# Approximate Nearest Neighbor for Polygonal Curves Under Fréchet Distance

**Siu-Wing Cheng**
Department of Computer Science and Engineering,
Hong Kong University of Science and Technology, Hong Kong, China

**Haoqiang Huang**
Department of Computer Science and Engineering,
Hong Kong University of Science and Technology, Hong Kong, China

## Abstract

We propose $\kappa$-approximate nearest neighbor (ANN) data structures for $n$ polygonal curves under the Fréchet distance in $\mathbb{R}^d$, where $\kappa \in \{1+\varepsilon, 3+\varepsilon\}$ and $d \geq 2$. We assume that every input curve has at most $m$ vertices, every query curve has at most $k$ vertices, $k \ll m$, and $k$ is given for preprocessing. The query times are $\tilde{O}(k(mn)^{0.5+\varepsilon}/\varepsilon^d + k(d/\varepsilon)^{O(dk)})$ for $(1+\varepsilon)$-ANN and $\tilde{O}(k(mn)^{0.5+\varepsilon}/\varepsilon^d)$ for $(3+\varepsilon)$-ANN. The space and expected preprocessing time are $\tilde{O}(k(mnd^d/\varepsilon^d)^{O(k+1/\varepsilon^2)})$ in both cases. In two and three dimensions, we improve the query times to $O(1/\varepsilon)^{O(k)} \cdot \tilde{O}(k)$ for $(1+\varepsilon)$-ANN and $\tilde{O}(k)$ for $(3+\varepsilon)$-ANN. The space and expected preprocessing time improve to $O(mn/\varepsilon)^{O(k)} \cdot \tilde{O}(k)$ in both cases. For ease of presentation, we treat factors in our bounds that depend purely on $d$ as $O(1)$. The hidden polylog factors in the big-$\tilde{O}$ notation have powers dependent on $d$.

**2012 ACM Subject Classification** Theory of computation → Computational geometry

**Keywords and phrases** Polygonal curves, Fréchet distance, approximate nearest neighbor

**Digital Object Identifier** 10.4230/LIPIcs.ICALP.2023.40

**Category** Track A: Algorithms, Complexity and Games

**Related Version** *Full Version*: `https://arxiv.org/abs/2304.14643`

## 1 Introduction

Given a set of trajectories, the *nearest neighbor* problem is to efficiently report the one most similar to a query trajectory. Trajectories are often represented as polygonal curves, and the nearest neighbor problem is encountered frequently in applications [19, 20, 21].

Various similarity metrics have been proposed for polygonal curves. We are interested in the *Fréchet distance* [3] which has attracted much attention in recent years. It is defined as follows. A parameterization of a curve $\tau$ is a function $\rho : [0, 1] \to \mathbb{R}^d$ such that, as $t$ increases from 0 to 1, the point $\rho(t)$ moves monotonically from the beginning of $\tau$ to its end. We may have $\rho(t_1) = \rho(t_2)$ for two distinct values $t_1$ and $t_2$. Two parameterizations $\rho$ and $\varrho$ for curves $\tau$ and $\sigma$, respectively, induce a *matching* $\mathcal{M}$: for all $t \in [0, 1]$, $\mathcal{M}$ matches $\rho(t)$ with $\varrho(t)$. A point can be matched with multiple partners. The distance between $\tau$ and $\sigma$ under $\mathcal{M}$ is $d_{\mathcal{M}}(\tau, \sigma) = \max_{t\in[0,1]} d(\rho(t), \varrho(t))$, where $d(\cdot, \cdot)$ denotes the Euclidean distance. The Fréchet distance is $d_F(\tau, \sigma) = \min_{\mathcal{M}} d_{\mathcal{M}}(\tau, \sigma)$. We call a minimizing matching a *Fréchet matching*.

Let $T = \{\tau_1, \ldots, \tau_n\}$ be a set of $n$ polygonal curves with at most $m$ vertices each. Given any value $\kappa \geq 1$, the *$\kappa$-approximate nearest neighbor (ANN) problem* is to construct a data structure so that for any query curve $\sigma$, we can quickly report a curve $\tau_l \in T$ with $d_F(\sigma, \tau_l) \leq \kappa \cdot \min_{\tau_i \in T} d_F(\sigma, \tau_i)$. We assume that every query curve has at most $k$ vertices, and $k$ is given for preprocessing. In the literature, if $k = m$, it is called the *symmetric* version; if $k < m$, it is called the *asymmetric version*. If the query curve is sketched by the user, it is likely that $k \ll m$ and this is the scenario for which we design our data structures. We define the $(\kappa, \delta)$-*ANN problem* as follows: for any query curve, we report "no" or a curve $\tau_l \in T$ with $d_F(\sigma, \tau_l) \leq \kappa\delta$; if we report "no", it must be the case that $\min_{\tau_i \in T} d_F(\sigma, \tau_i) > \delta$.

There have been many results on the ANN problem under the *discrete* Fréchet distance $\tilde{d}_F$, which restricts the definition of $d_F$ to parameterizations $\rho$ and $\varrho$ that match each vertex of $\tau$ with at least one vertex of $\sigma$, and vice versa. As a result, $d_F(\tau, \sigma) \leq \tilde{d}_F(\tau, \sigma)$. It is possible that $d_F(\tau, \sigma) \ll \tilde{d}_F(\tau, \sigma)$; for example, $\sigma$ is a long horizontal line segment, and $\tau$ is a parallel copy near $\sigma$ with an extra vertex in the middle. The advantage of $\tilde{d}_F$ is that it can be computed using a simple dynamic programming algorithm [11].

