigma_t)$. The goal is to report all the input tuples $\mathbf{p}_i$ such that $\mathbf{p}_i^{(j)}$ lies inside σ_j for every $1 \leq j \leq t$. In this case, seemingly, the best thing to do is to build a multilevel data structure composed of t levels. Such a data structure will consume $O(n \log^{t-1} n)$ space and will have the query time of $O(n^{1-1/d} \log^{t-1} n)$ using the best known results in the literature on multilevel data structures [7]. The crucial difference here is that both space and query time degrade *exponentially* in t as opposed to the polynomial dependency in the previous case. The main open question here is whether this exponential factor is required.

The picture becomes more interesting once one looks at the history of multilevel data structures and once one realizes that there are many ways to build them. In Matoušek's [17] original paper, one would sacrifice a $\log^2 n$ factor for space and a $\log n$ factor for the query time but this comes at a larger pre-processing time. If one wishes to reduce the preprocessing time, then the loss increases to an unspecified number of $\log n$ factors per level. Chan [7] offers the currently best known way to build multilevel data structures at only one $\log n$ factor loss for space and query time per level (in fact, in some cases, we can do even better). This brings us to the main lower bound question regarding multilevel data structures.

Question 1. *Is it possible to avoid the additional logarithmic factors for every level in the space and query time of a multilevel data structure?*

We at least partially settle this open question by showing that the space/query time tradeoff must blow up by at least a roughly $\log n$ factor for every increase in t. To do that, we show that a particular problem that can be solved using multilevel data structures is hard.

We remark that the above question is ambiguous since we did not provide a mathematically precise definition of a "multilevel data structure". Such a definition would have to capture the versatility of the multilevel approach to data structuring. For instance, multilevel partition trees can have different fan-outs at different levels, they can selectively use duality restricted to individual levels, or they can use different auxiliary data structures mixed in with them. A crucial and arguably most useful property of the multilevel structures is that different levels can handle completely independent subproblems. By lack of a precise definition that is commonly agreed upon and perhaps in the hope to prove an even stronger statement, we take a different approach: We prove a lower bound for a concrete relevant multilevel data structuring problem (Problem 3 on page 4).

The problem only involves independent points and simplicies (the basic components of a simplex range reporting problem) and thus any multilevel data structure must be able to solve the problem. This means, a lower bound for this problem gives a lower bound for the general class of multilevel data structures. From this point of view, our lower bound is in fact stronger: it shows that the multilevel stabbing problem is strictly more difficult than the ordinary simplex range searching problem, even if we are not restricted to use "multilevel data structures."

Not only that, but we also show that the lower bound also generalizes to geometric search structures based on Fréchet distance: preprocess a set of n polygonal curves of complexity t such that given a query polygonal curve of complexity t, we can find all input curves within some distance

ρ of the query (Problems 1 and 2 on page 3). This lower bound is not obvious and it also provides additional motivation to study multilevel data structures. The fact that we can extend our lower bound to such a practically relevant problem emphasizes the relevance of our lower bounds.

2 Definitions and Problem Statement

A polygonal chain s is a sequence of vertices $s_1, \ldots, s_t \in \mathbb{R}^d$. The discrete Fréchet distance of two chains s and q is defined using the concept of traversals. A *traversal* is a sequence of pairs of indices $\{(i_1, j_1), (i_2, j_2), \ldots, (i_k, j_k)\}$ such that $i_1 = 1, j_1 = 1, i_k = t_s$, and $j_k = t_q$ and one of the following holds for each pair (i_m, j_m) with $m > 1$: (i) $i_m = i_{m-1}$ and $j_m = j_{m-1} + 1$, or (ii) $i_m = i_{m-1} + 1$ and $j_m = j_{m-1} + 1$, or (iii) $i_m = i_{m-1} + 1$ and $j_m = j_{m-1}$. The discrete Fréchet distance is defined as

$$d_F(s, q) = \min_{T \in \mathcal{T}} \max_{(i,j) \in T} \|s_i - q_j\| \tag{1}$$

Finding the traversal that minimizes the Fréchet distance is called the *alignment problem*.

The continuous Fréchet distance is defined for continuous curves. For a polygonal chain s, we obtain a polygonal curve by linearly interpolating s_i and s_{i+1}, i.e., adding the edge $\overline{s_i s_{i+1}} = \{\alpha s_i + (1 - \alpha) s_{i+1} \mid \alpha \in [0, 1]\}$ in between each pair of consecutive vertices. The curve s has a uniform parametrization that allows us to view it as a parametrized curve $s : [0, 1] \to \mathbb{R}^d$. The Fréchet distance between two such parametrized curves is defined as

$$d_F(s, q) = \min_{f:[0,1]\to[0,1]} \max_{\alpha \in [0,1]} \|s(\alpha) - q(f(\alpha))\|, \tag{2}$$

where f ranges over all continuous and monotone bijections with $f(0) = 0$ and $f(1) = 1$.

In this paper we consider the following two problems based on the Fréchet distance.

Problem 1 (Discrete Frechet Queries)**.** *Let S be an input set of n polygonal chains in $\mathbb{R}^d$ where each polygonal chain has size at most t_s. Given a parameter ρ, we would like to store S in a data structure such that given a query polygonal chain q of length at most t_q, we can find all the chains in S that are within the discrete Frechet distance ρ of q, see Equation* (1)*.*

Problem 2 (Continuous Frechet Queries)**.** *Let S be an input set of n polygonal curves in $\mathbb{R}^d$ where each polygonal curve consists of at most t_s vertices. Given a parameter ρ, we would like to store S in a data structure such that given a query polygonal curve q consisting of t_q vertices, we can find all the curves in S that are within the continuous Frechet distance ρ of q, see Equation* (2)*.*

For both problems, the output size is the number of input curves that match the query requirements.

Since we will be working with tuples of points and geometric ranges, we introduce the following notations to simplify the description of our results.

A *t-point* $\mathbf{p}$ in $\mathbb{R}^d$ is a tuple of t points $(\mathbf{p}^{(1)}, \mathbf{p}^{(2)}, \ldots, \mathbf{p}^{(t)})$ where each $\mathbf{p}^{(i)}$ is a point in $\mathbb{R}^d$. The concepts of t-hyperplanes and t-halfspaces, and etc. are defined similarly. A slab s is the region between two parallel hyperplanes. The *thickness* of s is the distance between the hyperplanes and it is denoted by $\tau(s)$. A *t-slab* $\mathbf{s}$ in $\mathbb{R}^d$ is a tuple of t slabs $(\mathbf{s}^{(1)}, \mathbf{s}^{(2)}, \ldots, \mathbf{s}^{(t)})$ where each $\mathbf{s}^{(i)}$ is a slab. The *thickness* of $\mathbf{s}$ is defined as $\prod_{j=1}^{t} \tau(\mathbf{s}^{(j)})$. A t-point $\mathbf{p}$ is inside a t-slab $\mathbf{s}$ if $\mathbf{p}^{(i)}$ is inside $\mathbf{s}^{(i)}$ for

every $1 \leq i \leq t$. We will adopt the convention that the i-th point $\mathbf{p}$ in a t-point is denoted by $\mathbf{p}^{(i)}$. The same applies to the other definitions.

We will also show a lower bound for the following concrete problem.

Problem 3 (Multilevel Stabbing Problem)**.** *Let S be an input set containing n t-slabs. We would like to store S in a data structure such that given a query t-point $\mathbf{p}$ we can find all the t-slabs $\mathbf{s} \in S$ such that $\mathbf{s}$ contains $\mathbf{p}$.*

The pointer machine model. The model of computation that we use for our lower bound is the pointer machine model. This model is very suitable for proving lower bounds for range reporting problems. Consider an abstract data structure problem where the input is a set S of elements and where a query q (implicitly) specifies a subset $S_q \subset S$ that needs to be output by the data structure. In the pointer machine model, the storage of the data structure is represented using a directed graph M with constant out-degree where each vertex in M corresponds to one memory cell. Each memory cell can store one element of S. The constant out-degree requirement means that each memory cell can point to at most a constant number of other memory cells. The elements of S are assumed to be atomic, meaning, to answer a query q, for each element $v \in S_q$, the data structure must visit at least one vertex (i.e., cell) that stores v. To visit that subset of cells, the data structure starts from a special vertex of M (called the root) and follows pointers: the data structure can visit a memory cell u only if it has already visited a cell v such that v points to u. There is no other restriction on the data structure, i.e., it can have unlimited information and computational power. The size of the graph M lower bounds the storage usage of the data structure and the number of nodes visited while answering a query lower bounds the query time of the data structure. Thus, when proving lower bounds in the pointer machine model, it is sufficient to show that if the data structure operates on a small graph M, then during the query time, it has to visit a lot of cells (or vice versa).

3 Related Work on Fréchet Queries

Few data structures are known which support Fréchet queries of some type. We review the space and query time obtained by these data structures. In the following, let n denote the number of curves in the data structure and let t denote the maximum number of vertices of each curve. The data structures can be distinguished by the type of queries answered: (i) nearest neighbor queries [15, 11], (ii) range counting queries [10, 13],

Before we discuss these data structures, we would like to point out that under certain complexity-theoretic assumptions both (i) and (ii) above become much harder for long curves, and in particular for $t \in \omega(\log n)$. More specifically, there is a known reduction from the orthogonal vectors problem which implies that, unless the orthogonal vectors hypothesis fails, there exists no data structure for range searching or nearest neighbor searching under the (discrete or continuous) Fréchet distance that can be built in $O\big(n^{2-\varepsilon}\text{poly}(t)\big)$ time and achieves query time in $O\big(n^{1-\varepsilon}\text{poly}(t)\big)$ for any $\varepsilon > 0$ (see also the discussion in [11]).

A data structure by Indyk supports approximate nearest-neighbor searching under the discrete Fréchet distance [15]. The query time is in $O\big(t^{O(1)} \log n\big)$ and the approximation factor is in $O(\log t + \log\log n)$. The data structure uses space in $O\Big(|X|^{\sqrt{t}}(t^{\sqrt{t}}n)^2\Big)$, where $|X|$ is the size of the domain on which the curves are defined. The data structure precomputes all answers to queries

with curves of length $\sqrt{t}$, leading to a very high space consumption. A recent result by Driemel and Silvestri [11] shows that it is possible to construct locality-sensitive hashing schemes for the discrete Fréchet distance. One of the main consequences is a $O(t)$-approximate near-neighbor data structure that achieves $O(t \log n)$ query time and $O(n \log n + tn)$ space.

As for the continuous Fréchet distance, de Berg, Gudmundsson and Cook study the problem of preprocessing a single polygonal curve into a data structure to support range counting queries among the subcurves of this curve [10]. The data structure uses a multilevel partition tree to store compressed subcurves. This representation incurs an approximation factor of $2 + 3\sqrt{2}$ in the query radius. For any parameter $n \leq s \leq n^2$, the space used by the data structure is in $O(s \text{ polylog}(n))$. The queries are computed in time in $O\Big(\frac{n}{\sqrt{s}} \text{ polylog}(n)\Big)$. However, the data structure does not support more complex query curves than line segments.

The motivation to study the subcurves of a single curve originated from the application of analyzing single trajectories of individual team sports players during the course of an entire game. A different application, namely the map matching of trajectories onto road maps [5] led Gudmundsson and Smid to study slightly more general input—consider the geometric graph that represents a road map and a range query among the set of paths in the graph. Gudmundsson and Smid study the case where the input belongs to a certain class of geometric trees [13]. Based on the result of de Berg, Gudmundsson and Cook they describe a data structure which supports approximate range emptiness queries and can report a witness path if the range is non-empty. Furthermore, the queries can be more complex than mere line segments. The data structure has size $O(n \text{ polylog}(n))$, preprocessing time in $O(n \text{ polylog}(n))$ and answers queries with polygonal curves of t vertices in $O(t \text{ polylog}(n))$ time.

It should be noted that the latter two data structures [10, 13] make strict assumptions on the length of the edges of the query curves with respect to the query radius which seems to simplify the problem. While it is widely believed, based on complexity-theoretic assumptions, that there is no $O(t^{2-\varepsilon})$-time algorithm for any $\varepsilon > 0$ that can decide if the discrete or continuous Fréchet distance between two curves is at most a given value of δ (see Bringmann [6]), this problem drastically simplifies if δ is smaller than half of the maximal length of an edge of the two curves. In particular, a simple linear scan can solve the decision problem in $O(t)$ time. Our results do not make any assumptions on the length of the edges of the curves or the distribution of the edges.

4 Our Results

We show the first upper and lower bounds for exact range searching under the discrete and continuous Fréchet distance. Our lower bounds are in fact obtained for the multilevel stabbing problem and it proves that the space $S(n)$ required for answering the multilevel stabbing queries in $Q(n) + O(k)$ time must obey

$$S(n) = \Omega((\frac{n}{Q(n)})^2) \cdot \frac{\left(\frac{\log(n/Q(n))}{\log\log n}\right)^{t-1}}{2^{O(2^t)}} \quad \text{as well as} \quad S(n) = \Omega\left(\frac{n}{Q(n)}\right)^2 \Theta\left(\frac{\log(n/Q(n))}{t^{3+o(1)}\log\log n}\right)^{t-1-o(1)}. \tag{3}$$

Here k is the size of the output and t is the number of levels. Based on what we have discussed, not only this proves the first separation between the simplex range reporting data structures and

multilevel data structures, but it also shows space should increase exponentially in t, as long as $t \leq (\log n)^{1/3-\varepsilon}$.

For the Fréchet distance queries, a set of n polygonal curves in $\mathbb{R}^d$ is given as input, where each input curve consist of at most t_s vertices. A query with a curve of t_q vertices and query radius ρ returns the set of input curves that have Fréchet distance at most ρ to q.

(i) Assume there exists a data structure that achieves $Q(n)$ query time and uses $S(n)$ space in the pointer model. We show that the $S(n)$ must obey the same lower bound in Eq. 3 where $t \leq \min(t_s/4, t_q/2)$.

In addition, we show how to build multilevel partition trees for the discrete and the continuous Fréchet distance using semi-algebraic range searching:

(ii) For the discrete Fréchet distance we descibe a data structure which uses space in $\mathcal{O}\big(n(\log\log n)^{t_s-1}\big)$ and achieves query time in $\mathcal{O}\Big(n^{1-1/d} \cdot \log^{O(t_s)} n \cdot t_q^{\mathcal{O}(d)}\Big)$, assuming $t_q = \log^{O(1)} n$.

(iii) For the continuous Fréchet distance we describe a data structure for $d = 2$ which uses space in $\mathcal{O}\Big(n(\log\log n)^{O(t_s^2)}\Big)$ and achieves query time in $\mathcal{O}\Big(\sqrt{n}\log^{O(t_s^2)} n\Big)$, assuming $t_q = \log^{O(1)} n$. For the second data structure, the query radius has to be known at preprocessing time.

5 Outlines of the Technical Proofs

5.1 Outline of the lower bounds

We first prove lower bounds for the reporting variant of the multilevel stabbing problem in the pointer machine model. By what we discussed, this gives a lower bound for multilevel data structures. Next, we build sets of input curves and query curves that show the same lower bounds can be realized under the Fréchet distance. Before we sketch the lower bound construction, we say a few words about the lower bound framework we use.

5.1.1 The framework of the proofs

Our reporting lower bound uses a volume argument by Afshani [1]. This argument can be used to show lower bounds for stabbing reporting queries, i.e., the input is a set of ranges and a query with a point returns all ranges that contain this point. Imagine, we want to answer any query in $Q(n) + O(k)$ time where k is the size of the output. In order to set up the volume argument we need to define a set of queries $\mathcal{Q}$ that has volume one and a set of input ranges, such that (i) each query point is covered by sufficiently many ranges (by at least $Q(n)$ ranges), and (ii) the volume of the intersection of any subset of ℓ ranges is sufficiently small, i.e., at most v. Then, the framework shows that the space is asymptotically lower bounded by $Q(n)v^{-1}2^{-O(\ell)}$. The intuition of why the framework works is the following: the intersection of some subset of ranges is the locus of (query) points that must output those particular subset of ranges. Thus, if the intersection of every subset of ℓ ranges is small, then our set of queries $\mathcal{Q}$ contains many different queries that each output a different subset of input ranges. Thus, precomputing (and implicitly storing) partial answers must increase the space according to these volumes.

5.1.2 Multilevel Stabbing Problem

We start with the unit cube $\mathcal{Q}$ in $\mathbb{R}^{2t}$ where t denotes the number of levels. In particular, we view $\mathcal{Q}$ as the Cartesian product of t unit squares: $\mathcal{Q} = \mathcal{Q}_1 \times \cdots \times \mathcal{Q}_t$. The input is a set of n t-slabs in $\mathcal{Q}$.

The query is a t-point in $\mathcal{Q}$. The main part of the proof is an intricate construction of the t-slabs.

The main result here is Lemma 1 (see page 11). We will not repeat the exact technical claim and instead we will focus on the general ideas and the intuition behind them. The first step is to build $r = Q(n)$ different sets of t-slabs $\mathcal{S}_1, \ldots, \mathcal{S}_r$ of roughly n/r size, such that the slabs in each set $\mathcal{S}_i$ tile $\mathcal{Q}$, i.e., any t-point is covered by exactly one t-slab. This will directly satisfy condition (i) of the framework described in Section 5.1.1. The difficult part is to find a good construction that guarantees that every subset of ℓ t-slabs have a small intersection.

To build $\mathcal{S}_i$, we build a set $\mathcal{S}_{i,j}$ of two-dimensional slabs in each unit square $\mathcal{Q}_j$ such that they together tile $\mathcal{Q}_j$. Then, $\mathcal{S}_i$ is taken to be the set of t-slabs that one obtains by creating the Cartesian product of all the slabs created in $\mathcal{Q}_1, \ldots, \mathcal{Q}_t$. See Figure 5 on page 20 for an illustration. In order to obtain small intersection volume we would like to adjust the thickness of the two-dimensional slabs. While adjusting the thickness of the slabs in each universe, we make sure that we create roughly n/r t-slabs in $\mathcal{S}_i$: This boils down to making sure that the product of the thicknesses of the two-dimensional slabs is a fixed value τ. We have t degrees of freedom to pick the orientation of the slabs and thus we can represent the set of angles that define the orientation of the slabs in each $\mathcal{Q}_j$ by a point in $\mathbb{R}^t$; we call these points, "parametric points". Thus, every set $\mathcal{S}_i$ has one parametric point and in our construction there are r parametric points in total.

The parametric points need to be placed very carefully. In particular our construction places the parametric point such that the volume of the smallest axis-aligned rectangle created by any two parametric points is maximized (see Lemma 3 in page 14). Intuitively, this means that if the points are "well-spread" so that no small axis-aligned box can contain two points, then the volume of the intersection of the slabs is also going to be large.

Regarding the thicknesses of the slabs, we have $t-1$ degrees of freedom since the product of the thicknesses is set to be a fixed value τ. However, we need to place more restrictions on the thicknesses. We make sure that the different thicknesses are sufficiently different. In particular, we set the width to be in the form of R^w for some fixed parameter R and some integer w. This means that for each slab we allow roughly a logarithmic number of different possible thicknesses. Thus, the $t-1$ degrees of freedom in choosing the thicknesses are translated to freedom in choosing $t-1$ integers in some narrow range (between 0 and roughly $\log_R n$). Note that this freedom is only present for the first $t-1$ two-dimensional slabs, that is, in $\mathcal{Q}_1, \ldots, \mathcal{Q}_{t-1}$ and the thickness of the last slab is determined based these values and the value of τ. This further implies that the sum of the integers that we choose should also be within the same narrow range. Nonetheless, unlike the case of angles, our choices in picking these integers are represented *combinatorially* as a single value and we treat it like a color. In other words, we define a set of all the available colors (roughly $((\log_R n)/t)^{t-1}$) and then associate each set $\mathcal{S}_i$ with a color; the color determines the thickness of slabs in $\mathcal{Q}_1, \mathcal{Q}_2, \ldots, \mathcal{Q}_t$.