Indyk and Motwani [17] and Har-Peled [14] proved that a solution for the $(\kappa, \delta)$-ANN problem for points in a metric space gives a solution for the $\kappa(1 + O(\varepsilon))$-ANN problem. The result has been simplified in the journal version [15]. The method is general enough that it works for polygonal curves under $d_F$ and $\tilde{d}_F$. Theorem 1 in Section 2 states the deterministic result in our context; the reduction increases the space and query time by polylogarithmic factors. If a probabilistic $(\kappa, \delta)$-ANN solution with failure probability $f$ is used, the bounds in Theorem 1 also hold, and the ANN solution has an $O(f \log n)$ failure probability.

Indyk [16] proposed the first $(\kappa, \delta)$-ANN solution under $\tilde{d}_F$, where $\kappa = O(\log m + \log\log n)$, for the case that $k = m$ and the vertices come from a discrete point set $X$. It uses $O\big(|X|^{\sqrt{m}}(m^{\sqrt{m}}n)^2\big)$ space and answers a query in $O\big(m^{O(1)} \log n\big)$ time.[1] Driemel and Silverstri [10] developed probabilistic $(\kappa, \delta)$-ANN solutions under $\tilde{d}_F$ with a failure probability $1/n$; they achieve the following combinations of ($\kappa$, query time, space) for the case of $k = m$: $\big(4d^{3/2}m, O(m), O(n \log n + mn)\big)$, $\big(4d^{3/2}, O(2^{4dm} m \log n), O(2^{4md} n \log n + mn)\big)$, and $\big(4d^{3/2}m/t, O(2^{2t} m^t \log n), O(2^{2t} m^{t-1} n \log n + mn)\big)$ for any integer $t \geq 1$. The approximation ratio has been reduced to $1 + \varepsilon$ by two research groups later. Filtser et. al. [13] proposed two deterministic $(1+\varepsilon, \delta)$-ANN data structures under $\tilde{d}_F$; one answers a query in $O(kd)$ time and uses $n \cdot O(\frac{1}{\varepsilon})^{kd}$ space and $O(mn(d \log m + O(\frac{1}{\varepsilon})^{kd}))$ expected preprocessing time; the other answers a query in $O(kd \log \frac{dkn}{\varepsilon})$ time and uses $n \cdot O(\frac{1}{\varepsilon})^{kd}$ space and $O(mn \log \frac{n}{\varepsilon} \cdot (d \log m + O(\frac{1}{\varepsilon})^{kd}))$ worst-case preprocessing time. Emiris and Psarros [12] obtained probabilistic $(1+\varepsilon)$-ANN and $(1 + \varepsilon, \delta)$-ANN data structures under $\tilde{d}_F$ with failure probabilities $1/2$ for the case of $k = m$. The $(1+\varepsilon)$-ANN data structure answers a query in $\tilde{O}(d 2^{4m} m^{O(1/\varepsilon)})$ time and uses $\tilde{O}(dm^2 n) \cdot (2 + d/\log m)^{O(dm^{O(1/\varepsilon)} \log(1/\varepsilon))}$ space and preprocessing time. The $(1+\varepsilon, \delta)$-ANN data structure answers a query in $O(d 2^{4m} \log n)$ time and uses $O(dn) + (mn)^{O(m/\varepsilon^2)}$ space and preprocessing time. The failure probabilities can be reduced to $1/n$ with an increase in the query time, space, and preprocessing time by an $O(\log n)$ factor.

Most known results under $d_F$ are for $\mathbb{R}$. For curves in $\mathbb{R}$ (time series), Driemel and Psarros [9] developed the first $(\kappa, \delta)$-ANN data structures under $d_F$ with the following combinations of ($\kappa$, query time, space): $\big(5+\varepsilon, O(k), O(mn) + n \cdot O(\frac{1}{\varepsilon})^k\big)$, $\big(2+\varepsilon, O(2^k k), O(mn) + n \cdot O(\frac{m}{k\varepsilon})^k\big)$, and $\big(24k + 1, O(k \log n), O(n \log n + mn)\big)$. The last one is probabilistic, and the failure probability is $1/\text{poly}(n)$. Later, Bringman et al. [5] obtained improved solutions with the following combinations of ($\kappa$, query time, space): $\big(1 + \varepsilon, O(2^k k), n \cdot O(\frac{m}{k\varepsilon})^k\big)$, $\big(2 + \varepsilon, O(k), n \cdot O(\frac{m}{k\varepsilon})^k\big)$, $\big(2 + \varepsilon, O(2^k k), O(mn) + n \cdot O(\frac{1}{\varepsilon})^k\big)$, $\big(2 + \varepsilon, O(\frac{1}{\varepsilon})^{k+2}, O(mn)\big)$, and $\big(3 + \varepsilon, O(k), O(mn) + n \cdot O(\frac{1}{\varepsilon})^k\big)$. They also obtained lower bounds that are conditioned on the Orthogonal Vectors Hypothesis: for all $\varepsilon, \varepsilon' \in (0, 1)$, it is impossible to achieve the combination $\big(2 - \varepsilon, O(n^{1-\varepsilon'}), \text{poly}(n)\big)$ in $\mathbb{R}$ when $1 \ll k \ll \log n$ and $m = kn^{\Theta(1/k)}$, or $\big(3 - \varepsilon, O(n^{1-\varepsilon'}), \text{poly}(n)\big)$ in $\mathbb{R}$ when $m = k = \Theta(\log n)$, or $\big(3 - \varepsilon, O(n^{1-\varepsilon'}), \text{poly}(n)\big)$ in $\mathbb{R}^2$ when $1 \ll k \ll \log n$ and $m = kn^{\Theta(1/k)}$. Mirzanezhad [18] described a $(1 + \varepsilon, \delta)$-ANN data structure for $\mathbb{R}^d$ that answer a query in $O(kd)$ time and uses $O(n \cdot \max\{\sqrt{d}/\varepsilon, \sqrt{d}D/\varepsilon^2\}^{dk})$

[1] A tradeoff is also presented in [16].

space, where $D$ is the diameter of the set of input curves. If $k$ is not given, the approximation ratio and space increase to $5+\varepsilon$ and $n \cdot O(\frac{1}{\varepsilon})^{dm}$, respectively. There is no bound on $D$ in the space complexity of the first solution. We summarize all these previous results in Table 1 for easier comparison.

**Table 1** Comparison of our data structures to the previous results.

| Distance | Space | Query time | Approximation |
|---|---|---|---|
| Continuous Fréchet, $\mathbb{R}$ | $O(mn) + n \cdot O\left(\frac{1}{\varepsilon}\right)^k$ | $O(k)$ | $(5+\epsilon, \delta)$-ANN [9] |
| | $O(mn) + n \cdot O\left(\frac{m}{k\varepsilon}\right)^k$ | $O(2^k k)$ | $(2+\epsilon, \delta)$-ANN [9] |
| | $O(n \log n + mn)$ | $O(k \log n)$ | $(24k+1, \delta)$-ANN [9][a] |
| | $n \cdot O\left(\frac{m}{k\varepsilon}\right)^k$ | $O(2^k k)$ | $(1+\varepsilon, \delta)$-ANN [5] |
| | $n \cdot O\left(\frac{m}{k\varepsilon}\right)^k$ | $O(k)$ | $(2+\varepsilon, \delta)$-ANN [5] |
| | $O(mn) + n \cdot O\left(\frac{1}{\varepsilon}\right)^k$ | $O(2^k k)$ | $(2+\varepsilon, \delta)$-ANN [5] |
| | $O(mn)$ | $O\left(\frac{1}{\varepsilon}\right)^{k+2}$ | $(2+\varepsilon, \delta)$-ANN [5] |
| | $O(mn) + n \cdot O\left(\frac{1}{\varepsilon}\right)^k$ | $O(k)$ | $(3+\varepsilon, \delta)$-ANN [5] |
| Continuous Fréchet, $\mathbb{R}^d$ | $O\left(n \cdot \max\{\sqrt{d}/\varepsilon, \sqrt{d}D/\varepsilon^2\}^{dk}\right)$ | $O(kd)$ | $(1+\varepsilon, \delta)$-ANN [18] |
| | $n \cdot O\left(\frac{1}{\varepsilon}\right)^{dm}$ | $O(kd)$ | $(5+\varepsilon, \delta)$-ANN [18] |
| | $\tilde{O}\left(k(mnd^d/\varepsilon^d)^{O(k+1/\varepsilon^2)}\right)$ | $\tilde{O}\left(k(mn)^{0.5+\varepsilon}/\varepsilon^d + k(d/\varepsilon)^{O(dk)}\right)$ | $(1+\varepsilon, \delta)$-ANN, Theorem 9 |
| | $\tilde{O}\left(k(mnd^d/\varepsilon^d)^{O(k+1/\varepsilon^2)}\right)$ | $\tilde{O}\left(k(mn)^{0.5+\varepsilon}/\varepsilon^d\right)$ | $(3+\varepsilon, \delta)$-ANN, Theorem 11 |
| Continuous Fréchet, $\mathbb{R}^2$ and $\mathbb{R}^3$ | $O\left(\frac{1}{\varepsilon}\right)^{4d(k-1)+1} (mn)^{4(k-1)} k \log^2 n$ | $O\left(\frac{1}{\varepsilon}^{2d(k-2)}\right) k \log \frac{mn}{\varepsilon} \log n$ | $(1+\varepsilon, \delta)$-ANN, Theorem 9 |
| | $O\left(\frac{1}{\varepsilon}\right)^{2d(k-1)+1} (mn)^{2(k-1)} k \log^2 n$ | $O\left(k \log \frac{mn}{\varepsilon} \log n\right)$ | $(3+\varepsilon, \delta)$-ANN, Theorem 11 |
| Discrete Fréchet, $\mathbb{R}^d$ | $O\left(\lvert X\rvert^{\sqrt{m}}(m^{\sqrt{m}}n)^2\right)$ | $O\left(m^{O(1)} \log n\right)$ | $(O(\log m + \log\log n), \delta)$-ANN [16] |
| | $O(n \log n + mn)$ | $O(m)$ | $(4d^{3/2}m, \delta)$-ANN [10] |
| | $O(2^{4md} n \log n + mn)$ | $O(2^{4dm} m \log n)$ | $(4d^{3/2}, \delta)$-ANN [10] |
| | $O(2^{2t} m^{t-1} n \log n + mn)$ | $O(2^{2t} m^t \log n)$ | $(4d^{3/2}m/t, \delta)$-ANN [10] |
| | $n \cdot O\left(\frac{1}{\varepsilon}\right)^{kd}$ | $O(kd)^b$ | $(1+\varepsilon, \delta)$-ANN [13] |
| | $O(dn) + (mn)^{O(m/\varepsilon^2)}$ | $O\left(d 2^{4m} \log n\right)$ | $(1+\varepsilon, \delta)$-ANN[c] [12] |

[a] A randomized data structure with a failure probability of $1/\text{poly}(n)$.
[b] The query time is achieved by implementing the dictionary with a hash table. The query time is $O(kd \log \frac{dkn}{\varepsilon})$ when implementing the dictionary with a trie.
[c] A randomized data structure with a failure probability of $\frac{1}{2}$.