Thus, after placing r parametric points in $\mathbb{R}^t$, we need to color each point with a color. This coloring needs to be done carefully as well. The placement and the coloring of the points are done using one lemma (Lemma 3 in page 14).

However, more work is required to make the construction work. We need to impose some favorable combinatorial structure on the set of colors that we create by removing some of the colors. This is done by sampling a small number of colors.

Finally, we try to bound the volume of the intersection of any ℓ of the n t-slabs that we created. Any two slabs in $\mathcal{S}_i$ are disjoint and thus for a non-zero intersection, the ℓ slabs should come from ℓ different sets, e.g., $\mathcal{S}_1, \ldots, \mathcal{S}_\ell$. The straightforward argument gives us a bound on v that ultimately

gives the same lower bound as simplex range reporting. So we perform a non-obvious analysis. We look at two possible cases: Either (i) two of the parametric points of $\mathcal{S}_1, \ldots, \mathcal{S}_\ell$ have the same color and in this case we use the properties of our coloring (see Lemma 3 in page 14) to ensure that such points are "well-spread"; in particular, if we have n_c colors, we can make sure that the parametric points of each color are a factor roughly n_c "better spread", meaning, the volume of the smallest axis-aligned rectangle that contains two points of the same color is a factor n_c larger than the volume of the smallest rectangle that contains two points of different colors. Ultimately, this buys us a n_c factor in our lower bound. Observe that the value n_c grows exponentially on t (up to some maximum value). The other case is when (ii) all the parametric points of $\mathcal{S}_1, \ldots, \mathcal{S}_\ell$ have distinct colors. By using the favorable combinatorial property that we had imposed earlier on the set of colors, we find 3 colors among the many different distinct colors and an index j such that these colors have three distinct values at coordinate j. This in turn implies that the slabs in $\mathcal{Q}_j$ have 3 distinct thicknesses. However, thicknesses differ by at least a factor R and thus further analysis buys us a factor R on value of v. By combining the two cases, we show that we can improve our lower bound by either a factor n_c or factor R. We set our parameters such that R and n_c are roughly equal and we obtain the lower bound.

5.1.3 Constructions for the Fréchet Distance

In order to apply the above construction to the Fréchet range searching we dualize the Fréchet query ranges to some extent. Our dualization differs significantly between the two variants of the problem. For the discrete Fréchet distance we observe that the set of points that lie within Fréchet distance ρ to a line segment are contained in the intersection of the two circles of radius ρ centered at the two endpoints. We call the intersection of two circles a *lens*. Thus, we create a set of lenses as input instead of a set of slabs and we let the vertices of the query curve act as stabbing queries. Refer to Figure 1 on page 9 for an illustration of this straight-forward approach. We observe that inside the unit square, lenses can be made to almost look like slabs, that is, for any slab, we can create a lens that is fully contained in the slab such that the area of the symmetric difference between the slab and the lens is made arbitrarily small. As a result, after a little bit more work, we can show that the construction for the multilevel stabbing problem directly gives a lower bound for the discrete Fréchet queries problem.

In contrast, our construction for the continuous Fréchet distance dualizes the *lines* supporting the edges of the query curve, creating a separate "universe" for every odd edge (in lieu of a universe for every vertex). Here, our construction is such that the locus of query curves in the dual space that lie within Fréchet distance ρ to a specific input curve forms a set of slabs—one in each universe. To this end we let the input curve follow a zig-zag shape. We use one zig-zag curve per universe. Refer to Figure 7 on page 23 for an example of a zig-zag used in the construction. Our analysis uses the basic fact that the set of lines intersecting a vertical interval in the primal space corresponds to the set of points enclosed in a slab in the dual space. We combine this fact with a well-known connection between the Fréchet distance and ordered line-stabbing initially observed by Guibas *et al.* [14]. This observation says that the line supporting the query edge needs to stab the disks of radius ρ centered at the input curve in their correct order. For our zig-zag curves this has the effect that the line needs to intersect the vertical interval formed by the two intersection points of the circles of radius ρ centered at the two corners of the zig-zag. We can control the width, orientation and position of the resulting slab in the dual space by varying the length and the position of this vertical interval. Using these proof elements, we can show that the lower bound of the multilevel

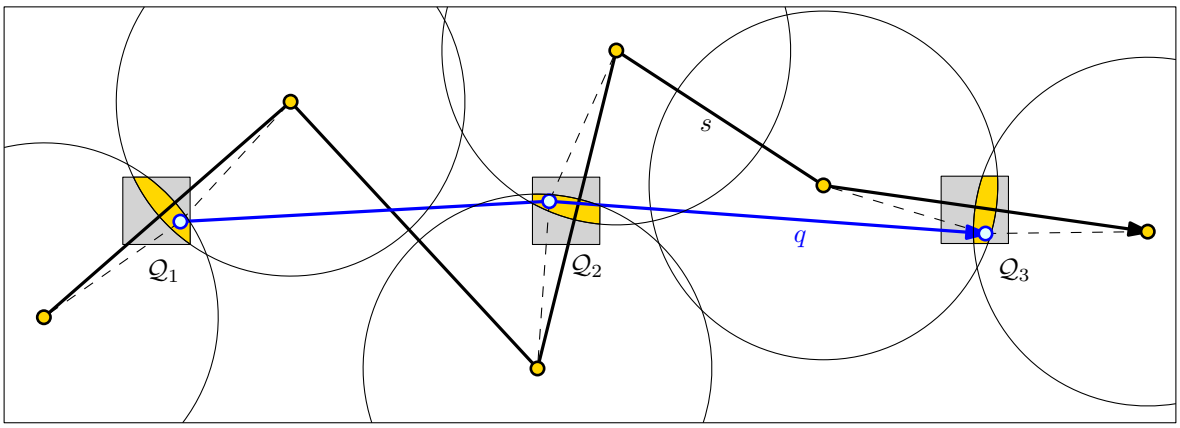


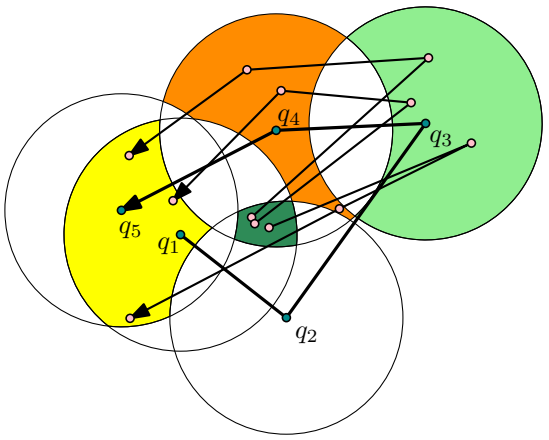
Figure 1: Illustration of the lower bound construction for the discrete Fréchet distance showing universes $\mathcal{Q}_1$, $\mathcal{Q}_2$ and $\mathcal{Q}_3$. For every i, a query curve q has its ith vertex inside $\mathcal{Q}_i$. The intersection of two disks centered at the vertices of the ith odd edge of an input curve s forms a "near-slab" and needs to contain the ith vertex of a query curve if s is contained in the query range centered at q.

stabbing problem which is analyzed in the beginning, carries over to the continuous Fréchet distance as well.

5.2 Outline of the data structures

To obtain our upper bounds, we perform an extensive analysis of the definition of the Fréchet distance that allows us to restate the alignment problem using a sequence of semialgebraic range queries. One of the challenges here is to design a set of filters that do not create duplicates in the output across the different range queries that need to be performed. We first focus on the discrete Fréchet distance, where the analysis is significantly cleaner and simpler. The dynamic programming algorithm which is commonly used to compute the discrete Fréchet distance uses a Boolean matrix, the so-called free space matrix, to decide which alignments between the curves are feasible. The entry (i, j) of this matrix indicates if the Euclidean distance between the ith vertex of one curve and the jth vertex of the other curve is at most ρ. The two curves have Fréchet distance at most ρ if and only if there exists a traversal that only uses the 1-entries in the free-space matrix. The set of possible truth assignments to this matrix induces a partition on the input curves with respect to their free space matrix with the query curve. Furthermore, each set in this partition is either completely contained in the query range or it is completely disjoint from the query range. We show how to construct a multilevel data structure that allows us to query independently for each of those sets which are contained in the query range.

Our query processing works in three phases. First, we compute all feasible free space matrices based on the arrangement of balls centered at vertices of the query. Next, we refine this arrangement to obtain cells of constant complexity that can be described by the zero set of a polynomial function. In the third phase we query the data structure with each free space matrix separately, using semialgebraic range searching in each level of the data structure to filter the input curves that have their ith vertex inside a specific cell of the refined arrangement. To see how this works, consider the set of ith vertices of the input curves that lie in a fixed cell of the arrangement of balls centered at the vertices of the query curve. The corresponding input curves share the same truth assignment in



	s_1	s_2	s_3	s_4
q_1	1	0	0	1
q_2	1	0	0	0
q_3	0	1	0	0
q_4	1	0	1	0
q_5	0	0	0	1

Figure 2: Example of a query matrix for the discrete Fréchet distance with a feasible traversal (right). The truth assignment in a fixed column corresponds to a cell in the arrangement of balls centered at the vertices of the query curve (left). The figure also shows three input curves that have this free-space matrix with the query curve and would thus be reported.

the ith column of the free-space matrix with q. Refer to Figure 2 for an example.

We now build a standard multilevel partition tree on the polygonal chains. In the ith level we store the ith points of the input curves. Our query algorithm processes the free-space matrix in a column-by-column fashion, where we use the convention that the column index refers to a point on the input curves and a row index refers to a point on the query curve. This makes the storage layout of the data structure independent of the number of vertices of the query curve.

For the continuous Fréchet distance the approach is similar, at least on a high level. The main difference is that the Boolean matrix that guides the queries is more complicated, since we operate on the continuous free-space diagram instead of the discrete free-space matrix. We first define high-level predicates that carry enough information to decide the Fréchet distance. Each predicate involves a constant number of edges and vertices from the input and query curves, e.g., testing the feasibility of a monotone path for a combination of a row and two vertical lines in the free-space diagram. Next we show how to represent these predicates using more basic operations, e.g., above-below relationships between points and lines that can be dualized. Finally, the query algorithm will test groups of these predicates for each feasible truth assignment separately. Also here we manage to keep the layout of the data structure independent of the complexity of the query curve.

There are two main challenges in dealing with the continuous case. One is to obtain the more complicated discrete matrix that captures all possible free-space diagrams of the fixed query curve with any *arbitrary* possible input curve. The second challenge is to make sure we can express all our predicates in the framework of semialgebraic range searching in two dimensions. Our solution is non-obvious since the Fréchet distance is not defined as a closed-form algebraic expression. This second challenge is the main issue that prevents us from directly generalizing our data structure to higher dimensional queries.

5.3 Organization

We prove the lower bounds in Section 6. We first show the lower bound for the multilevel stabbing problem. The construction is given in Section 6.2. We discuss the range reporting lower bound in Section 6.3. In Section 6.4 we show how to implement the construction for the two variants of Fréchet queries. We describe our data structures in Section 7. In Section 7.1 we describe the machinery that we use to build our data structures. In Section 7.2 we develop a data structure for discrete Fréchet queries. In Section 7.3 we extend these ideas and develop a data structure for continuous Fréchet queries. We conclude with some open problems in Section 8.

6 Lower Bounds

As discussed, we prove lower bounds for a concrete *problem*, that is, the multilevel stabbing problem. To do that, we need to construct a "difficult" input instance of n t-slabs with certain desirable properties. This construction is at the heart of our lower bounds and this is what we are going to attempt in this section.

6.1 Definitions

Our lower bounds for the multilevel stabbing problem is based on an intricate construction that we outline in this subsection. Define the space $\mathcal{Q} = \mathcal{Q}_1 \times \mathcal{Q}_2 \times \ldots \mathcal{Q}_t$ where each $\mathcal{Q}_i$ is the unit cube in the plane. $\mathcal{Q}$ now represents the set of all possible queries: a query t-point $\mathbf{p}$ is represented by the point $\mathbf{p} \in \mathcal{Q}$ which corresponds to a point $\mathbf{p}^{(i)}$ in $\mathcal{Q}_i$, for every $1 \leq i \leq t$. Observe that the points $\mathbf{p}^{(i)}$ are completely independent. Similarly, an input t-slab $\mathbf{s}$ is represented by picking t independent slabs, one slab $\mathbf{s}^{(i)}$ in $\mathcal{Q}_i$ for each $1 \leq i \leq t$.

Consider a (measurable) subset $f \subset \mathbb{R}^{D'}$ that lies in a D-dimensional flat V of $\mathbb{R}^{D'}$, $D \leq D'$. We denote the D-dimensional Lebesgue measure of f with $\mathrm{Vol}_D(f)$. For a set of points $q_1, \ldots, q_s \in \mathbb{R}^D$, we denote the smallest axis-aligned box that contains them all with $\mathrm{Box}_D(q_1, \ldots, q_s)$. Finally, for two t-slabs $\mathbf{s}_1$ and $\mathbf{s}_2$, we say $\mathbf{s}_1$ is a *translation* of $\mathbf{s}_2$ if for every index j, the slabs $\mathbf{s}_1^{(j)}$ and $\mathbf{s}_2^{(j)}$ are parallel and have the same thickness.

6.2 The 2D Construction

Lemma 1. *Consider parameters t, τ, r, R, n_c and ℓ under constraints to be specified shortly. We can build a set $\mathcal{S}$ of r t-slabs such that $\tau(\mathbf{s}) = \tau$, for every $\mathbf{s} \in \mathcal{S}$. Furthermore, (i) for any ℓ t-slabs $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_\ell \in \mathcal{S}$ and any ℓ t-slabs $\mathbf{s}'_1, \ldots, \mathbf{s}'_t$ such that $\mathbf{s}'_j$ is a translation of $\mathbf{s}_j$, we have $Vol_{2t}(\mathbf{s}'_1 \cap \mathbf{s}'_2 \cap \cdots \cap \mathbf{s}'_\ell) \leq \max\left\{\frac{\tau^2 r t^{t+o(t)}}{n_c}, \frac{\tau^2 r t^{t+o(t)}}{R}\right\}$.*

The constraints are that $\Omega(1) \leq r \leq n/2$, $\Omega(1) \leq R \leq n^{1/(2t)}$, $2 \leq \ell < r$ and that n_c is defied as $n_c = (\frac{\log_R n}{t})^{t-1}$ when $\ell \geq 2^t$ and $n_c = \Theta((\frac{\log_R n}{t})^{t-1} 2^{-t} (\frac{\log_R n}{t})^{-t/\ell})$ when $\ell < 2^t$.

As we shall see later, combined with the existing framework, the above lemma offers our desired lower bounds with only little bit more work. Thus, the main challenge is actually proving the above lemma. The main idea is the following: To define each t-slab in $\mathcal{S}$, we have the freedom to pick t different angles, one angle for each universe $\mathcal{Q}_j$, $1 \leq j \leq t$. We also have the freedom to alter the thickness of the slabs we constructed in each $\mathcal{Q}_j$. Thus, we have "t degrees of freedom" to pick the angles and "$t-1$ degrees of freedom" to pick the slab thickness. The former t degrees

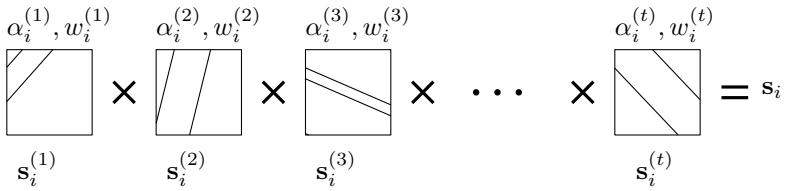


Figure 3: Each $\mathbf{s}_i$ is defined as the Cartesian product of t 2-dimensional slabs. The thickness of $\mathbf{s}_i^{(j)}$ is $R^{-w_i^{(j)}}$.

of freedom are represented as points (that we call "parametric points") in $\mathbb{R}^t$ and the latter are represented combinatorially as "colors". To make the construction work, we do not allow for all possible combinations of "colors" and instead we prune the colors using a combinatorial technique. We ultimately isolate a sub-problem that is very connected to orthogonal range searching. This is a very satisfying since it was suspected that there could be connections between orthogonal range searching and multilevel non-orthogonal range searching [2]. As a result, we manage to incorporate some techniques from orthogonal range searching lower bound in our construction (see Theorem 2). However, combining the colors (i.e., the "orthogonal component") and the set of parametric points (the non-orthogonal component) requires a careful analysis.

6.2.1 Parameters Defining the t-slabs.

To construct each slab in $\mathcal{S}$, we use $2t-1$ parameters: assume, we would like to construct a slab $\mathbf{s}_i \in \mathcal{S}$. We use t real-valued parameters $\alpha_i^{(1)}, \alpha_i^{(2)}, \ldots, \alpha_i^{(t)}$ and $t-1$ integral parameters $w_i^{(1)}, w_i^{(2)}, \ldots, w_i^{(t-1)}$. We call these $2t-1$ parameters the *defining parameters of* $\mathbf{s}_i$. $\alpha_i^{(j)}$ is the angle slab $\mathbf{s}_i^{(j)}$ makes with the X-axis, and the thickness of $\mathbf{s}^{(j)}$ is defined as $\tau(\mathbf{s}_i^{(j)}) = R^{-w_i^{(j)}}$, for $1 \leq j \leq t-1$. However, since we would like to end up with a t-slab $\mathbf{s}_i$ such that $\tau(\mathbf{s}_i) = \tau$, we define $w_i^{(t)} = \log_R(\tau^{-1}) - \sum_{j=1}^{t-1} w_i^{(j)}$ and $\tau(\mathbf{s}_i^{(t)}) = R^{-w_i^{(t)}}$. Note that $w_i^{(t)}$ is not necessarily an integer. We have:

Observation 1.

$$\tau(\mathbf{s}_i) = \prod_{1 \leq j \leq t} \tau(\mathbf{s}_i^{(j)}) = \tau(\mathbf{s}_i^{(t)}) \prod_{1 \leq j \leq t-1} \tau(\mathbf{s}_i^{(j)}) = R^{-(\log_R(\tau^{-1}) - \sum_{j=1}^{t-1} w_i^{(j)}) - \sum_{j=1}^{t-1} w_i^{(j)}} = \tau.$$

Definition 1. *Consider a t-slab $\mathbf{s}_i \in \mathcal{S}$. Let $\alpha_i^{(1)}, \ldots, \alpha_i^{(t)}$ and color $(w_i^{(1)}, \ldots, w_i^{(t)})$ be the defining parameters of $\mathbf{s}_i$. We call the point $\phi(\mathbf{s}_i) = (\alpha_i^{(1)}, \ldots, \alpha_i^{(t)}) \in \mathbb{R}^t$ the parametric point of $\mathbf{s}_i$ and denote it with $\phi(\mathbf{s}_i)$ and with the tuple $(w_i^{(1)}, w_i^{(2)}, \ldots, w_i^{(t-1)})$ being its color.*

We have to make very careful choices when picking the defining parameters of $\mathbf{s}_i$. We discuss how to pick the colors in Section 6.2.3.

We now establish some basic facts about this construction.