We develop $(\kappa, \delta)$-ANN data structures under $d_F$ in $\mathbb{R}^d$ for $\kappa \in \{1+\varepsilon, 3+\varepsilon\}$ and $d \geq 2$. We assume that every query curve has at most $k$ vertices, $k \ll m$, and $k$ is given for preprocessing. To simplify the bounds, we assume that $k \geq 3$ throughout this paper. There are three design goals. First, the query times are sublinear in $mn$. Second, the space complexities depend only on the input parameters. Third, the space complexities are neither proportional to $\min\{m^{\Omega(d)}, n^{\Omega(d)}\}$ nor exponential in $\min\{m, n\}$. It would be desirable to remove all exponential dependencies on $d$, but we are not there yet.

We achieve a query time of $\tilde{O}(k(mn)^{0.5+\varepsilon}/\varepsilon^d + k(d/\varepsilon)^{O(dk)})$ for $(1+\varepsilon, \delta)$-ANN. We remove the exponential dependence on $k$ for $(3+\varepsilon, \delta)$-ANN and obtain an $\tilde{O}(k(mn)^{0.5+\varepsilon}/\varepsilon^d)$ query time. The space and expected preprocessing time are $\tilde{O}(k(mnd^d/\varepsilon^d)^{O(k+1/\varepsilon^2)})$ in both cases. For ease of presentation, we treat any factor in our bounds that depends only on $d$ as $O(1)$. The hidden polylog factors in the big-$\tilde{O}$ notation have powers dependent on $d$. In two and three dimensions, we improve the query times to $O(1/\varepsilon)^{O(k)} \cdot \tilde{O}(k)$ for $(1+\varepsilon, \delta)$-ANN and $\tilde{O}(k)$ for $(3+\varepsilon, \delta)$-ANN. The space and expected preprocessing time improve to $O(mn/\varepsilon)^{O(k)} \cdot \tilde{O}(k)$ in both cases. Using the reduction in [15] (Theorem 1 in Section 2), we obtain $(1+\varepsilon)$-ANN and $(3+\varepsilon)$-ANN data structures by increasing the query time and space by an $O(\log n)$ and an $O(\frac{1}{\varepsilon}\log^2 n)$ factors, respectively. More precise bounds are stated in Theorems 9 and 11.

Our $(1+\varepsilon, \delta)$-ANN result is based on two new ideas. First, we develop a novel encoding of query curves that are based on local grids in the input vertex neighborhoods. Second, we draw a connection to an approximate segment shooting problem which we solve efficiently. We present these ideas in Sections 2 and 4. Our $(3+\varepsilon)$-ANN result is obtained by simplifying the encoding. We present this result in Section 3.

We work in the word RAM model. We use $(v_{i,1}, \ldots, v_{i,m})$ to denote the sequence of vertices of $\tau_i$ from beginning to end – $\tau_i$ is oriented from $v_{i,1}$ to $v_{i,m}$. We use $\tau_{i,a}$ to denote the edge $v_{i,a}v_{i,a+1}$. For any two points $x, y \in \tau_i$, we say that $x \leq_{\tau_i} y$ if $x$ does not appear behind $y$ along $\tau_i$, and $\tau_i[x, y]$ denotes the subcurve between $x$ and $y$. Given two subsets $X, Y \subseteq \tau_i$, $X \leq_{\tau_i} Y$ if and only if for every point $x \in X$ and every point $y \in Y$, $x \leq_{\tau_i} y$. A ball centered at the origin with radius $r$ is denoted by $B_r$. Given two subsets $X, Y \subset \mathbb{R}^d$, $d(X, Y) = \min_{x\in X, y\in Y} d(x, y)$; their *Minkowski sum* is $X \oplus Y = \{x + y : x \in X, y \in Y\}$; if $X = \{p\}$, we write $p \oplus Y$ for simplicity. For any $x, y \in \mathbb{R}^d$, $xy$ denotes the *oriented segment from $x$ to $y$*, and $\mathrm{aff}(xy)$ is the *oriented support line of $xy$* that shares the orientation of $xy$.

## 2 $(1 + O(\varepsilon), \delta)$-ANN

Har-Peled et al. [15, Theorem 2.10] proved a reduction from the $(1 + \varepsilon)$-ANN problem to the $(1 + \varepsilon, \delta)$-ANN problem. Although the result is described for points in a metric space with a probabilistic data structure for the $(1 + \varepsilon, \delta)$-ANN problem, the method is general enough to work for polygonal curves under $d_F$ or $\tilde{d}_F$ in $\mathbb{R}^d$ and any deterministic solution for the $(1 + \varepsilon, \delta)$-ANN problem. We rephrase their result in our context below.