Observation 2. *Consider ℓ t-slabs $\mathbf{s}_1, \ldots, \mathbf{s}_\ell$. We have $Vol_{2t}(\mathbf{s}_1 \cap \mathbf{s}_2 \cap \ldots \mathbf{s}_\ell) = \prod_{i=1}^{t} Vol_2(\mathbf{s}_1^{(i)} \cap \mathbf{s}_2^{(i)} \cap \cdots \cap \mathbf{s}_\ell^{(i)})$.*

[2] For example, Chan [7] compares non-orthogonal multilevel data structures to d-dimensional range trees that can be viewed as d-levels of 1-dimensional data structures.

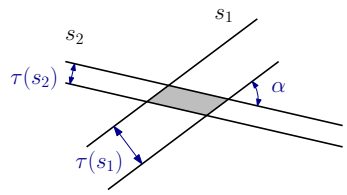


Figure 4: Distance between the parallel lines forming slabs s_1 and s_2 is $\tau(s_1)$ and $\tau(s_2)$ respectively. If the angle between the slabs is α, then the area of the shaded region is $\Theta(\tau(s_1)\tau(s_2)/\alpha)$.

We will also use the following elementary geometry observation regarding the area of the intersection of two slabs.

Observation 3. *Consider two 2-dimensional slabs s_1 and s_2 of thickness $\tau(s_1)$ and $\tau(s_2)$ respectively. And let α be the angle between them. Then, $Vol_2(s_1 \cap s_2) = O(\tau(s_1)\tau(s_2)/\alpha)$. (See Figure 4.)*

Lemma 2. *Let $\mathbf{s}_1, \mathbf{s}_2 \in \mathcal{S}$ be two t-slabs and let $\phi(\mathbf{s}_1)$ and $\phi(\mathbf{s}_2)$ be the parametric points of $\mathbf{s}_1$ and $\mathbf{s}_2$ respectively. Then regardless of the colors of these two t-slabs, $Vol_{2t}(\mathbf{s}_1 \cap \mathbf{s}_2)$ is asymptotically bounded by*

$$\frac{\tau^2}{Vol_t(Box_t(\phi(s_1), \phi(s_2))}.$$

Proof. By Observations 3 and 2, $\mathrm{Vol}_{2t}(s_1 \cap s_2)$ is asymptotically upper bounded by

$$\prod_{i=1}^{t} \frac{\tau(\mathbf{s}_1^{(i)})\tau(\mathbf{s}_2^{(i)})}{|\alpha_1^{(i)} - \alpha_2^{(i)}|}.$$

By Observation 1, the nominator equals τ^2 since the thickness of both $\mathbf{s}_1$ and $\mathbf{s}_2$ is τ. Then, the lemma then follows from observing that $\prod_{i=1}^{t} |\alpha_1^{(i)} - \alpha_2^{(i)}|$ is exactly $\mathrm{Vol}_t(\mathrm{Box}_t(\phi(s_1), \phi(s_2))$. □

6.2.2 Coloring and the Parametric Points

In this subsection, we will discuss how to pick the parametric points of the slabs in $\mathcal{S}$. Essentially, we will place a set of r points in $\mathbb{R}^t$ using the upcoming constructions. We extend the following construction that is used in lower bounds for the orthogonal problems.

Theorem 1. *[2, 3, 9] For any parameter N, we can place a set P of N points inside the unit cube in $\mathbb{R}^D$ such that for any two points $p, q \in P$, we have $Vol_D(Box_D(p, q)) = \Omega(1/N)$ where the constant in the asymptotic notation depend on D.*

To choose the parametric points, we use the method in [2]. But since in our case, the dimension t is not longer considered a constant, we need to provide a tight analysis, and determine its precise dependency on the dimension (by using the prime number theorem).

Theorem 2. *For any parameter N, we can place a set P of N points inside the cube $[N]^D$ in $\mathbb{R}^D$ such that for any two points $p, q \in P$ we have $Vol_D(Box_D(p, q)) = \Omega(\frac{N^{D-1}}{D^{D+o(D)}})$.*

Proof. We use the same construction in [2]: We pick the first $D-1$ prime numbers $a_1, \ldots, a_{D-1}$ and we place N points on the integer points in $[N-1]^D$. The coordinates of the i-th point p_i is $p_i = (\overline{i_{(a_1)}}, \overline{i_{(a_2)}}, \ldots, \overline{i_{(a_{D-1})}}, i)$, for $0 \leq i < N$, where $\overline{i_{(x)}}$ is the "reversed" (or inverted) representation of i in base x using $\lfloor \log_x N \rfloor + 1$ digits [3]. In [2], it is only proved that the volume of the axis-aligned box that contains 2 points is $\Omega(\frac{1}{N})$ with a "constant" that depends on D. Here, we need to make the dependence on D explicit.

Consider two points $p_k, p_{k'}$ such that $k < k'$. Let $d = k' - k$. Observe that if x^y divides d but x^{y+1} does not, then the representation of d in base x contains exactly y leading zeros (and as a conclusion, $\overline{d_{(x)}}$ contains exactly zeros at its y most significant digits). This implies that for any natural number z such that, $z + d < N$, $\overline{z_{(x)}}$ and $\overline{z + d_{(x)}}$ agree on exactly y of their most significant digits, which yields the bound $|\overline{z_{(x)}} - \overline{z + d_{(x)}}| < N/x^y$. Let B be the smallest box that contains p_k and $p_{k'}$, $L_1, \ldots, L_D$ be the side lengths of B and v be its volume. Thus, $L_1 L_2 \ldots L_D = v$. Let ℓ_i be the integer such that $N/a_i^{\ell_i - 1} > L_i \geq N/a_i^{\ell_i}$. Based on the above observation, $(k' - k)_{(a_i)}$ must contain $\ell_i - 1$ leading zeros or in other words, $a_i^{\ell_i - 1}$ divides $k' - k$. Let $F = a_1^{\ell_1 - 1} a_2^{\ell_2 - 1} \ldots a_{D-1}^{\ell_{D-1} - 1}$. Since a_i's are relatively prime, it follows that F also divides $k' - k$. However, observe that

$$F = a_1^{\ell_1} a_2^{\ell_2} \ldots a_{D-1}^{\ell_{D-1}} \geq \frac{N^{D-1}}{L_1 L_2 \ldots L_{D-1}} = \frac{N^{D-1} L_D}{v} = \frac{N^{D-1}(k' - k)}{v}.$$

Let $X = \prod_{i=1}^{D-1} a_i$. We claim $v \geq N^{D-1}/(2DX)$, since otherwise we will reach a contradiction. To see this, assume to the contrary that $v < N^{D-1}/(2DX)$. Assuming this, we get

$$F = a_1^{\ell_1 - 1} a_2^{\ell_2 - 1} \ldots a_{D-1}^{\ell_{D-1} - 1} \geq \frac{N^{D-1}(k' - k)}{Xv} > k' - k \tag{4}$$

which is a contradiction since F must divide $k' - k$.

It remains to estimate X. By Prime number theorem, we know that $a_i = O(i \log i)$. Thus, using Stirling's approximation, we have

$$X = \prod_{i=1}^{D-1} O(i \log i) \leq 2^{O(D)} \cdot D! \cdot (\log D)^D = D^{D + o(D)}.$$

□

Using the above theorem, we prove the following result.

Lemma 3. *Consider the unit cube $\mathcal{Q}$ in $\mathbb{R}^D$. Let N and n_c be two integral parameters such that $n_c < N/2$. We can place a set W of N points in $\mathcal{Q}$ and assign each an integer color from 0 to n_c such that the following hold: (i) for any two points p and q we have $Vol_D(Box_D(p, q)) = \Omega(\frac{1}{ND^{D+o(D)}})$ and (ii) for any two points p and q that have the same color we have $Vol_D(Box_D(p, q)) = \Omega(\frac{n_c}{ND^{D+o(D)}})$.*

Proof. Consider the unit cube U' in R^{D+1}. We use Theorem 2 (after re-scaling the cube $[N]$ to the unit cube), and we place a set W' of N points in U'.

We now define the set W. Consider a point $p' \in W'$ and let x be the value of the last coordinate of p'. We project p' into the first D-dimensions to get a point p and color it with color $i = \lfloor x n_c \rfloor$.

[3] That is, if $i = c_0 + c_1 x + c_2 x^2 + \cdots + c_{\lfloor \log_x N \rfloor + 1} x^{\lfloor \log_x N \rfloor + 1}$, then $\overline{i_x} = c_{\lfloor \log_x N \rfloor + 1} + c_{\lfloor \log_x N \rfloor} x + \cdots + c_0 x^{\lfloor \log_x N \rfloor + 1}$. Basically, we write i in base x with the most significant digits to the left and then read the digits from right to left to obtain $\overline{i_x}$.

We show that the projected points satisfy the two claims in the lemma. Claim (i) is trivial: By Theorem 2, for any two points $p, q \in W$ that were obtained from the two points $p', q' \in W'$ we have $\text{Vol}_D(\text{Box}_{D+1}(p', q')) = \Omega(\frac{1}{ND^{D+o(D)}})$. Simply observe that

$$\text{Vol}_D(\text{Box}_D(p, q)) \geq \text{Vol}_{D+1}(\text{Box}_{D+1}(p', q')) = \Omega(\frac{1}{ND^{D+o(D)}}).$$

Thus, it remains to prove claim (ii). Consider two points $p, q \in W$ with the same color that correspond to two points $p', q' \in W'$. Since p and q have the same color, it follows that the difference between the value of their D-th coordinate is at most $1/n_c$. This fact combined by Theorem 2 implies

$$\text{Vol}_D(\text{Box}_D(p, q)) \geq \text{Vol}_{D+1}(\text{Box}_{D+1}(p', q'))n_c = \Omega(\frac{n_c}{ND^{D+o(D)}}).$$

□

6.2.3 Choosing the Colors.

In the previous subsection, we discussed constructions that will help us place the parametric points. Here, we will pick the set of colors that are used to color them. First, we establish an invariant.

Invariant (I). Let $X := \frac{\log_R \tau}{t}$. We will maintain one invariant that $w_i^{(j)} \geq 0$ and $w_i^{(j)} < X$, for each $1 \leq j \leq t-1$. This invariant is to make sure that our construction is well-defined, in particular, to make sure that for each slab $\mathbf{s}_i \in \mathcal{S}$, $\tau(\mathbf{s}_i^{(j)})$, for all $1 \leq j \leq t$, are in the valid range $(0, 1]$. As a result, any tuple of $t-1$ integers that satisfy this invariant, will yield well-defined values for the thickness of the slabs used in our construction.

We will first need to estimate the total number of different colors that satisfy this invariant. Let $\mathscr{C}$ be the set of all the colors satisfying Invariant (I). In other words, $\mathscr{C}$ is the set of all $t-1$ tuples $(w_1, \ldots, w_{t-1})$ where each w_j, $1 \leq j \leq t-1$, is a non-negative integer and furthermore, $0 < R^{-w^{(j)}} \leq 1$, for $1 \leq j \leq t$ where $w^{(t)} = \log_R(\tau^{-1}) - \sum_{j=1}^{t-1} w^{(j)}$.

Observation 4. $|\mathscr{C}| \geq X^{t-1}$.

Proof. If we force $0 \leq w_j < X$, for $1 \leq j \leq t-1$, then we will have $w^{(t)} = \log_R(\tau^{-1}) - \sum_{j=1}^{t-1} w^{(j)} > 0$ and thus $0 < R^{-w^{(j)}} \leq 1$, for $1 \leq j \leq t$. Clearly, the number of tuples is at least as claimed. □

Pruning the colors. Fix an integral parameter ℓ. We call a subset $C \subset \mathscr{C}$ of ℓ colors an ℓ-subset. We say an ℓ-subset C is *bad* if by looking at the dimensions of the colors in C, we see only 2 distinct values at each dimension and C is *good* if it is not bad. Alternatively, C is good if we can find three colors $c_1, c_2, c_3 \in C$, $c_1 = (w_1^{(1)}, w_1^{(2)}, \ldots, w_1^{(t-1)})$, $c_2 = (w_2^{(1)}, w_2^{(2)}, \ldots, w_2^{(t-1)})$, and $c_3 = (w_3^{(1)}, w_3^{(2)}, \ldots, w_3^{(t-1)})$, and an index j such that $w_1^{(j)}, w_2^{(j)}$, and $w_3^{(j)}$ are all distinct. Let $\mathscr{C}_g$ be the largest subset of $\mathscr{C}$ that contains no bad ℓ-subsets (in other words, every ℓ-subset of $\mathscr{C}_g$ is good).

Lemma 4. *If $\ell > 2^{t-1}$, then $\mathscr{C}$ contains no bad ℓ-subset and thus $\mathscr{C}_g = \mathscr{C}$.*

Proof. Consider a bad ℓ-subset C. We can have at most 2 distinct values at each coordinate of the tuples in C. Therefore the number of tuples in C cannot exceed 2^{t-1}. In turn, there are no bad ℓ-subsets if $\ell > 2^{t-1}$. □

Lemma 5. *If $\ell \leq 2^{t-1}$, then $|\mathscr{C}_g| = \Omega(X^{t-1}2^{-t}\Theta(X)^{-2t/\ell})$.*

Proof. We claim that there exists a subset $\mathscr{C}' \subset \mathscr{C}$ that contains the claimed number of colors without containing any bad ℓ-subset and this clearly proves the lemma.

We prove the claim using random sampling: we take a random sample of small enough size and then remove the bad subsets.

Let p be a parameter to be determined later. Let $\mathscr{C}'$ be a subset of $\mathscr{C}$ where each color is sampled independently and with probability p. Clearly, we have $\mathbb{E}(|\mathscr{C}'|) = p|\mathscr{C}| = pX^{t-1}$. From each bad ℓ-subset, we remove one color. The set of remaining colors will be the claimed set $\mathscr{C}'$. By construction, $\mathscr{C}'$ will not contain any bad ℓ-subsets but the main point is to show that $\mathscr{C}'$ will actually retain a significant fraction of the colors.

Let $\#_{\text{bad}}$ be the total number of bad ℓ-subsets $C \in \mathscr{C}$. We first estimate $\#_{\text{bad}}$. By definition of a bad ℓ-subset, for every dimension we see only 2 distinct values among tuples in C. Thus, $\binom{X}{2}$ is the total number of ways we can choose these distinct values, at a particular dimension. After choosing the distinct values, for every tuple, every dimension has only 2 possible choices. Thus we have,

$$\#_{\text{bad}} \leq \left(\binom{X}{2}2^{\ell}\right)^{t-1}.$$

Now, consider a bad ℓ-subset $C \subset \mathscr{C}$. Observe that C survives in $\mathscr{C}'$ with probability p^{ℓ} and thus the expected number of colors that we will remove is at most $\#_{\text{bad}}p^{\ell}$. If we can choose the parameter p such that $\#_{\text{bad}}p^{\ell} \leq 1$, then we are expected to only remove one color and thus the expected number of colors left in $\mathscr{C}'$ after the pruning step is at least $pX^{t-1}/2$. Thus, we need to pick a value p such that

$$\begin{aligned}\#_{\text{bad}} \cdot p^{\ell} \leq \left(\binom{X}{2}2^{\ell}\right)^{t-1} p^{\ell} \leq 1 \Longleftarrow \\ \Theta\left(X\right)^{2t} 2^{t\ell}p^{\ell} \leq 1 \Longleftarrow \\ p = 2^{-t}\Theta\left(X\right)^{-\frac{2t}{\ell}}.\end{aligned}$$

Picking p as above, implies that the number of points left in $\mathscr{C}'$ is at least

$$X^{t-1}2^{-t}\Theta\left(X\right)^{-\frac{2t}{\ell}}.$$

□

6.2.4 The Final Construction.

We use Lemmas 4 and 5 (depending on the value of ℓ), to pick the set $\mathscr{C}_g$ of colors. Then, we use Lemma 3, where D is set to t, N is set to r and n_c is set to $|\mathscr{C}_g|$. Thus, Lemma 3 yields us a point set W. The coordinates of the i-th point ϕ_i in W defines the parametric point of $\mathbf{s}_i$ and the color of ϕ_i defines the thickness of the two-dimensional slabs that create $\mathbf{s}_i$. Thus, the set W completely defines the set $\mathcal{S}$ of r t-slabs that we aimed to build.

The last challenge is to bound the volume of the intersection of these slabs. We will do this in the remainder of this subsection.

Lemma 6. *Consider ℓ t-slabs $\mathbf{s}_1, \ldots, \mathbf{s}_\ell \in \mathcal{S}$ where the defining parameters of $\mathbf{s}_i$ are $(\alpha_i^{(1)}, \alpha_i^{(2)}, \ldots, \alpha_i^{(t)})$ and $(w_i^{(1)}, w_i^{(2)}, \ldots, w_i^{(t)})$. Then $Vol(\mathbf{s}_1 \cap \mathbf{s}_2 \cdots \cap \mathbf{s}_\ell)$ is asymptotically upper bounded by*

$$\prod_{i=1}^{t} \min_{j,j',j\neq j'} \left\{ \frac{\tau(\mathbf{s}_j^{(i)})\tau(\mathbf{s}_{j'}^{(i)})}{|\alpha_j^{(i)} - \alpha_{j'}^{(i)}|} \right\}.$$

Proof. Consider $\mathcal{Q}_i$ and observe that $\mathbf{s}_1^{(i)}, \mathbf{s}_2^{(i)}, \ldots, \mathbf{s}_\ell^{(i)}$ are slabs in $\mathcal{Q}_i$. Clearly, the region $\mathbf{s}_1^{(i)} \cap \mathbf{s}_2^{(i)} \cap \cdots \cap \mathbf{s}_\ell^{(i)}$ is contained inside every region $\mathbf{s}_j^{(i)} \cap \mathbf{s}_{j'}^{(i)}$, for $1 \leq j < j' \leq \ell$. Thus,

$$\mathrm{Vol}_2(\mathbf{s}_1^{(i)} \cap \mathbf{s}_2^{(i)} \cap \cdots \cap \mathbf{s}_\ell^{(i)}) \leq \min_{1\leq j<j'\leq \ell} \left\{ \mathrm{Vol}_2(\mathbf{s}_j^{(i)} \cap \mathbf{s}_{j'}^{(i)}) \right\}.$$

The lemma follows from Observation 3 since we have $\mathrm{Vol}_2(\mathbf{s}_j^{(i)} \cap \mathbf{s}_{j'}^{(i)}) = \frac{\tau(\mathbf{s}_j^{(i)})\tau(\mathbf{s}_{j'}^{(i)})}{|\alpha_j^{(i)} - \alpha_{j'}^{(i)}|}$. □

We now present the main result of this subsection. We recall the claim made in Lemma 1.

Lemma 1. *Consider parameters t, τ, r, R, n_c and ℓ under constraints to be specified shortly. We can build a set $\mathcal{S}$ of r t-slabs such that $\tau(\mathbf{s}) = \tau$, for every $\mathbf{s} \in \mathcal{S}$. Furthermore, (i) for any ℓ t-slabs $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_\ell \in \mathcal{S}$ and any ℓ t-slabs $\mathbf{s}'_1, \ldots, \mathbf{s}'_t$ such that $\mathbf{s}'_j$ is a translation of $\mathbf{s}_j$, we have $Vol_{2t}(\mathbf{s}'_1 \cap \mathbf{s}'_2 \cap \cdots \cap \mathbf{s}'_\ell) \leq \max\left\{ \frac{\tau^2 r t^{t+o(t)}}{n_c}, \frac{\tau^2 r t^{t+o(t)}}{R} \right\}$.*

The constraints are that $\Omega(1) \leq r \leq n/2$, $\Omega(1) \leq R \leq n^{1/(2t)}$, $2 \leq \ell < r$ and that n_c is defied as $n_c = (\frac{\log_R n}{t})^{t-1}$ when $\ell \geq 2^t$ and $n_c = \Theta((\frac{\log_R n}{t})^{t-1} 2^{-t} (\frac{\log_R n}{t})^{-t/\ell})$ when $\ell < 2^t$.

Proof. Observe that the volume of the intersection any number of t-slabs is invariant under translation. Thus, it suffices to look at the intersection of $\mathbf{s}_1, \mathbf{s}_2, \ldots,$ and $\mathbf{s}_\ell$. Let $\phi_1, \ldots, \phi_\ell$ be their (colored) parametric points. We consider a few cases.

Case I. In this case, we assume that two of the parametric points have the same color. W.l.o.g, assume the points ϕ_1, ϕ_2 have the same color. Using Lemma 3 we know that $\mathrm{Vol}_t(\mathrm{Box}_t(\phi_1, \phi_2)) = \Omega(n_c/(rt^{t+o(t)}))$ where $n_c = |\mathscr{C}_g|$. We can now bound

$$\begin{aligned}\mathrm{Vol}_{2t}(\mathbf{s}_1 \cap \mathbf{s}_2 \cap \cdots \cap \mathbf{s}_\ell) \leq \mathrm{Vol}_{2t}(\mathbf{s}_1 \cap \mathbf{s}_2) &= O\left(\frac{\tau^2}{\mathrm{Vol}_t(\mathrm{Box}_t(\phi_1, \phi_2))} \right) \\ &\leq O\left(\frac{\tau^2 r t^{t+o(t)}}{n_c} \right).\end{aligned}$$

Case II. In this case, all the ℓ colors associated to $\phi_1, \ldots, \phi_\ell$ are distinct. By construction of our set of colors, we know that the set of ℓ colors assigned to $\phi_1, \ldots, \phi_\ell$ is a good ℓ-subset. This means, w.l.o.g, we can find three parametric points $\phi_1 = (\alpha_1^{(1)}, \ldots, \alpha_1^{(t)}), \phi_2 = (\alpha_2^{(1)}, \ldots, \alpha_2^{(t)})$, and $\phi_3 = (\alpha_3^{(1)}, \ldots, \alpha_3^{(t)})$ with colors $(w_1^{(1)}, \ldots, w_1^{(t-1)}), (w_2^{(1)}, \ldots, w_2^{(t-1)})$, and $(w_3^{(1)}, \ldots, w_3^{(t-1)})$ such that there is an index $j < t$ where $w_1^{(j)}, w_2^{(j)}$, and $w_3^{(j)}$ are all distinct.

To simplify the notation, let us rename $w_1 = w_1^{(j)}$, $w_2 = w_2^{(j)}$, and $w_3 = w_3^{(j)}$, $\tau_1 = \tau(\mathbf{s}_1^{(j)})$, $\tau_2 = \tau(\mathbf{s}_2^{(j)})$, and $\tau_3 = \tau(\mathbf{s}_3^{(j)})$, and $\alpha_1 = \alpha_1^{(j)}$, $\alpha_2 = \alpha_2^{(j)}$, and $\alpha_3 = \alpha_3^{(j)}$. Remember that we have

constructed the slabs such that $\tau(\mathbf{s}_i^{(j)}) = R^{-w_i^{(j)}}$ and thus $\tau_1 = R^{-w_1}, \tau_2 = R^{-w_2}$, and $\tau_3 = R^{-w_3}$. W.l.o.g, we can assume α_1 and α_2 make the closest pair among the three values of α_1, α_2, and α_3 and that $w_1 > w_2$. We use Lemma 6:

$$\mathrm{Vol}_{2t}(\mathbf{s}_1 \cap \mathbf{s}_2 \cap \mathbf{s}_3) \leq \prod_{i=1}^{t} \min \left\{ \frac{\tau(\mathbf{s}_1^{(i)})\tau(\mathbf{s}_2^{(i)})}{|\alpha_1^{(i)} - \alpha_2^{(i)}|}, \frac{\tau(\mathbf{s}_1^{(i)})\tau(\mathbf{s}_3^{(i)})}{|\alpha_1^{(i)} - \alpha_3^{(i)}|}, \frac{\tau(\mathbf{s}_2^{(i)})\tau(\mathbf{s}_3^{(i)})}{|\alpha_2^{(i)} - \alpha_3^{(i)}|} \right\}. \tag{5}$$

To upper bound the above, we consider two further cases.

Case II.A. In this case, we assume $w_3 > w_2$. In the right hand side of Equation 5 above, for every index $i \neq j$, we pick the first term. In other words, we can write,

$$\mathrm{Vol}_{2t}(\mathbf{s}_1 \cap \mathbf{s}_2 \cap \mathbf{s}_3) \leq \min \left\{ \frac{\tau_1\tau_2}{|\alpha_1 - \alpha_2|}, \frac{\tau_1\tau_3}{|\alpha_1 - \alpha_3|}, \frac{\tau_2\tau_3}{|\alpha_2 - \alpha_3|} \right\} \cdot \prod_{i=1, i\neq j}^{t} \frac{\tau(\mathbf{s}_1^{(i)})\tau(\mathbf{s}_2^{(i)})}{|\alpha_1^{(i)} - \alpha_2^{(i)}|}.$$

However, since $|\alpha_2 - \alpha_1| \leq |\alpha_3 - \alpha_2|, |\alpha_3 - \alpha_1|$, we have:

$$\begin{aligned}
\mathrm{Vol}_{2t}(\mathbf{s}_1 \cap \mathbf{s}_2 \cap \mathbf{s}_3) &\leq \frac{\min\{\tau_1\tau_2, \tau_1\tau_3, \tau_2\tau_3\}}{|\alpha_1 - \alpha_2|} \cdot \prod_{i=1, i\neq j}^{t} \frac{\tau(\mathbf{s}_1^{(i)})\tau(\mathbf{s}_2^{(i)})}{|\alpha_1^{(i)} - \alpha_2^{(i)}|} \\
&= \min\{\tau_1\tau_2, \tau_1\tau_3, \tau_2\tau_3\} \cdot \frac{\prod_{i=1, i\neq j}^{t} \tau(\mathbf{s}_1^{(i)})\tau(\mathbf{s}_2^{(i)})}{\mathrm{Vol}_t(\mathrm{Box}_t(\phi_1, \phi_2))} \\
&= \min\{\tau_1\tau_2, \tau_1\tau_3, \tau_2\tau_3\} \cdot \frac{\frac{\tau}{\tau_1}\frac{\tau}{\tau_2}}{\mathrm{Vol}_t(\mathrm{Box}_t(\phi_1, \phi_2))}.
\end{aligned}$$

We now claim $\tau_1\tau_3 \leq \tau_1\tau_2/R$: remember that by our choice of index j, all the values w_1, w_2 and w_3 are distinct integers. Thus, $w_3 > w_2$ implies $w_3 \geq w_2 - 1$ which in turn implies $\tau_3 \leq \tau_2/R$ and thus $\tau_1\tau_3 \leq \tau_1\tau_2/R$. Using this and the fact that by Lemma 3, we have $\mathrm{Vol}_t(\mathrm{Box}_t(\phi_1, \phi_2)) = \Omega(1/(rt^{t+o(t)}))$, we get that

$$\mathrm{Vol}_{2t}(\mathbf{s}_1 \cap \mathbf{s}_2 \cap \mathbf{s}_3) \leq \frac{r\tau^2 t^{t+o(t)}}{R}.$$

Case II.B. The remaining case is $w_3 < w_2$ which combined with the assumption that $w_2 < w_1$ implies $w_3 < w_2 < w_1$. As before, since w_1, w_2, and w_3 are distinct integers, it follows that $\tau_1 \leq \tau_2/R$, and $\tau_2 \leq \tau_3/R$.

Because of the special geometry of the line, if (α_1, α_2) is the closest pair, then it follows that $|\alpha_1 - \alpha_3| = \Theta(|\alpha_2 - \alpha_3|)$.

In Eq. 5 and for every index $i \neq j$ we pick the third term inside each minimization term. In other words, we write,

$$\begin{aligned}
\mathrm{Vol}_{2t}(\mathbf{s}_1 \cap \mathbf{s}_2 \cap \mathbf{s}_3) &\leq \min \left\{ \frac{\tau_1\tau_2}{|\alpha_1 - \alpha_2|}, \frac{\tau_1\tau_3}{|\alpha_1 - \alpha_3|}, \frac{\tau_2\tau_3}{|\alpha_2 - \alpha_3|} \right\} \cdot \prod_{i=1, i\neq j}^{t} \frac{\tau(\mathbf{s}_2^{(i)})\tau(\mathbf{s}_3^{(i)})}{|\alpha_2^{(i)} - \alpha_3^{(i)}|} \\
&\leq \frac{\tau_1\tau_3}{|\alpha_1 - \alpha_3|} \cdot \prod_{i=1, i\neq j}^{t} \frac{\tau(\mathbf{s}_2^{(i)})\tau(\mathbf{s}_3^{(i)})}{|\alpha_2^{(i)} - \alpha_3^{(i)}|} \leq \frac{\tau_1\tau_3}{\Theta(|\alpha_2 - \alpha_3|)} \cdot \prod_{i=1, i\neq j}^{t} \frac{\tau(\mathbf{s}_2^{(i)})\tau(\mathbf{s}_3^{(i)})}{|\alpha_2^{(i)} - \alpha_3^{(i)}|} \\
&= \tau_1\tau_3 \cdot \frac{\frac{\tau}{\tau_2}\frac{\tau}{\tau_3}}{\Theta(\mathrm{Vol}_t(\mathrm{Box}_t(\phi_2, \phi_3)))} = \frac{\Theta(\tau^2)}{R\mathrm{Vol}_t(\mathrm{Box}_t(\phi_1, \phi_2))} \leq O\left(\frac{\tau^2 r t^{t+o(t)}}{R}\right).
\end{aligned}$$

Putting it all together. Combining all the above cases, we have shown that

$$v = \mathrm{Vol}_{2t}(\mathbf{s}_1 \cap \mathbf{s}_2 \cap \cdots \cap \mathbf{s}_m) \leq \max\left\{\frac{\tau^2 r t^{t+o(t)}}{n_c}, \frac{\tau^2 r t^{t+o(t)}}{R}\right\}.$$

□

6.3 Multilevel Reporting Lower Bound

To prove a lower bound for the multilevel reporting problem, we use the following theorem by Afshani. We need the notion of a *geometric stabbing problem*: the input is a set $\mathcal{R}$ of n geometric regions inside a D-dimensional region $\mathcal{Q}$ of volume 1. An element of $\mathcal{R}$ is called a range. The queries are points of $\mathcal{Q}$ and the output of a query q is the subset of ranges that contain q.

Theorem 3. *Assume we have a data structure for a geometric stabbing problem that uses at most $S(n)$ space and answers queries within $Q(n) + O(k)$ time in which n is the input size and k is the output size. Assume for this problem we can construct an input set $\mathcal{R}$ of n ranges such that (i) every point of $\mathcal{Q}$ is contained in at least r ranges in which r is a parameter greater than $Q(n)$ and (ii) the volume of the intersection of every α ranges is at most v, for two parameters $\alpha < r$ and v. Then, we must have $S(n) = \Omega(rv^{-1}/2^{O(\alpha)}) = \Omega(Q(n)v^{-1}/2^{O(\alpha)})$.*

Theorem 4. *Consider an algorithm $\mathcal{A}$ that given any set of n t-slabs in $\mathbb{R}^2$, builds a pointer-machine data structure $\mathcal{D}$ of size $S(n)$ that solves the MSP in $Q(n) + O(k)$ time, where k is the size of the output. In other words, given any t-point $\mathbf{p}$, the data structure can output all the input t-slabs $\mathbf{s}$ that contain $\mathbf{p}$ in $Q(n) + O(k)$. Then, $S(n) = \Omega((\frac{n}{Q(n)})^2) \cdot \frac{\left(\frac{\log(n/Q(n))}{\log\log n}\right)^{t-1}}{2^{O(2^t)}}$ as well as $S(n) = \Omega\left(\frac{n}{Q(n)}\right)^2 \Theta\left(\frac{\log(n/Q(n))}{t^{3+o(1)}\log\log n}\right)^{t-1-o(1)}$.*

Proof. We use Lemma 1, where we set the parameter $r = Q(n)$, $\tau = 2^{O(t)}r/n$, parameter ℓ to be determined, and $R = (\log n)^t$, which implies $X = \frac{\log_R(n/r)}{t} = \frac{\log(n/r)}{t^2 \log\log n}$. The lemma gives us a set $\mathcal{S}$ containing r t-slabs of thickness τ. We now create a set $\mathcal{R}$ of n t-ranges in the following way. For every $\mathbf{s}_i \in \mathcal{S}$, we create a set $\mathcal{S}_i$ containing $O(n/r)$ disjoint translations of $\mathbf{s}_i$ such that the t-slabs in $\mathcal{R}_i$ cover $\mathcal{Q}$ entirely. To be specific, consider a t-slab $\mathbf{s}_i = (\mathbf{s}_i^{(1)}, \ldots, \mathbf{s}_i^{(t)})$. We tile $\mathcal{Q}_j$ using disjoint copies of $\mathbf{s}_i^{(j)}$ to obtain a set $\mathcal{S}_{i,j}$ of two-dimensional slabs in $\mathcal{Q}_j$ (see Figure 5). The set $\mathcal{S}_i$ is the Cartesian produce of these sets, that is, $\mathcal{S}_i = \mathcal{S}_{i,1} \times \cdots \times \mathcal{S}_{i,t}$. Since each $\mathcal{S}_{i,j}$ tiles $\mathcal{Q}_j$, it follows that the set of slabs in $\mathcal{S}_i$ tile $\mathcal{Q}$. Furthermore, the number of slabs in $\mathcal{S}_{i,j}$ is $\Theta(\frac{1}{\tau(\mathbf{s}_i^{(j)})})$. This implies, the number of t-slabs in $\mathcal{S}_i$ is

$$|\mathcal{S}_i| \leq \prod_{j=1}^{t} \Theta(\frac{1}{\tau(\mathbf{s}_i^{(j)})}) = O\left(c^t \frac{1}{\tau(\mathbf{s}_i)}\right) = O\left(c^t \frac{1}{\tau}\right).$$

Since we have set $\tau = 2^{O(t)}r/n$, we can pick the constant in the exponent large enough such that $|\mathcal{S}_i| \leq n/r$.

We let $\mathcal{R} = \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_r$. By what we have just proved, $\mathcal{R}$ contains at most n t-slabs.

We look at the maximum volume, v, of the intersection of ℓ ranges $\mathbf{s}_1, \ldots, \mathbf{s}_\ell$. If two of these ℓ t-slabs are the translations of the same t-slab from $\mathcal{S}$, then by construction, the volume of their

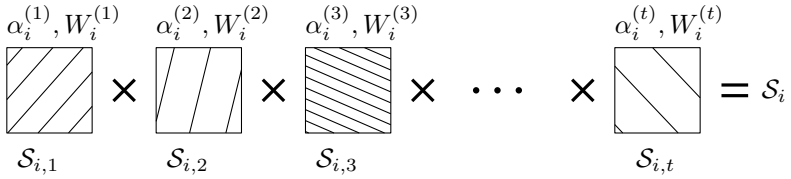


Figure 5: We tile each $\mathcal{Q}_j$, then define the set of t-slabs in our construction to be the Cartesian produce of these two-dimensional slabs.

intersection is empty. Otherwise, we obtain a bound on v by using Lemma 1. Since $n_c \leq X^{t-1}$, this implies $R \geq n_c$ and thus $\frac{r^3}{n^2 n_c} \geq \frac{r^3}{Rn^2}$ and thus Lemma 1 gives us $v \leq \frac{r^3 2^{O(t)} t^{t+o(t)}}{n^2 n_c} = \frac{r^3 t^{t+o(t)}}{n^2 n_c}$. If we pick $\ell > 2^{t-1}$ (in particular, if we set $\ell = 2^t$) we have $n_c = X^{t-1} = \left(\frac{\log(n/r)}{t^2 \log\log n}\right)^{t-1}$. Using the notation $a \gg b$ to denote $a = \Omega(b)$, Theorem 3, gives the following lower bound

$$S(n) \gg Q(n) \cdot \frac{n^2 \Theta\left(\frac{\log(n/Q(n))}{t^2 \log\log n}\right)^{t-1}}{Q(n)^3 t^{t+o(t)}} \cdot \frac{1}{2^{O(2^t)}} \gg \left(\frac{n}{Q(n)}\right)^2 \cdot \frac{\left(\frac{\log(n/Q(n))}{\log\log n}\right)^{t-1}}{2^{O(2^t)}}.$$

For small values of t (e.g., constant t), this lower bound shows that the space/query time trade-off should increase by roughly a $\log n$ factor for every increase in t. However, for larger values of t the above lower bound degrades too quickly because of the $2^{O(2^t)}$ factor so we switch to the other branch in Lemma 1. We set ℓ to be a value smaller than 2^t and obtain the bound $n_c = \Theta(X^{t-1} 2^{-t} X^{-t/\ell})^{t-1}$. Note that we can again pick $R = (\log n)^t$ which still satisfies $R \geq n_c$. Thus, we have $X := \frac{\log_R(n/r)}{t} = \frac{\log(n/r)}{t^2 \log\log n}$. In turn, we get that

$$n_c = \Theta(X^{t-1} 2^{-t} X^{-2t/\ell}) = \Theta\left(\frac{\log(n/r)}{t^2 \log\log n}\right)^{t-1} \cdot \left(\frac{\log(n/r)}{t^2 \log\log n}\right)^{-2t/\ell}.$$

This gives the lower bound

$$\begin{aligned} S(n) &\gg Q(n) \cdot \frac{n^2 \Theta\left(\frac{\log(n/Q(n))}{t^2 \log\log n}\right)^{t-1} \cdot \left(\frac{\log(n/Q(n))}{t^2 \log\log n}\right)^{-t/\ell}}{Q(n)^3 t^{t+o(t)}} \cdot \frac{1}{2^{O(\ell)}} \\ &\gg \left(\frac{n}{Q(n)}\right)^2 \frac{\Theta\left(\frac{\log(n/Q(n))}{t^{3+o(1)} \log\log n}\right)^{t-1}}{2^{O(\ell)} \cdot \left(\frac{\log(n/Q(n))}{t^2 \log\log n}\right)^{t/\ell}}. \end{aligned}$$

We now can set $\ell = \Theta(\sqrt{t \log(\frac{\log n/Q(n)}{t})})$ to balance out the two terms in the denominator. This gives us the space lower bound of

$$S(n) \gg \left(\frac{n}{Q(n)}\right)^2 \Theta\left(\frac{\log(n/Q(n))}{t^3 \log\log n}\right)^{t-1-o(1)}.$$

□

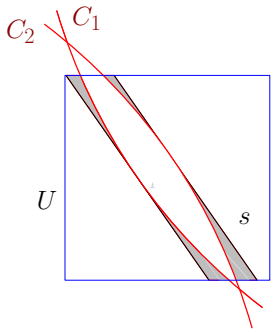


Figure 6: We approximate a slab using the intersection of two circles. Given a slab s, we can find two circles C_1 and C_2 of equal radius such that their intersection is fully inside s and when confined to the unit square U, their intersection almost covers the same area too. The symmetric difference of the slab and the intersection of the circles is shaded in grey. For any $\varepsilon > 0$, we can find the circles such that the area of the grey region is less than ε.