▶ **Theorem 1** ([15]). *Let $T$ be a set of $n$ polygonal curves in $\mathbb{R}^d$. If there is a data structure for the $(\kappa, \delta)$-ANN problem for $T$ under $d_F$ or $\tilde{d}_F$ that has space complexity $S$, query time $Q$, deletion time $D$, and preprocessing time $P$, then there is a $\kappa(1 + O(\varepsilon))$-ANN data structure for $T$ under $d_F$ or $\tilde{d}_F$ that has space complexity $O(\frac{1}{\varepsilon} S \log^2 n)$, query time $O(Q \log n)$, and expected preprocessing time $O\big(\frac{1}{\varepsilon \log^2 n} P + (Q + D)n \log n\big)$.*

By Theorem 1, we can focus on the $(1 + \varepsilon, \delta)$-ANN problem. Without loss of generality, we assume that each curve in $T$ has exactly $m$ vertices, and every query curve has exactly $k$ vertices. If necessary, extra vertices can be added in an arbitrary manner to enforce this assumption without affecting the Fréchet distance.

The high level idea of our preprocessing is to identify all query curves that are within a Fréchet distance $(1 + O(\varepsilon))\delta$ from each $\tau_i \in T$, group the curves that share similar structural characteristics, assign each group a unique key value, and store these key values in a trie $\mathcal{D}$. It is possible for a query curve to belong to multiple groups. Each key value is associated with the subset of curves in $T$ that induce that key value. Correspondingly, given a query curve $\sigma$, we generate all possible key values for $\sigma$ and search $\mathcal{D}$ with them. If some curve in $T$ is retrieved, it is the desired answer; otherwise, we report "no".

There are two challenges to overcome. First, it is impossible to examine all possible query curves. We can only check some space discretization in order to obtain a finite running time. To control the discretization error, it is easy to cover the input curves by a grid with an appropriate cell width; however, the grid size and hence the data structure size would then depend on some non-combinatorial parameters. We propose *coarse encodings* of query curves so that there are $O(\sqrt{d}/\varepsilon)^{4d(k-1)}(mn)^{4(k-1)}$ of them. A query curve may have $O(\sqrt{d}/\varepsilon)^{2d(k-2)}$ coarse encodings. The second challenge is to efficiently generate all possible coarse encodings of a query curve at query time. We reduce the coarse encoding generation to an approximate segment shooting problem. This step turns out to be the bottleneck in four and higher dimensions as we aim to avoid any factor of the form $m^{\Omega(d)}$ or $n^{\Omega(d)}$ in the space complexity. It is the reason for the $(mn)^{0.5+\varepsilon}$ term in the query time. In two and three dimensions, the approximate segment shooting problem can be solved more efficiently.

In the rest of this section, we present the coarse encoding and a $(1 + O(\varepsilon), \delta)$-ANN data structure, using an approximate segment shooting oracle. The approximate segment shooting problem can be solved by the results in [8] in two and three dimensions. We solve the approximate segment shooting problem in four and higher dimensions in Section 4.

## 2.1 Coarse encodings of query curves

Imagine an infinite grid in $\mathbb{R}^d$ of cell width $\varepsilon\delta/\sqrt{d}$. For any subset $R \subset \mathbb{R}^d$, we use $G(R)$ to denote the set of grid cells that intersect $R$. Let $\mathcal{G}_1 = \bigcup_{i\in[n],a\in[m]} G(v_{i,a} \oplus B_\delta)$. Let $\mathcal{G}_2 = \bigcup_{i\in[n],a\in[m]} G(v_{i,a} \oplus B_{(2+12\varepsilon)\delta})$. Both $\mathcal{G}_1$ and $\mathcal{G}_2$ have $O(mn/\varepsilon^d)$ size.

The coarse encoding of a curve $\sigma = (w_1, w_2, \ldots, w_k)$ is a 3-tuple $\mathcal{F} = (\mathcal{A}, \mathcal{B}, \mathcal{C})$. The component $\mathcal{C}$ is sequence of pairs of grid cells $((c_{j,1}, c_{j,2}))_{j\in[k-1]}$ such that $(c_{j,1}, c_{j,2}) \in (\mathcal{G}_1 \times \mathcal{G}_1) \cup \{\text{null}\}$. Both $\mathcal{A}$ and $\mathcal{B}$ are arrays of length $k-1$, and every element of $\mathcal{A}$ and $\mathcal{B}$ belongs to $\mathcal{G}_2 \cup \{\text{null}\}$. We first provide the intuition behind the design of $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ before describing the constraints that realize the intuition.

Imagine that a curve $\tau_i \in T$ is a $(1 + O(\varepsilon))$-ANN of $\sigma$. The data structure needs to cater for the preprocessing, during which the query curve $\sigma$ is not available; it also needs to cater for the query procedure, during which we do not want to directly consult the input curves in $T$ in order to avoid a linear dependence in $mn$.

In preprocessing, we use pairs of grid cells as surrogates of the possible query curve edges. The advantage is that we can enumerate all possible pairs of grid cells and hence cater for all possible query curve edges. Specifically, for $j \in [k-1]$, if $(c_{j,1}, c_{j,2}) \neq \text{null}$, it is the surrogate of $w_j w_{j+1}$, so $w_j w_{j+1}$ should pass near $c_{j,1}$ and $c_{j,2}$. Each non-null $(c_{j,1}, c_{j,2})$ corresponds to a contiguous subsequence $v_{i,a}, \ldots, v_{i,b}$ of vertices of $\tau_i$ that are matched to points in $w_j w_{j+1}$ in a Fréchet matching. Of course, we do not know the Fréchet matching, so we will need to enumerate and handle all possibilities. Also, since $w_j w_{j+1}$ is unknown in preprocessing, $v_{i,a}, \ldots, v_{i,b}$ can only be matched to a segment joining a vertex $x_j$ of $c_{j,1}$ to a vertex $y_j$ of $c_{j,2}$ so that $d_F(x'_j y'_j, \tau_i[v_{i,a}, v_{i,b}]) \leq (1 + O(\varepsilon))\delta$ for some subsegment $x'_j y'_j \subseteq x_j y_j$. This property will be enforced in the data structure construction later.