6.4 The Lower Bound for Fréchet Queries in 2D

We show how to use the construction from the previous section to prove the same lower bound for Fréchet queries for polygonal curves in the plane. We first consider discrete Fréchet queries as a warm up, since they are much easier to adapt our lower bound to.

6.4.1 Discrete Fréchet queries

The main idea is to simulate the phenomenon of a point stabbing a slab using a point and intersection of two equal-sized circles. In particular, we use the following observation (see Figure 6).

Observation 5. *Given a slab s and for any ε we can find a value $\rho_{\varepsilon,s}$ such that for any value $\rho \geq \rho_\varepsilon$, we can place two circles C_1 and C_2 of radius ρ in the plane such that $C_1 \cap C_2 \subset s$ and the area of their error area, $U \cap s \setminus (C_1 \cap C_2)$, is less than ε where U is the unit square.*

The idea is now very straightforward: we can replace the set of slabs used in our construction, with a set of lenses, i.e., the intersection of circles.

Fix a global parameter ε. Let $\mathcal{R}$ be the set of t-slabs used in the proof of Theorem 4. To iterate, every t-slab $\mathbf{s} \in \mathcal{R}$ is an element of the Cartesian product of t two-dimensional slabs $\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(t)}$. Using Observation 5, we can approximate every slab $s = \mathbf{s}^{(j)}$ with a lens formed by the intersection of two circles of radius at least $\rho_{\varepsilon,s}$. Let ρ_ε be the maximum value of this radius, over all slabs $\mathbf{s}^{(j)}$ and over all the t-slabs $\mathbf{s} \in \mathcal{R}$.

We create t unit square $\mathcal{Q}_1, \ldots, \mathcal{Q}_t$ and place them such that the distance between them is greater than $10\rho_\varepsilon$ (see Figure 1 on page 9; the unit squares are drawn closer in the figure for the purpose of illustration so one should imagine them far enough that the circles intersecting a unit square $\mathcal{Q}_j$, do not intersect any other unit square.). For every slab $\mathbf{s} \in \mathcal{R}$ we create a chain $c(\mathbf{s})$ of size $2t$, by connecting the centers of the circles that give rise to the lens that approximates $\mathbf{s}^{(j)}$. A query t-point $\mathbf{p}$ is simply represented by another chain $q(\mathbf{p})$ that connects the points $\mathbf{p}^{(j)}$, $1 \leq j \leq t$. See Figure 1.

The only issue we are left with is that the lenses only approximate the slabs, meaning, there will be t-points $\mathbf{p} \in \mathcal{Q}$ that behave differently with respect to the slabs compared to the lenses. Inside

every unit square $\mathcal{Q}_j$, we have created n lenses such that the error area of each lens is at most ε. Thus, the error area of all the slabs created inside $\mathcal{Q}_j$ is at most $n\varepsilon$. Over all unit squares $\mathcal{Q}_j$, this error area is $nt\varepsilon$. Remember that we had defined $\mathcal{Q} = \mathcal{Q}_1 \times \cdots \times \mathcal{Q}_t$. Define $\mathcal{Q}'$ as the subset of $\mathcal{Q}$ that includes all the t-points $\mathbf{p}$ such that none of the points $\mathbf{p}^{(j)}$ is inside an error area. By what we have observed, $\text{Vol}_{2t}(\mathcal{Q}') \geq 1 - nt\varepsilon$. By picking ε small enough, we can ensure that $\text{Vol}_{2t}(\mathcal{Q}') \geq 1/2$.

For a unit square $\mathcal{Q}_j$, we have placed all the centers of the circles that create the slabs inside $\mathcal{Q}_j$, within distance of ρ_ε of $\mathcal{Q}_j$. Since we have placed the unit squares $\mathcal{Q}_j$ far apart, it means that a point of $q(\mathbf{p})$ inside $\mathcal{Q}_j$ can only be matched to the centers of the circles that create the slabs inside $\mathcal{Q}_j$. Thus, a query t-point $\mathbf{p} \in \mathcal{Q}'$ is inside a t-slab $\mathbf{s}$ if and only the chain $q(\mathbf{p})$ is within discrete Fréchet distance ρ_ε of the chain $c(\mathbf{s})$.

Observe that in the framework of Afshani (Theorem 3), the region $\mathcal{Q}$ is the set of all possible queries and it is only required to have volume one. To finish off, we rescale $\mathcal{Q}$ and all the slabs used in our construction by a constant factor such that the volume of $\mathcal{Q}'$ equals one. Then, we apply the framework to the set of lenses (i.e., t-lenses) instead of t-slabs. Consider the requirement (i) in Theorem 3. The construction in the previous section ensures that for every t-points $\mathbf{p} = (\mathbf{p}^{(1)}, \ldots, \mathbf{p}^{(t)}) \in \mathcal{Q}'$, there are r t-slabs $\mathbf{s}_1, \ldots, \mathbf{s}_r$ that contain $\mathbf{p}$. Observe that this directly implies the existence of r t-lenses that contain $\mathbf{p}$ because $\mathbf{p}^{(j)}$ is not contained in any error region, for all $1 \leq j \leq t$. The requirement (ii) is trivially satisfied since lenses are created to be subsets of their corresponding slabs, meaning, the volume of an intersection of lenses will have a smaller volume than the intersection of their corresponding slabs.

Thus, the lower bound of Theorem 4 also applies to discrete Fréchet queries where the input chains have complexity $2t$ and the query curves have complexity t.

6.4.2 The continuous case

The construction in the previous subsection does not apply to the continuous Fréchet case. The main problem here is that unlike the discrete case, it is not required for vertices of the query chain to be mapped to the vertices of the input chain. As a result, an input chain $c(\mathbf{s})$ may match a query $q(\mathbf{p})$ even though the t-point $\mathbf{p}$ is not contained in the t-slab $\mathbf{s}$.

To resolve this issue, we describe a construction of input curves that will take the role of the $t-$slabs and we define a suitable set of query curves. Our construction does not vary the radius of the queries, we set the radius to 1.

In the following, a polygonal curve is implicitly defined by a sequence of vertices. To obtain the explicit curve, consecutive vertices need to be linearly interpolated. The Cartesian product of two sets of polygonal curves simply concatenates the sequence of vertices thereby effectively inserting the line segment that connects the endpoints of the corresponding curves.

Zig-zag gadget Our input construction consists of concatenatenations of basic gadgets which we call *zig-zag gadgets* and which are described as follows. The gadget is constructed using parameters $x_1, x_2, x_3 \in [-1, 1]$. It is a polygonal curve with four vertices p_1, p_2, p_3, p_4 defined as follows

$$\pi(x_1, x_2, x_3) = \left((0, -4), \left(x_1 + \cos(\theta), \frac{x_2 + x_3}{2}\right), \left(x_1 - \cos(\theta), \frac{x_2 + x_3}{2}\right), (0, 4)\right)$$

with θ defined by the equality $\sin(\theta) = \frac{|x_2 - x_3|}{2}$. An example is depicted in Figure 7. Note that the two interior vertices p_2 and p_3 of the curve are chosen such that the two unit circles centered at p_2 and p_3 intersect on the vertical line segment from (x_1, x_2) to (x_1, x_3).

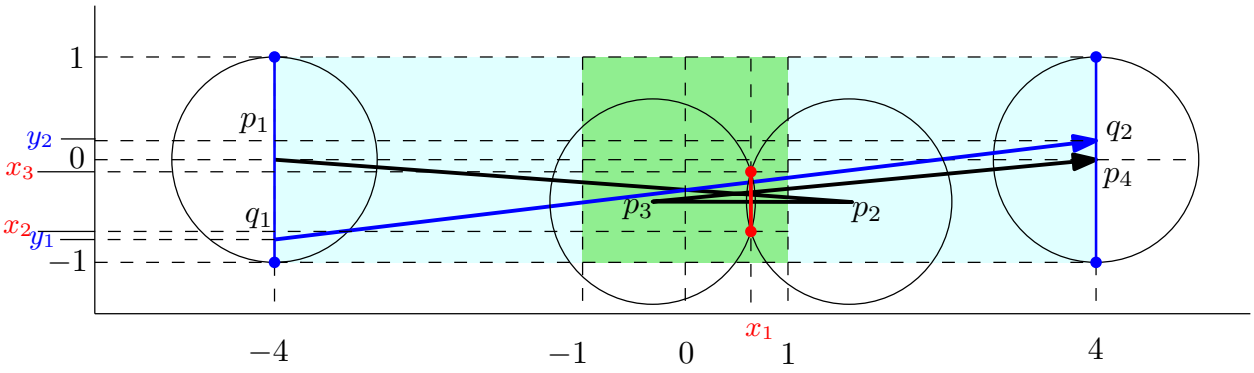


Figure 7: Zig-zag gadget given by p_1, p_2, p_3 and p_4 (black curve) and query edge given by q_1 and q_2 (blue edge). Any query edge that outputs the zigzag gadget needs to intersect the vertical interval bounded by the two points (x_1, x_2) and (x_1, x_3).

Queries To simplify our analysis we will restrict the set of queries to polygonal curves that have odd vertices on the vertical line at -4 and even vertices on the vertical line at 4. We define the set of queries $\mathcal{Q} = \mathcal{Q}_1 \times \mathcal{Q}_2 \times \ldots \mathcal{Q}_t$, where $\mathcal{Q}_i$ is the set of left-to-right line segments with vertices $q_1 = (-4, y_1)$ and $q_2 = (4, y_2)$ for $y_1, y_2 \in [-1, 1]$. Each such query can be represented by an ordered set of lines $\ell_1, \ldots, \ell_t$, such that ℓ_i is the line supporting the ith left-to-right edge of the query. Each line $\ell : y = ax + b$ can be represented as a point $p_\ell = (a, b)$ in the dual space of lines. We intend to use the volume argument of Theorem 3 in this dual space.

Observation 6. *The set $\mathcal{Q}_i$ in the dual space forms a parallelogram of area $1/2$.*[4] *This follows from the fact that a line ℓ supports a line segment in $\mathcal{Q}_i$ if and only if it intersects the two vertical intervals that define the set $\mathcal{Q}_i$. In the dual space this corresponds to the intersection of two slabs bounded by the lines $y = 4x + 1$, $y = 4x - 1$, $y = -4x + 1$ and $y = -4x - 1$.*

Lemma 7. *For any zig-zag gadget $p = \pi(x_1, x_2, x_3)$ with $x_1, x_2, x_3 \in [-1, 1]$ and $|x_2 - x_3| \leq 1$ and any line segment $q \in \mathcal{Q}_i$ we have $d_F(p, q) \leq 1$ if and only if q is supported by a line ℓ, whose dual p_ℓ lies in the slab that has slope x_1 and intersects the y-axis in the interval $[x_2, x_3]$.*

Proof. In the following, we refer to the intersection of the unit disks that are centered at p_2 and p_3 as the *lens* and we denote with $\mathcal{I}$ the vertical interval formed by the two intersection points (x_1, x_2) and (x_1, x_3). Guibas *et al.* [14] proved that for a line segment q it holds that $d_F(p, q) \leq 1$ if and only if q stabs the unit disks centered at the vertices of p in the order along p. In the lemma by Guibas *et al.*, the ordered stabbing requires a sequence of points on q to exist, in the order in which they appear on q, such that the ith point lies inside or on the boundary of the ith disk centered at the vertices along the curve p. We claim that q is an ordered stabber in this sense if and only if it intersects the interval $\mathcal{I}$. Since $q \in \mathcal{Q}_i$, it must be that it intersects the disk at p_1 and p_4 in the right order. This also implies that the slope of q lies in the interval $[-\frac{1}{4}, \frac{1}{4}]$ and that q is directed from left to right. Therefore, q needs to stab the disks at p_2 and p_3 in the lens. Since we ensured $|x_2 - x_3| \leq 1$, the slope of a line that stabs the lens outside $\mathcal{I}$ is either larger or equal $\sqrt{3}$ or smaller or equal $-\sqrt{3}$, thus the range of slopes of such lines is disjoint from the range of slopes of query

[4]Technically speaking, in order to obtain a set of queries with area 1 the space of queries needs to be scaled by a factor 2. However, this scaling does not affect our asymptotic bounds on the volumes.

line segments in $\mathcal{Q}_i$. Therefore, q intersects $\mathcal{I}$ if and only if $d_F(p,q) \leq 1$. Now, the set of lines that intersects $\mathcal{I}$ corresponds to the set of points in the dual space that lies in the intersection of two parallel halfspaces bounded by the lines $y = x_1x + x_2$ and $y = x_1x + x_3$. This is the slab with slope x_1 which intersects the y-axis in the interval $[x_2, x_3]$. □

Input As in the previous section, we build r different input sets $\mathcal{S}_1, \mathcal{S}_2, \ldots, S_r$ such that: (i) each $\mathcal{S}_i$ contains $\Theta(n/r)$ input curves (ii) for any two input curves $\mathbf{s}_1, \mathbf{s}_2 \in \mathcal{S}_i$, the set of queries that contains $\mathbf{s}_1$ is disjoint from the set of queries that contains $\mathbf{s}_2$. The set $\mathcal{S}_i$ is defined as the Cartesian product of the sets $\mathcal{S}_{i,1} \times S_{i,2} \times \ldots S_{i,t}$, where each $\mathcal{S}_{i,j}$ is a set of zig-zag gadgets defined by parameters $\alpha = \alpha_i^{(j)}$ and $W = W_i^{(j)}$; W here will play the role of the thickness.

We define a series of zig-zag gadgets with indices $1 \leq i' \leq \lceil \frac{2}{W} \rceil$: $\pi_{i'} = \pi(x_1^{i'}, x_2^{i'}, x_3^{i'})$ with

$$\begin{aligned} x_1^{i'} &= \tan(\alpha) \\ x_2^{i'} &= (i'-1)W - 1 \\ x_3^{i'} &= i'W - 1 \end{aligned}$$

The elements of this series form the set $\mathcal{S}_{i,j}$.

Following Lemma 7, we call the slab with slope $x_1^{(i')}$ and y-intercept $[x_2^{(i')}, x_3^{(i')}]$ the *dual slab* of the corresponding zig-zag gadget $\pi_{i'}$. Furthermore, we call the Cartesian product of t dual slabs, each corresponding a zig-zag gadget of an element of $\mathcal{S}_{i,j}$, a *dual t-slab* of the corresponding element of $\mathcal{S}_i$. Note that Lemma 7 puts some conditions on the parameters α and W. In order to use the described representation in the dual space we need that $-1 \leq x_1 \leq 1$, which translates to $-\frac{\pi}{4} \leq \alpha \leq \frac{\pi}{4}$, and we need $|x_2 - x_3| \leq 1$ which translates to $0 < W \leq 1$. However, this does not prevent us from using the lower bound construction from the previous section as long as the range of angles for α is constant.

Observation 7. *The angle of a dual slab of a zig-zag gadget in $\mathcal{S}_{i,j}$ is equal to $\alpha_i^{(j)}$ and the width of this slab is at most $W_i^{(j)}$.*

Observation 8. *The number of dual t-slabs in $\mathcal{S}_i$ is $\Theta(n/r)$.*

Lemma 8. *For any two input curves $\mathbf{s}_1, \mathbf{s}_2 \in \mathcal{S}_i$, the volume of the intersection of their corresponding dual t-slabs is zero.*

Proof. Note that the vertical intervals that define the zig-zag gadgets in $S_{i,j}$ tile the section of the vertical line at $x_1^{i'}$ which lies between the horizontal lines at -1 and 1. It follows, by Lemma 7, that the zig-zag gadgets of $S_{i,j}$ tile the set $\mathcal{Q}_i$ in the dual space using a series of pairwise disjoint slabs. Therefore, the query curves of entire $\mathcal{Q}$ are partitioned by the t-slabs of the set of query curves of S_i such that the volume of the intersection of any two dual t-slabs is zero. □

Thus, by combing the two constructions that we presented in this and the previous section, we have proved the following theorem.

Theorem 5. *Assume we have built a data structure for a given set $\mathcal{S}$ of n polygonal curves of size t, and a fixed radius ρ, such that for any query polygonal chain q of size t, we can find all the input curves within the continuous or discrete Fréchet distance ρ of q, in $Q(n) + O(k)$ time where k is the size of the output.*

Then, $S(n) = \Omega((\frac{n}{Q(n)})^2) \cdot \frac{\left(\frac{\log(n/Q(n))}{\log\log n}\right)^{t-1}}{2^{O(2^t)}}$ as well as $S(n) = \Omega\left(\frac{n}{Q(n)}\right)^2 \Theta\left(\frac{\log(n/Q(n))}{t^{3+o(1)}\log\log n}\right)^{t-1-o(1)}$.

7 A Data Structure

In this section, we will focus on building data structures to perform range searching on polygonal curves based on the notion of Fréchet distance. Our data structures will ultimately use the recent results on semialgebraic range searching, however, getting to the point where we can do that is non-trivial, specially for the continuous Fréchet queries.

Our data structures have two components: one component that is based on recent results on semialgebraic range searching and the second component that focuses on the Fréchet distance and tries to break down the data structure problem into sub-problems that are instances of semialgebraic range searching. Among these, the first component is very standard and not particularly interesting for the expert reader. The second component is where our contributions lie.

In the next subsection, we will briefly go over the standard existing techniques in range searching, combine them with the new results on semialgebraic range searching and show how we can build multilevel data structures that can handle more complex semialgebraic input and query objects. In the two subsequent chapters, we will consider the discrete and continuous Fréchet queries.

7.1 Multi-level Semialgebraic Range Searching

We will use the following recent result from semialgebraic range searching. Before stating the theorem, we will quickly cover some of the related definitions. By $\mathbb{R}_D[x_1, \ldots, x_d]$ we denote the set of all d-variate polynomials of degree at most D (on variables $x_1, \ldots, x_d$). For a polynomial $h \in \mathbb{R}_D[x_1, \ldots, x_d]$, we denote the set of zeros of h with $Z(h)$. In other words, $Z(h) = \big\{(x_1, \ldots, x_d) \in \mathbb{R}^d \mid h(x_1, \ldots, x_d) = 0\big\}$. For a given set P of points, we say $Z(h)$ crosses P if $Z(h)$ intersects any connected subset of $\mathbb{R}^d$ that contains P. A semialgebraic set is a subset of $\mathbb{R}^d$ that satisfies some, n_1, number of polynomial inequality of some degree, n_2, and using logical operands $\wedge, \vee$, and $\neg$.

Theorem 6 (The Semialgebraic Partition Theorem)**.** *Let P be a set of n points in $\mathbb{R}^d$ and let r be a parameter. There exists a constant K that only depends on d such that the following hold.*

We can find d integers $r \leq r_1, \ldots, r_d \leq r^K$ such that we can partition P into subsets $P = P^* \bigcup_{i=1}^{d} \bigcup_{j=1}^{t_i} P_{ij}$ in which P^* contains at most r^K points, each P_{ij} contains at most n/r_i points, $t_i = r^{O(1)}$, and crucially, for any d-variate polynomial $h \in \mathbb{R}_D[x_1, \ldots, x_d]$ where D is another constant $Z(h)$ intersects at most $O(r_i^{1-1/d})$ of the subsets $P_{i,1}, \ldots, P_{i,t_i}$.

Furthermore, each subset P_{ij} is contained in a semialgebraic set Δ_{ij} that is defined by at most $O(r^{O(1)})$ polynomial inequalities of degree $O(r^{O(1)})$. For any d-variate polynomial $h \in \mathbb{R}_D[x_1, \ldots, x_d]$ $Z(h)$ intersects at most $O(r_i^{1-1/d})$ of the subsets $\Delta_{i,1}, \ldots, \Delta_{i,t_i}$.