At query time, given $\sigma = (w_1, \ldots, w_k)$, we will make approximate segment shooting queries to determine a sequence of cell pairs $((c_{j,1}, c_{j,2}))_{j\in[k-1]}$. We do not always use $(c_{j,1}, c_{j,2})$ as a surrogate for the edge $w_j w_{j+1}$ though. As mentioned in the previous paragraph, a non-null $(c_{j,1}, c_{j,2})$ denotes the matching of a contiguous subsequence of input curve vertices to $w_j w_{j+1}$; however, we must also allow the matching of a contiguous subsequence of vertices of $\sigma$ to a single input edge. Therefore, after determining $((c_{j,1}, c_{j,2}))_{j\in[k-1]}$, we still have the choice of using $(c_{j,1}, c_{j,2})$ as is or substituting it by the null value. For a technical reason, $(c_{1,1}, c_{1,2})$ and $(c_{k-1,1}, c_{k-1,2})$ are always kept non-null, so we have $2^{k-3}$ possible sequences of pairs of cells. Take one of these sequences. If $(c_{r,1}, c_{r,2})$ and $(c_{s,1}, c_{s,2})$ are two non-null pairs such that $(c_{j,1}, c_{j,2}) = \text{null}$ for $j \in [r+1, s-1]$, it means that no vertex of $\tau_i$ is matched to $w_j w_{j+1}$ for $j \in [r+1, s-1]$. As a result, the vertices $w_{r+1}, \ldots, w_s$ of $\sigma$ are matched to the edge $v_{i,b} v_{i,b+1}$ of $\tau_i$, where $v_{i,b}$ is the last vertex of $\tau_i$ matched to $w_r w_{r+1}$ in the current enumeration. We use the pair of cells $\mathcal{A}[r]$ and $\mathcal{B}[s]$ as the surrogate of the edge $v_{i,b} v_{i,b+1}$. So we require $\mathcal{A}[r]$ and $\mathcal{B}[s]$ to be near $c_{r,2}$ and $c_{s,1}$, respectively, because $(c_{r,1}, c_{r,2})$ is the surrogate of $w_r w_{r+1}$, and $(c_{s,1}, c_{s,2})$ is the surrogate of $w_s w_{s+1}$. We have to try all possible locations of $\mathcal{A}[r]$ and $\mathcal{B}[s]$ in the vicinity of $c_{r,2}$ and $c_{s,1}$. $\mathcal{A}[r]$ and $\mathcal{B}[s]$ can be the surrogate for edges of multiple curves in $T$, which allows us to compare $\sigma$ with multiple input curves simultaneously at query time. The constraint to be enforced is that $w_{r+1}, \ldots, w_s$ can be matched to a segment joining a vertex $x_r$ of $\mathcal{A}[r]$ and a vertex $x_s$ of $\mathcal{B}[s]$ so that $d_F(x'_r x'_s, \sigma[w_{r+1}, w_s]) \leq (1 + O(\varepsilon))\delta$ for some subsegment $x'_r x'_s \subseteq x_r x_s$. Figure 1 shows a illustration for the intuition above.

We present the constraints for $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ that realize the intuition above. When $(c_{j,1}, c_{j,2}) \neq \text{null}$, a natural choice of $c_{j,1}$ is the first grid cell in $\mathcal{G}_1$ that we hit when walking from $w_j$ to $w_{j+1}$, i.e., segment shooting. In $\mathbb{R}^d$ where $d \in \{2, 3\}$, there are ray shooting data structures for boxes [8]. In higher dimensions, ray shooting results are known for a single convex