Using the semialgebraic partition theorem, we can solve the following multilevel semialgebraic range searching problem. The input is a set $\mathscr{P}$ n of t-points in $\mathbb{R}^d$. The query is tuple of t semialgebraic sets $(\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_t)$, where each semialgebraic set is defined by a constant number of polynomial inequalities of constant degree, and the goal is to find all the points $\mathbf{p} = (\mathbf{p}^{(1)}, \ldots, \mathbf{p}^{(t)}) \in \mathscr{P}$ such that the point $\mathbf{p}^{(i)}$ is contained in $\boldsymbol{\psi}_i$, for $1 \leq i \leq t$; say that such a t-point $\mathbf{p}$ is contained in the tuple $(\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_t)$.

As mentioned above, using the semialgebraic partition tree, and using classical techniques, we can prove the following theorem. The proof is included for completeness and also because of the fact that the existing literatures do not explicitly mention such a data structure.

Theorem 7. *Let $\mathscr{P}$ be a set of n t-points in $\mathbb{R}^d$. We can store $\mathscr{P}$ in a data structure using $O(nO(\log\log n)^{t-1})$ space that can answer the following queries. Given a tuple of t semialgebraic sets $(\boldsymbol{\psi}_1,\ldots,\boldsymbol{\psi}_t)$ where each $\boldsymbol{\psi}_i$ is a semialgebraic set determined by a constant number of polynomial inequalities of constant degree, we can output all the t-points $\mathbf{p}$ such that $\mathbf{p}$ is contained in $(\boldsymbol{\psi}_1,\ldots,\boldsymbol{\psi}_t)$. The query time is $O(n^{1-1/d}\log^{O(t)} n + k)$ where k is the size of the output.*

The rest of this section is devoted to the proof of the above theorem. We start with the description of the data structure.

The Data Structure. We will describe a data structure $\mathcal{D}(\mathscr{P})$ that is a multilevel data structure based on the Semialgebraic Partition Theorem. Let P_1 be the set of first points of all the t-points in $\mathscr{P}$. We use the Semialgebraic Partition Theorem with P set to P_1 and with parameter r set to n^{ε} for small enough constant ε to be determined later. This partitions P_1 into subsets $P_1 = P^* \bigcup_{i=1}^{d}\bigcup_{j=1}^{t_i} P_{ij}$. We call these subsets "canonical sets". Let $\mathscr{P}_{ij}$ be the canonical set that contains t-points whose first point is in the set P_{ij}. Let $\mathscr{P}'_{ij}$ be the set of $(t-1)$-points obtained by removing the first point of every t-point in $\mathscr{P}_{ij}$. Note that if $t=1$, then $\mathscr{P}'_{ij}$ is an empty set. For every subset $\mathscr{P}_{ij}$ we do two different kinds of recursion. Our first recursion is to build $\mathcal{D}(\mathscr{P}_{ij})$. Our second recursion is to build $\mathcal{D}(\mathscr{P}'_{ij})$. Our recursion stops as soon as $\mathscr{P}$ contains a constant number of points.

Space Analysis. We first analyze the space complexity of the data structure. Let $\mathscr{S}_k(n)$ be the space complexity of the data structure if it is run on an input of n k-points. Our goal is to estimate $\mathscr{S}_t(n)$. We have

$$\mathscr{S}_k(n) = |P^*| + \sum_{i=1}^{d}\sum_{j=1}^{t}\mathscr{S}_k(|S_{ij}|) + \sum_{i=1}^{d}\sum_{j=1}^{t}\mathscr{S}_{k-1}(|S'_{ij}|).$$

It is easy to see that $\mathscr{S}_1(n) = O(n)$ since the second recursion step do not happen if $t=1$ and thus each point is only stored in one sub-problem. We guess that $\mathscr{S}_k(n)$ solves to $O(nO(\log\log n)^{k-1})$ and try to prove this with induction. Thus, we can re-write the recursion as

$$\mathscr{S}_k(n) = |P^*| + \sum_{i=1}^{d}\sum_{j=1}^{t}\mathscr{S}_k(|S_{ij}|) + O(nO(\log\log n)^{k-2}).$$

Note that by the choice of r, each set $\mathscr{P}_{ij}$ has size at most $n/r_i \leq n/r = n^{1-\varepsilon}$. Thus, there are $O(\log\log n)$ levels of recursion. Observe that at each recursion level we are dealing with disjoint set of subproblems. This means that $S_k(n) = O(nO(\log\log n)^{k-1})$.

The Query Algorithm. Consider a query tuple $\boldsymbol{\psi}$ of t semialgebraic sets $\boldsymbol{\psi}_1,\ldots,\boldsymbol{\psi}_t$. Let $\mathscr{P}_{\boldsymbol{\psi}}$ be the set of t-points in $\mathscr{P}$ that satisfy the query, i.e., $\mathscr{P}_{\boldsymbol{\psi}}$ contains all the points $\mathbf{p} = (\mathbf{p}^{(1)},\ldots,\mathbf{p}^{(t)}) \in \mathscr{P}$ such that the point $\mathbf{p}^{(i)}$ is contained in $\boldsymbol{\psi}_i$, for $1 \leq i \leq t$. Let $\mathscr{P}_{\boldsymbol{\psi},1}$ be the set of t-points in $\mathscr{P}$ such that only $\mathbf{p}^{(1)}$ is contained in $\boldsymbol{\psi}_1$. Clearly, $\mathscr{P}_{\boldsymbol{\psi}} \subset \mathscr{P}_{\boldsymbol{\psi},1}$. Our goal is to find the set $\mathscr{P}_{\boldsymbol{\psi},1}$ as the disjoint union of a number of canonical sets, that is, to find a set $C_{\boldsymbol{\psi}_1}$ of canonical sets such that $\mathscr{P}_{\boldsymbol{\psi},1} = \bigcup_{c\in C_{\boldsymbol{\psi}_1}} c$. We use the data structure $\mathcal{D}(\mathscr{P})$. Remember that in the data structure $\mathcal{D}(\mathscr{P})$ we have partitioned P_1 (the set of first points of $\mathscr{P}$) into subsets $P_1 = P^* \bigcup_{i=1}^{d}\bigcup_{j=1}^{t_i} P_{ij}$. We explicitly process the t-points by looking at all the points in P^* (that is, if we are solving a range reporting

variant, we output them all, or if we are solving a semigroup variant, we add up all the weights corresponding to t-points of P^*). Next, we process the subsets $P_{i1}, P_{i2}, \ldots, P_{it_i}$ starting from $i = 1$. By the Semialgebraic Partition Theorem, each P_{ij} is contained in a semialgebraic set Δ_{ij} and that each polynomial defining the set $\boldsymbol{\psi}_1$ intersects only $O(r_i^{1-1/d})$ of the sets $\Delta_{i1}, \Delta_{i2}, \ldots, \Delta_{it_i}$. Thus, the polynomial defining the set $\boldsymbol{\psi}_1$ intersect at most $O(r_i^{1-1/d})$ sets. We go through all the sets $\Delta_{i1}, \Delta_{i2}, \ldots, \Delta_{it_i}$ and for each Δ_{ij} determine (case a) if Δ_{ij} is completely outside $\boldsymbol{\psi}_1$ (in which case we ignore it), or (case b) Δ_{ij} is completely inside $\boldsymbol{\psi}_1$ (in which case we add P_{ij} as a canonical set to $C_{\delta'}$) or (case c) if Δ_{ij} intersects the boundary of $\boldsymbol{\psi}_1$ and in this case we recurse on the data structure $\mathcal{D}(\mathscr{P}_{ij})$. Since each Δ_{ij} is determined by $r^{O(1)}$ polynomials of degree $r^{O(1)}$, and $t_i = r^{O(1)}$, these tests will take $r^{O(1)}$ time in total. By the end of the recursion, we will have the desired set $C_{\boldsymbol{\psi}_1}$.

We have following recursion to describe the number of canonical sets $f(n)$ placed in the set $C_{\boldsymbol{\psi}_1}$.

$$f(n) = r^{O(1)} + \sum_{i=1}^{d} O(r_i^{1-1/d}) f(n/r_i).$$

Note that we have $r = n^{\varepsilon}$ and that $r_i \geq r$. This is a standard recursion in the range searching area and it is not too difficult to see that is solves to $f(n) = O(n^{1-1/d} \log^{O(1)} n)$.

Having computed an implicit representation of $C_{\boldsymbol{\psi}_1}$, we do the following. Remember that for every canonical set $c \in C_{\boldsymbol{\psi}_1}$, we have build another data structure $\mathcal{D}(c')$ where c' is the set of $(t-1)$-points obtained by removing the first point of the t-points in c. Any t-point $\mathbf{p}$ represented by the canonical sets in $C_{\boldsymbol{\psi}_1}$ has the property that the point $\mathbf{p}^{(1)}$ is contained in $\boldsymbol{\psi}_1$. Thus, it remains to narrow the search such that $\mathbf{p}^{(i)}$ is also contained in $\boldsymbol{\psi}_i$ for $2 \leq i \leq t$. However, this is exactly equivalent to searching for points $\mathbf{p}' = (\mathbf{p}^{(2)}, \ldots, \mathbf{p}^{(t)})$ using the $t-1$ query tuples $\boldsymbol{\psi}_2, \ldots, \boldsymbol{\psi}_t$. Thus, we can simply recurse on each c' (using $\mathcal{D}(c')$) for every $c \in C_{\boldsymbol{\psi}_1}$. Let $f_k(n)$ be the total number of canonical sets obtained after having recursed on a set containing n k-points. We have the following recursion.

$$f_{k+1}(n) \leq r^{O(1)} + \sum_{i=1}^{d} \sum_{j=1}^{t_i} f_k(|P_{ij}|) + \sum_{i=1}^{d} O(r_i^{1-1/d}) f_{k+1}(n/r_i).$$

We guess that $f_k(n) = O(n^{1-1/d} \log^{Ck} n)$ for a constant C. Since the sets P_{ij} form a partition of the set P and they contain at most n t-points in total, the recursion simplifies to

$$f_{k+1}(n) \leq r^{O(1)} + O(n^{1-1/d} \log^{Ck} n) + \sum_{i=1}^{d} O(r_i^{1-1/d}) f_{k+1}(n/r_i).$$

Using the standard analysis from the range searching literature, it is not too difficult to show that by picking C large enough we get $f_{k+1} = O(n^{1-1/d} \log^{C(k+1)} n)$.

7.2 Discrete Frechet Queries

Let S be a set of n polygonal chains in $\mathbb{R}^d$ where each chain $s \in S$ contains at most t_s vertices. For simplicity, we can assume every chain contains exactly t_s vertices (by adding extra dummy vertices). Consider a query polygonal chain q of size t_q and a chain $s \in S$. Imagine we would like to determine if the discrete Frechet distance between q and s is at most ρ, for some parameter ρ. This can be done using the so-called *free-space-matrix*, which can be described as follows. Let $M_{q,s}$ be the $0-1$-matrix with t_q rows and t_s columns, where the entry $m(i,j)$ at row i and column j of

$M_{q,s}$ is 0 if the distance between the i-th vertex of q and the j-th vertex of s is greater than ρ and 1 otherwise. Testing if the Fréchet distance between s and q is at most ρ now amounts to testing if there exists an xy-monotone path connecting $m(1,1)$ to $m(t_q, t_s)$ that passes through the 1 entries. We treat each input chain s as a t_s-point and build the data structure from the previous subsection.

We now describe the query procedure. Let q be the query chain of t_q vertices. Consider spheres of radius ρ centered on the vertices of q. Let $\mathcal{A}$ be the arrangement created by the spheres. It is easy to see that the complexity of $\mathcal{A}$ is $O(t_q^{d+1})$ by just lifting them to halfspaces in $\mathbb{R}^{d+1}$. Now, consider a chain $s \in S$ and the corresponding free-space matrix $M = M_{q,s}$. Every column of M corresponds to a region in the arrangement $\mathcal{A}$. In other words, there are at most $t_q^{O(d)}$ 0-1 vectors could possibly appear as a column in matrix M. This in turn implies that the total number of matrices that can be the free-space matrix of some chain in S is upper bounded by $t_q^{O(n_i)}$. Let $\mathscr{M}$ be the set of these matrices. We can compute $\mathscr{M}$ easily in $t_q^{O(n_i)}$ time.

During the query time, we will go through the following stages. First, we generate the set of matrices in $\mathscr{M}$. For every matrix $M \in \mathscr{M}$, we will only output chains s with $M_{q,s} = M$. Clearly, this will output all the valid chains since $\mathscr{M}$ contains all the possible valid free-space diagrams and it will not produce any duplicates since $M_{q,s}$ is unique. Note that this blows up the query time by a $t_q^{O(t_s)}$ factor. Thus, in the second stage, we have a fixed matrix $M \in \mathscr{M}$ and we would like to output the set of chains s such that $M_{q,s} = M$. To do that, we "triangulate" $\mathcal{A}$: we lift the arrangement of spheres into $d+1$ dimension and triangulate the resulting arrangement of halfspaces, and then project back to $\mathbb{R}^d$. This corresponds to decomposing $\mathcal{A}$ into $O(t_q^{d+1})$ cells where each cell is a semialgebraic set determined by a constant number of polynomials of degree two. Let $\mathcal{A}'$ be the resulting triangulation. Consider a chain s such that $M_{q,s} = M$ and consider the i-th column v_i of M. The bit vector v_i encodes exactly which points of q are within distance r of the i-th vertex of s. In other words, the bit vector v_i identifies a unique cell δ_i in the arrangement $\mathcal{A}$ such that the i-th vertex of s must be contained in that cell. We are now almost done. Ideally, we would like to issue one query $(\delta_1, \ldots, \delta_{t_s})$ to find exactly what we want. However, the semialgebraic set δ_i might not be made using a constant number of polynomial inequalities. So we simply switch to the triangulated arrangement $\mathcal{A}'$. Let $\delta_{i,1}, \ldots, \delta_{i,x_i}$ be the set of cells formed in $\mathcal{A}'$ from triangulating δ_i where each cell is a semialgebraic cell formed by a constant number of polynomial inequalities of constant degree. We now form $\prod_{i=1}^{t_s} x_i$ queries by creating the Cartesian product of these cells, that is, $\{\delta_{1,1}, \ldots, \delta_{1,x_1}\} \times \{\delta_{i,2}, \ldots, \delta_{2,x_2}\} \times \left\{\delta_{i,t_s}, \ldots, \delta_{i,x_{t_s}}\right\}$.

Putting all these together, we can bound the total query time with

$$O(n^{1-1/d} \log^{O(t_s)} n \cdot t_q^{O(t_s)} \cdot t_q^{O(d)}) = O(n^{1-1/d} \log^{O(t_s)} n \cdot t_q^{O(t_s)})$$

assuming $t_q = O(\log^{O(1)} n)$.

Theorem 8. *Given a set S of n polygonal curves in $\mathbb{R}^d$ where each curve contains t_s vertices, we can store S in a data structure of $\mathcal{O}\big(n(\log\log n)^{t_s-1}\big)$ size such that given a query polygonal chain of size t_q and a parameter ρ, it can output all the input curves within discrete Fréchet distance of ρ to the query in $\mathcal{O}\Big(n^{1-1/d} \cdot \log^{O(t_s)} n \cdot t_q^{\mathcal{O}(d)}\Big)$, assuming $t_q = \log^{O(1)} n$.*

7.3 Continuous Frechet Queries

Let S be a set of polygonal chains as before. We now consider the range-searching problem for the continuous Fréchet distance. Consider a query polygonal chain q of size t_q and a chain $s \in S$.

Imagine we would like to determine if the Fréchet distance between q and s is at most ρ, for some parameter ρ. This can be done using the so-called *free-space diagram* which is a continuous version of the free-space matrix.

Free-space diagram We can interpret the polygonal chains s and q as continous curves $s : [0,1] \to \mathbb{R}^2$ and $q : [0,1] \to \mathbb{R}^2$ by linearly interpolating consecutive vertices of the chain. Consider the parametric space $[0,1] \times [0,1]$ of the two curves. The vertices of the curves partition this parametric space into rectangular cells, such that each cell corresponds to the parametric space of two edges, one from each curve. The *free-space* is the subset of points $(x,y) \in [0,1] \times [0,1]$ such that $\|s(x) - p(y)\| \leq \rho$. The free-space within each cell can be described as an ellipse clipped to the cell and is therefore convex. Now, testing if the Fréchet distance of the two curves is smaller or equal to r amounts to testing if there exists a (x,y)-monotone path that starts at $(0,0)$ and ends at $(1,1)$ and stays inside the free-space. We call such a path *feasible*.

7.3.1 High-level Predicates

We would like to encode reachability in the free-space diagram combinatorially using a small set of predicates. This will help us to build an efficient data structure for the range-reporting problem. We denote the vertices of s with $s_1, \ldots, s_{t_s}$ and the vertices of q with $q_1, \ldots, q_{t_q}$.

(P_1) *(Endpoints (start))* This predicate returns true if and only if $\|s_1 - q_1\| \leq \rho$

(P_2) *(Endpoints (end))* This predicate returns true if and only if $\|s_{t_s} - q_{t_q}\| \leq \rho$

(P_3) *(Vertex-edge (horizontal))* Given an edge of s, $\overline{s_j s_{j+1}}$, and a vertex q_i of q, this predicate returns true iff there exist a point $p \in \overline{s_j s_{j+1}}$, such that $\|p - q_i\| \leq \rho$.

(P_4) *(Vertex-edge (vertical))* Given an edge of q, $\overline{q_i q_{i+1}}$, and a vertex s_j of s, this predicate returns true iff there exist a point $p \in \overline{q_i q_{i+1}}$, such that $\|p - s_j\| \leq \rho$.

(P_5) *(Monotonicity (horizontal))* Given two vertices of s, s_j and s_k with $j < k$ and an edge of q, $\overline{q_i q_{i+1}}$, this predicate returns true if there exist two points p_1 and p_2 on the *line* supporting the directed edge, such that p_1 appears before p_2 on this line, and such that $\|p_1 - s_j\| \leq \rho$ and $\|p_2 - s_k\| \leq \rho$.

(P_6) *(Monotonicity (vertical))* Given two vertices of q, q_i and q_k with $i < k$ and an directed edge of s, $\overline{s_j s_{j+1}}$, this predicate returns true if there exist two points p_1 and p_2 on the *line* supporting the directed edge, such that p_1 appears before p_2 on this line, and such that $\|p_1 - q_i\| \leq \rho$ and $\|p_2 - q_k\| \leq \rho$.

Lemma 9. *Given the truth values of all predicates (P_1)-(P_6) of two curves s and q for a fixed value of ρ, one can determine if $d_F(s,q) \leq \rho$.*

Before we prove Lemma 9, we introduce the notion of a valid sequence of cells in the free-space diagram and the set of predicates that are induced by such a sequence. In the following, we denote with $C_{i,j}$ the cell of the free-space diagram that corresponds to the edges $\overline{q_i q_{i+1}}$ and $\overline{s_j s_{j+1}}$. We call a sequence of cells $\mathscr{C} = ((i_1, j_1), (i_2, j_2), \ldots, (i_k, j_k))$ *valid* if $i_1 = 1, j_1 = 1, i_k = t_q - 1, j_k = t_s - 1$ and if for any two consecutive cells (i_m, j_m) and (i_{m+1}, j_{m+1}) it holds that either $i_m = i_{m+1}$ and

$j_{m+1} = j_m + 1$ or $j_m = j_{m+1}$ and $i_{m+1} = i_m + 1$. Note that a sequence is valid if there exists a feasible path which passes through the sequence of cells in the right order. At the same time, there exists a valid sequence of cells for any such path. Any valid sequence of cells $\mathscr{C}$ induces a set of predicates $\mathscr{P}$ as follows.