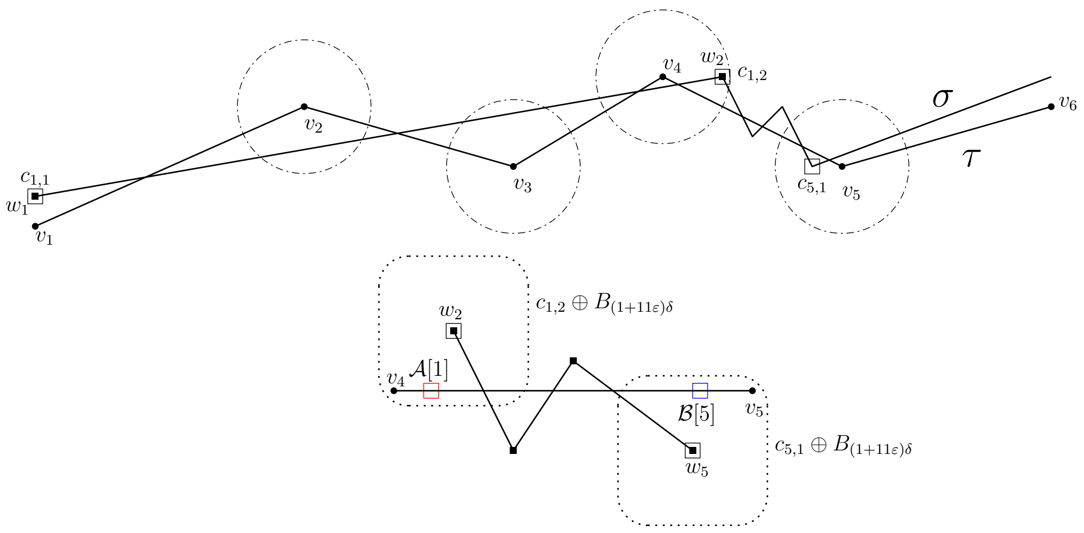


**Figure 1** The underlying intuition for deriving the coarse encoding of $\sigma$. We use $\tau$ instead of $\tau_i$ for ease of notation. Assume that $d_F(\tau, \sigma) \leq \delta$ and the vertices $v_1, v_2, v_3, v_4$ are matched to the segment $w_1w_2$ by the Fréchet matching. $w_1w_2$ must intersect some balls centered at $\tau$'s vertices, which means that $w_1w_2$ intersects $\mathcal{G}_1$. Let $c_{1,1}$ and $c_{1,2}$ be the first and the last cells in $\mathcal{G}_1$ that intersect $w_1w_2$ along the direction of $w_1w_2$. $(c_{1,1}, c_{1,2})$ can serve as a surrogate of the edge $w_1w_2$ in a sense that we can verify whether $v_1, v_2, v_3, v_4$ can be matched to $w_1w_2$ properly by checking whether they can be matched to a segment that joins vertices of $c_{1,1}$ and $c_{1,2}$ properly. This idea can be generalized to all edges of $\sigma$ with vertices of $\tau$ matched to them. The subcurve $\sigma[w_2, w_5]$ is matched to an edge $v_4v_5$ of $\tau$. We introduce $\mathcal{A}[1] \subset G(c_{1,2} \oplus B_{(1+11\varepsilon)\delta})$ and $\mathcal{B}[5] \subset G(c_{5,1} \oplus B_{(1+11\varepsilon)\delta})$. $(\mathcal{A}[1], \mathcal{B}[5])$ can serve as a surrogate of $v_4v_5$. $(\mathcal{A}[1], \mathcal{B}[5])$ can encode $\sigma[w_2, w_5]$ sufficiently because for every segment $x_1x_5$ that joins a vertex $x_1$ of $\mathcal{A}[1]$ and a vertex $x_5$ of $\mathcal{B}[5]$, there exists a subsegment $x_1'x_5' \subset x_1x_5$ such that $d_F(x_1'x_5', \sigma[w_2, w_5]) \leq (1 + O(\varepsilon))\delta$.

polytope and an arrangement of hyperplanes [1]; even in such cases, the query time is substantially sublinear only if the space complexity is at least the input size raised to a power of $\Omega(d)$. It would be $(mn)^{\Omega(d)}$ in our case. We define a $\lambda$-segment query problem below that approximates the ray shooting problem, and we will present an efficient solution for $\lambda = 11\varepsilon\delta$ in Section 4 that avoids an $(mn)^{\Omega(d)}$ term in the space complexity. As mentioned before, the ray shooting result in [8] suffices in two and three dimensions.

**$\lambda$-segment query.** A set $O$ of objects in $\mathbb{R}^d$ is preprocessed into a data structure so that for any oriented query segment $pq$, the *$\lambda$-segment query with $pq$ on $O$* returns one of the following answers:

- If $pq$ intersects an object in $O$, let $x$ be the first intersection point with an object in $O$ as we walk from $p$ to $q$. In this case, the query returns an object $o \in O$ such that $px$ intersects $o \oplus B_\lambda$. Figure 2 shows an illustration.
- Otherwise, the query returns null or an object $o \in O$ such that $d(o, pq) \leq \lambda$.

We are now ready to state the three constraints on $(\mathcal{A}, \mathcal{B}, \mathcal{C})$.

- **Constraint 1:**
  - **(a)** Both $(c_{1,1}, c_{1,2})$ and $(c_{k-1,1}, c_{k-1,2})$ belong to $\mathcal{G}_1 \times \mathcal{G}_1$.
  - **(b)** For $j \in [k-1]$, if $(c_{j,1}, c_{j,2}) \neq$ null, then:
    - **(i)** $c_{j,1}$ and $c_{j,2}$ are the grid cells returned by the $(11\varepsilon\delta)$-segment queries with $w_jw_{j+1}$ and $w_{j+1}w_j$ on $\mathcal{G}_1$, respectively;
    - **(ii)** the minimum point in $w_jw_{j+1} \cap (c_{j,1} \oplus B_{11\varepsilon\delta})$ lies in front of the maximum point in $w_jw_{j+1} \cap (c_{j,2} \oplus B_{11\varepsilon\delta})$ with respect to $\leq_{w_jw_{j+1}}$.