(i) $(P_1) \in \mathscr{P}$ and $(P_2) \in \mathscr{P}$

(ii) $(P_3)_{(i,j)} \in \mathscr{P}$ iff $(i, j-1), (i, j) \in \mathscr{C}$

(iii) $(P_4)_{(i,j)} \in \mathscr{P}$ iff $(i-1, j), (i, j) \in \mathscr{C}$

(iv) $(P_5)_{(i,j,k)} \in \mathscr{P}$ iff $(i, j-1), (i, k) \in \mathscr{C}$ and $j < k$

(v) $(P_6)_{(i,j,k)} \in \mathscr{P}$ iff $(i-1, j), (k, j) \in \mathscr{C}$ and $i < k$

We say that a valid sequence of cells is *feasible* if the conjunction of its induced predicates is true. We claim that any feasible path through the free-space induces a feasible sequence of cells and vice versa. Before we prove this claim, we prove the following helper lemma. Note the subtle difference to the definition of the monotonicity predicate.

Lemma 10. *Let $\mathscr{C}$ be a feasible sequence of cells and consider a monotonicity predicate P of the set of predicates $\mathscr{P}$ induced by $\mathscr{C}$. Let a_1 and a_2 be the vertices and let e be the directed edge associated with P. There exist two points p_1 and p_2 on e, such that p_1 appears before p_2 on e, and such that $\|p_1 - a_1\| \leq \rho$ and $\|p_2 - a_2\| \leq \rho$.*

Proof. Assume P is a horizontal monotonicity predicate $(P_5)_{(i,j,k)}$ in $\mathscr{P}$ (the arguments for vertical monotonicity predicates are similar). The predicate P was added because of cells $C_{(i,j-1)}$ and $C_{(i,k)}$ being present in $\mathscr{C}$. Since $\mathscr{C}$ is valid, it must be that $C_{(i,j)}$ and $C_{(i,k-1)}$ are also present (possibly with $j = k - 1$). Therefore, $\mathscr{P}$ also contains horizontal vertex-edge predicates $(P_3)_{(i,j)}$ and $(P_3)_{(i,k)}$. If all three predicates are true, then we want to follow that there exist points p_1 and p_2 on the *edge* (not just the supporting line) $\overline{q_i q_{i+1}}$ such that $\|p_1 - s_j\| \leq \rho$ and $\|p_2 - s_k\| \leq \rho$.

We consider two cases, based on whether the common intersection of the line ℓ supporting the edge $\overline{q_i q_{i+1}}$ and the two disks of radius ρ centered at s_j and s_k is empty, or in other words if $\text{Disk}(s_j, \rho) \cap \text{Disk}(s_k, \rho) \cap \overleftrightarrow{q_i q_{i+1}} = \emptyset$. If the intersection is empty, then the line intersects the disks in two disjoint intervals. In this case any pair of points $p_1 \in \overline{q_i q_{i+1}} \cap \text{Disk}(s_j, r)$ and $p_2 \in \overline{q_i q_{i+1}} \cap \text{Disk}(s_k, r)$ appears on ℓ in the correct order. Two such points p_1 and p_2 must exist since both vertex-edge predicates are true.

If the common intersection is not empty, then we argue that the edge $\overline{q_i q_{i+1}}$ must intersect this common intersection. Indeed, since both vertex-edge predicates are true, the edge intersects both disks. Since the edge is a connected set, it must also intersect the common intersection which lies in between the intersections of the line with the two disks. Now, if the edge intersects the common intersection, then we can choose $p_1 = p_2$ from this common intersection. We can make a similar argument for each vertical monotonicity predicate.

This implies that the points p_1 and p_2 of the monotonicity predicates are realizable on the corresponding edges as claimed. □

Proof of Lemma 9. We claim that any feasible path through the free-space induces a feasible sequence of cells and vice versa. Assume there exists a feasible path π that passes thorough the sequence of cells $\mathscr{C}$. Consider the endpoint predicate (P_1) (and respectively (P_2)). The existence of

π implies that $(0,0)$ (and respectively $(1,1)$) lies inside the free-space, which is equivalent to this predicate being true. Now, consider a horizontal vertex-edge predicate $(P_3)_{(i,j)}$ for consecutive pair of cells $C_{(i,j-1)}$, $C_{(i,j)}$ in the sequence $\mathscr{C}$. The path π is a feasible path that passes through the cell boundary between these two cells. This implies that the there exists a point on the edge $\overline{q_i q_{i+1}}$ which lies within distance ρ to the vertex s_j. This implies that the predicate is true. A similar argument can be made for each vertical vertex-edge predicate.

Next, we will discuss the monotonicity predicates. Consider a subsequence of cells of $\mathscr{C}$ that lies in a fixed row i and consider the set of predicates $\mathscr{P}' \subseteq \mathscr{P}$ that consists of horizontal monotonicity predicates $(P_5)_{(i,j,k)}$ for fixed i. Let $p_j, p_{j+1}, \ldots, p_k$ be the sequence of points along q that correspond to the vertical coordinates where the path π passes through the corresponding cell boundaries corresponding to vertices $s_j, s_{j+1} \ldots, s_k$. The sequence of points lies on the directed line supporting the edge $\overline{q_i q_{i+1}}$ and the points appear in their order along this line in the sequence due to the monotonicity of π. Since π is a feasible path it lies in the free-space and therefore we have $\|p_{k'} - s_{k'}\|$ for every $j \leq k' \leq k$. This implies that all predicates in $\mathscr{P}'$ are true. We can make a similar argument for the vertical monotonicity predicates $(P_6)_{(i,j,k)}$ for a fixed column j. This shows that a feasible path π that passes through the cells of $\mathscr{C}$ implies that the conjunction of induced predicates $\mathscr{P}$ is true.

It remains to show the other direction: Any feasible sequences of cells implies the existence of a feasible path. It is clear that the relationship between a feasible path π and the endpoint predicates as well as the vertex-edge predicates, as described above, gives us the existence of a continuous (not necessarily monotone) path π that stays inside the free-space and connects $(0,0)$ with $(1,1)$. We now have to argue that the monotonicity predicates imply that there always exists such a path that is also (x,y)-monotone.

Assume for the sake of contradiction that the conjunction of predicates in $\mathscr{P}$ is true, but there exists no feasible path through the sequence of cells $\mathscr{C}$. In this case, it must be that either a horizontal passage or a vertical passage is not possible. Concretely, in the first case, there must be two vertices s_j and s_k and a directed edge $e = \overline{q_i q_{i+1}}$, such that there exist no two points p_1 and p_2 on e, such that p_1 appears before p_2 on e, and such that $\|p_1 - s_j\| \leq \rho$ and $\|p_2 - s_k\| \leq \rho$. However, $(P_5)_{i,j,k}$ is contained in $\mathscr{P}$ and by Lemma 10 two such points p_1 and p_2 must exist. We obtain a contradiction. In the second case, the argument is similar. Therefore, a feasible sequences of cells implies a feasible path, as claimed. □

Lemma 11. *Given a truth assignment to all predicates of two curves q and s, we can decide if there exists a feasible sequence of cells in $O(t_s t_q(t_s + t_q))$ time without knowing q or s.*

Proof. Let $C_{i,j}$ denote the cell in the free space diagram that corresponds to the ith edge on q (the ith row) and the jth edge on s (the jth column). For the sake of this proof, we re-define the notion of a valid sequence of cells. We keep the definition as before, except that we drop the requirement that $i_k = t_q - 1, j_k = t_s - 1$. That is, a valid path may end at any cell. Such a path is feasible if the conjunction of its induced predicates is true, as before. We want to process these cells in the lexicographical ordering of their indices (i,j) and determine for each cell whether it is reachable by such a feasible sequence of cells. Concretely, this happens for a cell $C_{i,j}$ if and only if there exists a feasible sequence of cells that ends with (i,j). Furthermore, we are interested in the second-to-last step. If there exists a feasible sequence that ends with the two pairs $(i-1,j),(i,j)$ we say that $C_{i,j}$ is *reachable from below*, and similarly if there exists a feasible sequence that ends with the two pairs $(i,j-1),(i,j)$ we say that $C_{i,j}$ is *reachable from the left*. In addition, for each processed cell $C_{i,j}$,

we maintain two indices:

(i) if $C_{i,j}$ is reachable from the left, we maintain the maximal column index $j' \leq j$ such that $C_{i,j'}$ is reachable from below.

(ii) if $C_{i,j}$ is reachable from the below, we maintain the maximal row index $i' \leq j$ such that $C_{i',j}$ is reachable from the left.

We call these indices the *previous right turn* and the *previous left turn*. Intuitively, the indices describe the index of the cell where the feasible path that reached the cell from below previously turned left, and respectively where the feasible path that reached the cell from the left previously turned right.

We now describe the algorithm. If $(P_1) \wedge (P_2)$ evaluates to false, the algorithm returns false. Otherwise, we mark the cell $C_{1,1}$ as reachable from below and reachable from the left. Processing a cell $C_{i,j}$ is done by executing the following steps:

- If $C_{i-1,j}$ is reachable from the left and if $(P_3)_{i,j}$ evaluates to true, then we mark $C_{i,j}$ as reachable from below and we set its previous left turn to $i-1$.
- If $C_{i-1,j}$ is reachable from below, let i' denote its previous left turn. If $(P_3)_{i,j}$ evaluates to true and if $(P_6)_{i,j,i''}$ evaluates to true for $i' \leq i'' \leq i-1$, then we mark $C_{i,j}$ as reachable from below and we set its previous left turn to i'.
- If $C_{i,j-1}$ is reachable from below and if $(P_4)_{i,j}$ evaluates to true, then we mark $C_{i,j}$ as reachable from the left and we set its previous left turn to $j-1$.
- If $C_{i,j-1}$ is reachable from below, let j' denote its previous right turn. If $(P_4)_{i,j}$ evaluates to true and if $(P_5)_{i,j,j''}$ evaluates to true for $j' \leq j'' \leq j-1$, then we mark $C_{i,j}$ as reachable from the left and we set its previous right turn to j'.
- Finally, if in the above steps we marked $C_{i,j}$ both as reachable from below and reachable from the left, we set the previous right turn to j and the previous left turn to i.

Clearly, processing each cell takes time at most $O(t_q + t_s)$. In total we are processing $O(t_q t_s)$ cells. Therefore the total running time is $O(t_q t_s(t_q + t_s))$. The correctness of the algorithm can be proven by induction on the cells in their processing order. □

7.3.2 Low-level Predicates

In the following, we describe a set of simpler predicates that help us build a data structure. Each group of low-level predicates will be used to represent one high-level predicate. Let a_1 be the vertex and let $\overline{b_1 b_2}$ be the edge of a vertex-edge predicate (P_3) (respectively, (P_4)). We define the following three predicates.

(a) $\|a_1 - b_1\| \leq \rho$.

(b) $\|a_1 - b_2\| \leq \rho$.

(c) a_1 is contained in the rotated rectangle R with side lengths $2r$ by $\|b_1 - b_2\|$ which is contained in the Minkowski sum of the edge with the disk of radius ρ. (Refer to Figure 8 (left))

Lemma 12. *Given the truth values of the predicates (a)-(c) one can determine the truth value of the predicate (P_3) (respectively, (P_4)).*[5]

Proof. Consider a vertex-edge predicate with corresponding vertex a_1 and edge $\overline{b_1 b_2}$ and assume we know the true values of predicates (a),(b) and (c). We can determine the truth value of the

[5]This property was also used in the data structure by de Berg *et al.* [10]

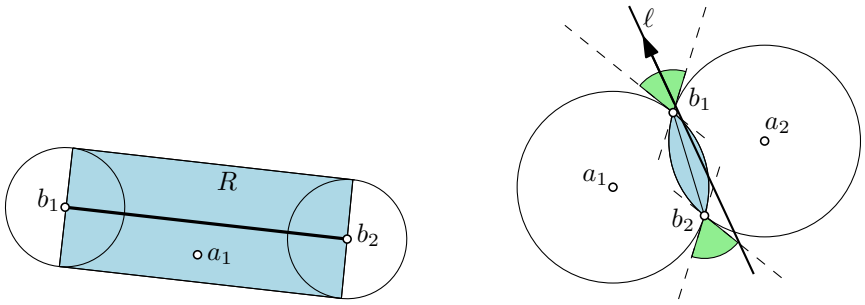


Figure 8: Geometric objects involved in the low-level predicates.

high-level certificate as $a \vee b \vee c$. Indeed, the union of the two disks and the rectangle is equal to the Minkowski sum of the edge with a disk of radius ρ. This is exactly the locus of values for a_1 for which the predicate should return true. □

We also want to break down the monotonicity predicates into a constant number of simpler predicates. Let a_1, a_2 be the vertices and let ℓ be the line supporting the directed edge e of a monotonicity predicate (P_5) (respectively, (P_6)). We begin with the following simple predicates
(d) The line ℓ intersects the circle of radius ρ centered at a_1.
(e) The line ℓ intersects the circle of radius ρ centered at a_2.
(f) The angle between the translation vector $(a_2 - a_1)$ and the edge e is at most $\frac{\pi}{2}$.

In addition, we will distinguish the case that the two circles of radius ρ centered at the two vertices intersect each other. This is captured by the following predicate.
(g) $\|a_1 - a_2\| \leq 2\rho$

Lemma 13. *If $d \wedge e \wedge f$ evaluates to true, then the corresponding monotonicity predicate evaluates to true. Furthermore, if g evaluates to false or if the line does not intersect the lens formed by the two disks at a_1 and a_2, then $d \wedge e \wedge f$ is equivalent to the corresponding monotonicity predicate.*

Proof. Consider a parametrized representation of the line given by a point $p_\ell \in \mathbb{R}^2$ and a direction vector $v_\ell \in \mathbb{R}^2$ with $\|v_\ell\| = 1$

$$\ell : \{p_\ell + tv_\ell \mid t \in \mathbb{R}\}.$$

Let $a_1' = p_\ell + \langle a_1 - p_\ell, v_\ell \rangle v_\ell$ and let $a_2' = p_\ell + \langle a_2 - p_\ell, v_\ell \rangle v_\ell$, where $\langle \cdot, \cdot \rangle$ denotes the inner product. Note that the point a_1' (respectively a_2') is an orthogonal projection onto the line ℓ and minimizes the distance to a_1 (respectively, a_2). Therefore (d) implies $\|a_1 - a_1'\| \leq \rho$ and (e) implies $\|a_2 - a_2'\| \leq \rho$. Furthermore, (f) implies that a_1' appears before a_2' on the line. This implies that the corresponding monotonicity predicate evaluates to true. Now, if (g) evaluates to false, then the line ℓ intersects the two disks in disjoint intervals. In this case, the order along ℓ of any two points from these two intervals (p_1 in the intersection interval with the disk at a_1 and p_2 in the intersection interval with the disk at a_2) indicates the correct truth value of the monotonicity predicate. Therefore also a_1' and a_2' do. The same argument holds for the case that (g) evaluates to true and the line does not intersect the lens formed by the two disks. □

In case (g) evaluates to true, let b_1 and b_2 be the two intersection points of the two circles. For this case we introduce the following additional predicates.
(h) The line ℓ passes in between the two points b_1 and b_2

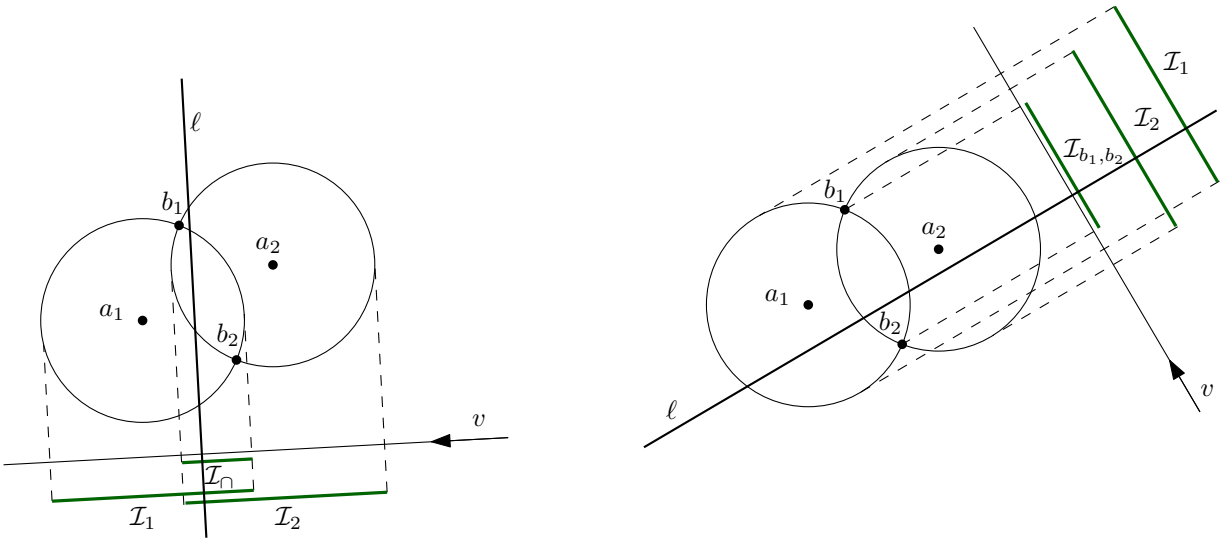


Figure 9: Examples of the two cases analyzed in the proof of Lemma 14.

(i) The angle of ℓ is contained in the range of angles of tangents of the circular arc between b_1 and b_2 of the circle of radius ρ centered at a_1. (Refer to Figure 8 (right))

The purpose of the next lemma is to describe the cases in which the line intersects the lens that is formed by the two disks at a_1 and a_2. If (g) evaluates to false, the two points b_1 and b_2 are undefined and we define the below predicates to be false.

Lemma 14. *If and only if $h \vee (d \wedge e \wedge i)$ evaluates to true, then the line ℓ intersects the lens formed by the two disks of radius ρ at a_1 and a_2.*

Proof. Clearly, if (h) evaluates to true, then the line intersects the lens, since the line segment between b_1 and b_2 is contained in the lens. For the remainder of the proof we intend to do a case analysis based on the angle of ℓ with respect to the truth value of (i). Refer to Figure 9.

In the following we denote with $\pi_v(U)$ the projection of a set U onto the subspace spanned by v:

$$\pi_v(U) = \{\langle v, p\rangle \mid p \in U\}.$$

Let v be the normal to the line ℓ and consider the projections of the two disks onto this subspace $\mathcal{I}_1 = \pi_v(\text{Disk}(a_1, r))$ and $\mathcal{I}_2 = \pi_v(\text{Disk}(a_2, r))$ as well as the projection of the lens $\mathcal{I}_\cap = \pi_v(\text{Disk}(a_1, r) \cap \text{Disk}(a_2, r))$

Note that, by symmetry of the lens, the range of angles of tangents is the same on both arcs of the lens. Therefore, if (i) evaluates to true then the angle of ℓ lies in this range. In this case, we have that $\mathcal{I}_\cap = \mathcal{I}_1 \cap \mathcal{I}_2$, since the two bounding tangents to the lens which are parallel to ℓ are also tangents—one to each of the two disks. Therefore, the projection of any line ℓ' parallel to ℓ lies in the interval $\mathcal{I}_1 \cap \mathcal{I}_2$ if and only if it intersects the lens. This condition is equivalent to $(d \wedge e)$.

Now assume that (i) evaluates to false. Consider the projection of the line segment bounded by b_1 and b_2 onto the same subspace as before $\mathcal{I}_{b_1,b_2} = \pi_v\big(\overline{b_1 b_2}\big)$ For this range of angles (when (i) evaluates to false), we have that $\mathcal{I}_\cap = \mathcal{I}_{b_1,b_2}$. Therefore, in this case, the line intersects the lens if and only if it intersects the line segment bounded by b_1 and b_2. This condition is equivalent to (h). □

Lemma 15. *Assume (g) evaluates to true. If the line ℓ intersects the lens formed by the two disks at a_1 and a_2, then the corresponding monotonicity predicate evaluates to true.*

Proof. Let p denote some point in the intersection of the line with the lens. We have that $\|p-a_1\| \leq \rho$ and $\|p - a_2\| \leq \rho$, since the intersection point lies inside the intersection of the two disks. We can set $p_1 = p$ and $p_2 = p$ to satisfy the condition for the monotonicity predicate to be true. □

Lemma 16. *Given the truth values of the predicates (d)-(i) one can determine the truth value of the predicate (P_5) (respectively, (P_6)).*

Proof. We can determine the truth value of the corresponding monotonicity predicate as follows

$$(d \wedge e \wedge f) \vee (h \vee (d \wedge e \wedge i)) \tag{6}$$

Indeed, together, Lemma 13, Lemma 14 and Lemma 15 testify that Equation (6) implies the high-level predicate. In the other direction, we argue based on the case whether the line intersects the lens. If the line does not intersect the lens, or the lens does not exist (i.e., (g) evaluates to false), then by Lemma 13, the predicate implies Equation (6). If the line does intersect the lens, then the predicate must be true and by Lemma 14, Equation (6) must be true as well. □

7.3.3 The Data Structure

We start by describing a set of binary matrices that will guide the layout of the data structure and the query algorithm. The matrices play a role similar to the free-space matrix in Section 7.2. The entries of these matrices together define a truth assignment to the overall set of predicates. The query algorithm will process the matrices column by column. We describe four matrices, one for each group of high-level predicates. The predicates (P_1) and (P_2) are tested separately.

1. *Horizontal vertex-edge predicates*
   The matrix $\mathscr{M}_{(P_3)}$ consists of t_q rows and $6 \cdot t_s$ columns. We group the columns in t_s groups of 6 columns each. The entries of the jth group in the ith row are associated with the high-level predicate $(P_3)_{(i,j)}$. For each i and j, the predicate is a function of the vertex q_i of the query curve and the edge $\overline{s_j s_{j+1}}$. Each entry stores the truth value of a low-level predicate associated with this high-level predicate. We define these low-level predicates as follows. Consider the rotated rectangle R around $\overline{s_j s_{j+1}}$ defined in predicate (c). Denote with ℓ_1 and ℓ_2 the two lines that bound R from above and denote with ℓ_3 and ℓ_4 the two lines that bound R from below. A point p is included in the R if and only if p lies below ℓ_1 and ℓ_2 and p lies above ℓ_3 and ℓ_4. The 6 entries of the jth group of columns in row i are defined as follows: (1) $s_j \in \text{Disk}(q_i, r)$ (2) $s_{j+1} \in \text{Disk}(q_i, r)$ (3) dual(ℓ_1) lies below dual(q_i) (4) dual(ℓ_2) lies below dual(q_i) (5) dual(ℓ_3) lies above dual(q_i) (6) dual(ℓ_4) lies above dual(q_i).

2. *Vertical vertex-edge predicates*
   Similar to the matrix $\mathscr{M}_{(P_3)}$, the matrix $\mathscr{M}_{(P_4)}$ consists of t_q rows and $6 \cdot t_s$ columns. We group the columns in t_s groups of 6 columns each. The entries of the jth group in the ith row are associated with the high-level predicate $(P_4)_{(i,j)}$. For each i and j, the predicate $(P_4)_{(i,j)}$ is a function of the vertex s_j and the edge $\overline{q_i q_{i+1}}$ of the query curve. Consider the rotated rectangle R around $\overline{q_i q_{i+1}}$ defined in predicate (c). Denote with ℓ_1 and ℓ_2 be the two lines that bound R from above and denote with ℓ_3 and ℓ_4 be the two lines that bound R from

below. The 6 entries are defined as follows: (1) $s_j \in \text{Disk}(q_i, r)$ (2) $s_j \in \text{Disk}(q_{i+1}, r)$ (3) s_j lies below ℓ_1 (4) s_j lies below ℓ_2 (5) s_j lies above ℓ_3 (6) s_j lies above ℓ_4.

3. *Horizontal monotonicity predicates*
The matrix $\mathscr{M}_{(P_5)}$ consists of t_q rows and $\frac{9}{2}t_s(t_s-1)$ columns. We group the columns in $\frac{t_s(t_s-1)}{2}$ groups of 9 columns each. Each group is associated with a fixed value of j and k, with $j < k$ and $j, k \in [t_s]$. The entries of a specific group in the ith row are associated with the high-level predicate $(P_5)_{(i,j,k)}$. This predicate is a function of the two vertices s_j and s_k and the edge $\overline{q_i q_{i+1}}$ of the query curve. Let $\alpha_q \in [0, 2\pi]$ be the angle of the translation vector $(q_{i+1} - q_i)$ with the x-axis and let α_s be the angle of the translation vector $(s_k - s_j)$ with the x-axis. We denote with ℓ the line that supports $\overline{q_i q_{i+1}}$. Let ℓ_+ and ℓ_- be two lines parallel to ℓ which lie at distance ρ to ℓ and such that ℓ_+ lies above ℓ_-. If the circles of radius ρ centered at s_j and s_k intersect, then let b_+ and b_- denote their intersection points, such that b_+ has the larger y-coordinate of the two points. Consider the range of angles of tangents to the intersection of the two disks at s_j and s_k. This range consists of two disjoint intervals of the circular range of angles $[0, 2\pi]$. At least for one of the two intervals the left endpoint is smaller than the right endpoint (i.e., the interval does not contain 2π). Let $[\alpha_-, \alpha_+] \subseteq [0, 2\pi]$ be this interval. Let $\alpha_\ell \in [0, 2\pi]$ be one of the two angles of the undirected line ℓ (we need to query with both of them). The 9 entries are defined as follows: (1) s_j lies below ℓ_+ (2) s_j lies above ℓ_- (3) s_k lies below ℓ_+ (4) s_k lies above ℓ_- (5) $\alpha_s \in [\alpha_q - \frac{\pi}{2}, \alpha_q + \frac{\pi}{2}]$ (6) b_+ lies above ℓ (7) b_- lies below ℓ (8) $\alpha_- \leq \alpha_\ell$ (9) $\alpha_+ \geq \alpha_\ell$.

4. *Vertical monotonicity predicates*
The matrix $\mathscr{M}_{(P_6)}$ consists of $\frac{t_q(t_q-1)}{2}$ rows and $8 \cdot t_s$ columns. We group the columns in t_s groups of 8 columns each. Each row is associated with a fixed value of i and k, with $i < k$ and $i, k \in [t_s]$. The entries of the jth group of columns in a specific row associated with some value of j and k are associated with the high-level predicate $(P_5)_{(i,j,k)}$. This predicate is a function of the two vertices q_i and q_k and the edge $\overline{s_j s_{j+1}}$. Let $\alpha_q \in [0, 2\pi]$ be the angle of the translation vector $(q_k - q_i)$ with the x-axis and let α_s be the angle of the translation vector $(s_{j+1} - s_j)$ with the x-axis. We denote with ℓ the line that supports $\overline{s_j s_{j+1}}$. Let ℓ_+ and ℓ_- be two lines parallel to ℓ which lie at distance ρ to ℓ and such that ℓ_+ lies above ℓ_-. If the circles of radius ρ centered at q_i and q_k intersect, then let b_+ and b_- denote their intersection points, such that b_+ has the larger y-coordinate. Let $\mathcal{I}_q$ be the (not necessarily connected) range of angles of tangents to the intersection of the two disks at q_i and q_k. The 8 entries are defined as follows: (1) dual(ℓ_+) lies above dual(q_i) (2) dual(ℓ_-) lies below dual(q_i) (3) dual(ℓ_+) lies above dual(q_k) (4) dual(ℓ_-) lies below dual(q_k) (5) $\alpha_s \in [\alpha_q - \frac{\pi}{2}, \alpha_q + \frac{\pi}{2}]$ (6) dual(ℓ) lies below dual(b_+) (7) dual(ℓ) lies above dual(b_-) (8) $\alpha_s \in \mathcal{I}_q$.

Following the lemmas in Section 7.3 we can make the following observation.

Observation 9. *Given an instance of each matrix $\mathscr{M}_{(P_3)}, \mathscr{M}_{(P_4)}, \mathscr{M}_{(P_5)}$, and $\mathscr{M}_{(P_6)}$ and given the values of the two predicates (P_1) and (P_2), one can determine the values of the high-level predicates using Lemma 13, Lemma 14 and Lemma 15.*

Observation 10. *The total number of columns of the matrices $\mathscr{M}_{(P_3)}, \mathscr{M}_{(P_4)}, \mathscr{M}_{(P_5)}$, and $\mathscr{M}_{(P_6)}$ is bounded by $O(t_s^2)$.*

We are now ready to state our problem in the terms of a multilevel semialgebraic range searching problem. The input for the multilevel semialgebraic range searching problem is a set $\mathscr{P}$ of n t-points in $\mathbb{R}^2$ which we define as follows. Each of the columns of the matrices $\mathscr{M}_{(P_3)}, \mathscr{M}_{(P_4)}, \mathscr{M}_{(P_5)}$, and $\mathscr{M}_{(P_6)}$ defines a point $p \in \mathbb{R}^2$ (or a value $p \in \mathbb{R}$, in this case we set the second coordinate to 0) derived from each input curve. Let c be the total number of columns of the four matrices[6]. We define a t-point $(\mathbf{p}^{(1)}, \mathbf{p}^{(2)}, \ldots, \mathbf{p}^{(t)})$ for each input curve s with $t = c + 2$ such that $\mathbf{p}^{(k)}$ for $k \leq c$ is the point of s that is defined by the predicate in the kth column, where $k \in [1, c]$ uniquely identifies a column across the four matrices. For $k = c + 1$ we define $\mathbf{p}^{(k)} = s_1$ and for $k = c + 2$ we defined $\mathbf{p}^{(k)} = s_{t_s}$. Note that these two point sets are associated with the predicates (P_1) and (P_2), which are not captured by the matrices. We define the set of t-points obtained this way to be the set $\mathscr{P}$.

Before we describe the queries, we note that using the data structure described in Section 7.1 we use space in $S(n) = O(nO(\log\log n)^{t-1})$, where t is in $O(t_s^2)$ by Observation 10.

In the following, we describe the queries. As before, each query is a tuple of t semialgebraic sets $(\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_t)$, where each semialgebraic set is defined by a constant number of polynomial inequalities of constant degree.

Each entry of the matrices $\mathscr{M}_{(P_3)}, \mathscr{M}_{(P_4)}, \mathscr{M}_{(P_5)}$, and $\mathscr{M}_{(P_6)}$ defines a query range derived from the query curve that is either a disk or a halfspace. In the first phase of the query algorithm we compute the arrangement of these ranges for each column. Every cell of the partition corresponds to a truth assignment to this column. In this way we generate all possible truth assignments that correspond to non-empty query ranges. Each truth assignment to a column can be stored as a bit vector. The Cartesian product of the set of bit vectors generated this way yields a set of truth assignments to the matrices $\mathscr{M}_{(P_3)}, \mathscr{M}_{(P_4)}, \mathscr{M}_{(P_5)}$, and $\mathscr{M}_{(P_6)}$. From this set we want to use those truth assignments only that have a feasible sequence of cells. We can test this for each generated matrix using Observation 9 and Lemma 11 in $O(t_s t_q(t_s + t_q))$ time.

In the second phase of the query algorithm we have a fixed truth assignment and for each column we compute the cell of the arrangement corresponding to this truth assignment. We now refine the cells as described in Section 7.2, by lifting circular ranges to $\mathbb{R}^3$ and mapping back the edges of the triangulation to $\mathbb{R}^2$ in order to obtain cells that can be described by a constant number of polynomial inequalities of constant degree. This generates a set of ranges for each column. Finally, we take the Cartesian product of these ranges as done in Section 7.2.

Putting all of the above together, we can bound the total query time with

$$O(\sqrt{n}\log^{O(t_s^2)} n \cdot t_q^{O(t_s^2)}) = O(\sqrt{n}\log^{O(t_s^2)} n)$$

assuming $t_q = O(\log^{O(1)} n)$.

Theorem 9. *Given a set S of n polygonal curves in $\mathbb{R}^d$ where each curve contains t_s vertices, we can store S in a data structure of $\mathcal{O}\Big(n(\log\log n)^{O(t_s^2)}\Big)$ size such that given a query polygonal chain of size t_q and a parameter ρ, it can output all the input curves within continuous Fréchet distance of ρ to the query in $\mathcal{O}\Big(\sqrt{n}\cdot\log^{O(t_s^2)} n\Big)$, assuming $t_q = \log^{O(1)} n$.*

[6] Indeed, the ordering of the columns does not matter as long as it is consistent throughout the data structure and query algorithm.

8 Conclusions

We studied the space/query-time tradeoff of multi-level data structures for range searching under the Fréchet distance. The aim of our study was two-fold. On the one hand, we wanted to answer a fundamental question related to range searching: Do multilevel data structures need to have an exponential dependency on the number of levels? We answer the question to the negative by proving a lower bound on the space/query time tradeoff for a concrete problem which we refer to as multilevel stabbing problem. In particular, our lower bound shows that finding a general technique for removing the exponential dependency on the number of levels is not feasible. On the other hand, we were interested in the complexity of range searching among polygonal curves under the Fréchet distance. Previous to our work, the complexity of this problem was largely open. We give upper and lower bounds on the space/query-time tradeoff for both the discrete and continuous versions of the Fréchet distance. The fact that we can extend our lower bound to such a practically relevant problem further supports our claimed negative answer to the broader range searching question mentioned above. Our data structures invoke semialgebraic range searching within the framework of multilevel partition trees. Here, the major challenge lies in the Fréchet distance not being defined as a closed form algebraic expression. In other words, previous to our work, it was not at all obvious how semialgebraic range searching could be applied to range searching under the Fréchet distance. Our upper bounds for this problem are in line with the lower bounds, as the number of levels is in the order of t, the complexity of the polygonal curves. For the continuous Fréchet distance, the number of levels increases to $O(t^2)$. We think that it can be reduced to $O(t)$ by selectively using dualization in some of the levels. However, doing so requires dealing with more technical details and given the technical nature of the current results, we have decided to pursue this improvement as a part of future work.

Finally, we can identify a number of interesting open problems that remain. We close our discussion by stating some of them in no particular order.

1. Can the use of semialgebraic range searching be circumvented for Fréchet queries? A positive answer could help us solve the following open question.
2. Can we develop data structures for (discrete or continuous) Fréchet queries that match the lower bounds in the high-space/low-query-time regime?
3. How fast can we do approximate range searching under the (discrete or continuous) Fréchet distance? Using the standard $(1+\varepsilon)$-approximation of spherical ranges, we can use simplex-range searching to get the full spectrum of the space/query time tradeoff, but even more efficient data structures might be possible.
4. Can we prove lower bounds for answering approximate Fréchet distance queries? One particular challenge here is that the known lower bound frameworks cannot handle approximations. For example, this might mean that we need to extend the pointer machine lower bound framework of Afshani [1].
5. Can we do range searching under the continuous Fréchet distance among polygonal curves in three or higher dimensions? In particular, we are interested in a data structure that uses $O(n \log^{t^{O(1)}} n)$ space and answers queries in $O(n^{1-1/d} \log^{t^{O(1)}} n)$ time where t is the maximum complexity of a query or input curve. We suspect the answer might be negative.
6. Can we do range searching under related distance measures such as dynamic time warping?
7. Do our lower bounds extend to multilevel stabbing queries in the semigroup model?
8. Can we prove stronger lower bounds for polygonal curves in three or higher dimensions? We

hope that our upper bounds for the discrete Fréchet distance could be matched.

For resolving the last two questions, the main challenge lies in proving the equivalent of Theorem 2 on page 13 and a case analysis such as the one in Section 6.2.4.

References

[1] P. Afshani. Improved pointer machine and I/O lower bounds for simplex range reporting and related problems. In *Symposium on Computational Geometry (SoCG)*, pages 339–346, 2012.

[2] P. Afshani, L. Arge, and K. D. Larsen. Orthogonal range reporting in three and higher dimensions. In *Proceedings of Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 149–158, 2009.

[3] P. Afshani, L. Arge, and K. D. Larsen. Orthogonal range reporting: query lower bounds, optimal structures in 3-d, and higher-dimensional improvements. In *Symposium on Computational Geometry (SoCG)*, pages 240–246, 2010.

[4] P. K. Agarwal and J. Erickson. Geometric range searching and its relatives. In B. Chazelle, J. E. Goodman, and R. Pollack, editors, *Advances in Discrete and Computational Geometry*. AMS Press, 1999.

[5] M. Ali, J. Krumm, T. Rautman, and A. Teredesai. ACM SIGSPATIAL GIS Cup 2012. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '12, pages 597–600, New York, NY, USA, 2012. ACM.

[6] K. Bringmann. Why walking the dog takes time: Fréchet distance has no strongly subquadratic algorithms unless SETH fails. In *Proceedings of Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 661–670, 2014.

[7] T. M. Chan. Optimal partition trees. In *Symposium on Computational Geometry (SoCG)*, pages 1–10. ACM, 2010.

[8] B. Chazelle. Lower bounds on the complexity of polytope range searching. *Journal of the American Mathematical Society*, 2:637–666, 1989.

[9] B. Chazelle. Lower bounds for orthogonal range searching: I. the reporting case. *Journal of the ACM (JACM)*, 37(2):200–212, 1990.

[10] M. De Berg, A. F. Cook, and J. Gudmundsson. Fast Fréchet queries. *Computational Geometry*, 46(6):747–755, 2013.

[11] A. Driemel and F. Silvestri. Locality-sensitive hashing of curves. In *Symposium on Computational Geometry (SoCG)*, pages 37:1–37:16, 2017.

[12] J. Gudmundsson and M. Horton. Spatio-temporal analysis of team sports. *ACM Comput. Surv.*, 50(2):22:1–22:34, Apr. 2017.

[13] J. Gudmundsson and M. Smid. Fast algorithms for approximate Fréchet matching queries in geometric trees. *Computational Geometry*, 48(6):479 – 494, 2015.

[14] L. J. Guibas, J. E. Hershberger, J. S. Mitchell, and J. S. Snoeyink. Approximating polygons and subdivisions with minimum-link paths. *International Journal of Computational Geometry & Applications*, 3(04):383–415, 1993.

[15] P. Indyk. Approximate nearest neighbor algorithms for Fréchet distance via product metrics. In *Symposium on Computational Geometry*, pages 102–106, 2002.

[16] J. K. Laurila, D. Gatica-Perez, I. Aad, B. J., O. Bornet, T.-M.-T. Do, O. Dousse, J. Eberle, and M. Miettinen. The mobile data challenge: Big data for mobile computing research. In *Pervasive Computing*, 2012.

[17] J. Matoušek. Range searching with efficient hierarchical cuttings. *Discrete & Computational Geometry*, 10(2):157–182, 1993.

[18] F. Zheng and T. Kaiser. *Digital Signal Processing for RFID*. Wiley, 2016.