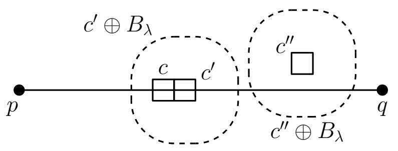


**Figure 2** The $\lambda$-segment query with $pq$ on the boxes $\{c, c', c''\}$ can return $c$ or $c'$ but not $c''$.

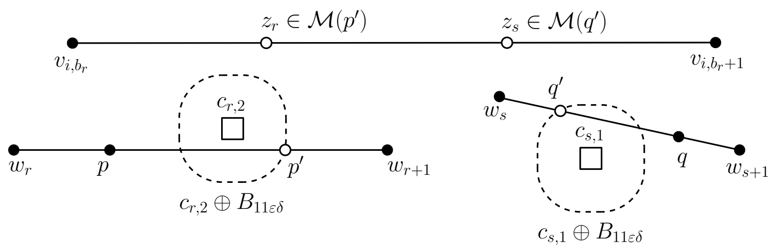
- **Constraint 2:**
  **(a)** $\mathcal{B}[1]$ and $\mathcal{A}[k-1]$ belong to $\mathcal{G}_2$.
  **(b)** $w_1 \in \mathcal{B}[1]$ and $w_k \in \mathcal{A}[k-1]$.
- **Constraint 3:**
  **(a)** For $j \in [2, k-2]$, if $(c_{j,1}, c_{j,2}) = \text{null}$, then $\mathcal{A}[j]$ and $\mathcal{B}[j]$ are null.
  **(b)** For $j \in [k-1]$, if $(c_{j,1}, c_{j,2}) \neq \text{null}$, then $\mathcal{A}[j]$ and $\mathcal{B}[j]$ belong to $\mathcal{G}_2$, $d(c_{j,1}, \mathcal{B}[j]) \leq (1+11\varepsilon)\delta$, and $d(c_{j,2}, \mathcal{A}[j]) \leq (1+11\varepsilon)\delta$.
  **(c)** Let $\mathcal{J}$ be the set of $(r,s) \in [k-1] \times [k-1]$ such that $r < s$, $(c_{r,1}, c_{r,2}) \neq \text{null}$, $(c_{s,1}, c_{s,2}) \neq \text{null}$, and $(c_{j,1}, c_{j,2}) = \text{null}$ for $j \in [r+1, s-1]$. For every $(r,s) \in \mathcal{J}$, let $x_r$ and $x_s$ be the smallest vertices of $\mathcal{A}[r]$ and $\mathcal{B}[s]$ according to the lexicographical order of their coordinates, there exist $x'_r, x'_s \in x_r x_s$ such that $x'_r \leq_{x_r x_s} x'_s$ and $d_F(x'_r x'_s, \sigma[w_{r+1}, w_s]) \leq (1+\varepsilon)\delta$.

We remark that if $w_j w_{j+1}$ intersects the interior of the union of cells in $\mathcal{G}_1$, constraint 1(b)(ii) is satisfied automatically for $(c_{j,1}, c_{j,2})$ given constraint 1(b)(i). When $w_j w_{j+1}$ does not intersect the interior of the union of cells in $\mathcal{G}_1$, it is possible that the $(11\varepsilon\delta)$-segment queries return two cells that violate constraint 1(b)(ii). In this case, the input vertices are too far from $w_j w_{j+1}$ to be matched to any point in $w_j w_{j+1}$ within a distance $\delta$, so we can set $(c_{j,1}, c_{j,2})$ to be null.

The next result shows that any query curve $\sigma$ near a curve $\tau_i \in T$ has a coarse encoding with some additional properties. These properties will be useful in the analysis. Let $\mathcal{M}$ denote a matching between $\sigma$ and some $\tau_i \in T$. For any subcurve $\sigma' \subseteq \sigma$, we use $\mathcal{M}(\sigma')$ to denote the subcurve of $\tau_i$ matched to $\sigma'$ by $\mathcal{M}$.

▶ **Lemma 2.** *Let $\sigma = (w_1, \ldots, w_k)$ be a curve of $k$ vertices. Let $\mathcal{M}$ be a matching between $\sigma$ and $\tau_i \in T$ such that $d_{\mathcal{M}}(\tau_i, \sigma) \leq \delta$. Let $\tilde{\pi}_j = \{v_{i,a} : a \in [m-1], v_{i,a} \in \mathcal{M}(w_j w_{j+1}) \setminus \mathcal{M}(w_j)\}$ for all $j \in [k-1]$. Define $\pi_j = \tilde{\pi}_j$ for all $j \in [k-2]$, $\pi_{k-1} = \{v_{i,m}\} \cup \tilde{\pi}_{k-1}$, and $\pi_0 = \{v_{i,1} \ldots, v_{i,m}\} \setminus \bigcup_{j=1}^{k-1} \pi_j$. There is a coarse encoding $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ for $\sigma$ that satisfies the following properties.*
**(i)** *For $j \in [2, k-1]$, $\pi_j = \emptyset$ if and only if $(c_{j,1}, c_{j,2}) = \text{null}$.*
**(ii)** *For all $(r,s) \in \mathcal{J}$, if $r = 1$ and $\pi_1 = \emptyset$, let $b_1 = 1$; otherwise, let $b_r = \max\{b : v_{i,b} \in \pi_r\}$. For all $(r,s) \in \mathcal{J}$, there exist a point $z \in \mathcal{A}[r] \cap \tau_{i,b_r}$ and another point $z' \in \mathcal{B}[s] \cap \tau_{i,b_r}$ such that $z \leq_{\tau_{i,b_r}} z'$.*

**Proof.** We define the component $\mathcal{C}$ as follows. Given that $v_{i,1} \in \mathcal{M}(w_1)$, $w_1 w_2$ intersects the interior of the union of cells in $\mathcal{G}_1$, so the $(11\varepsilon\delta)$-segment query with $w_1 w_2$ on $\mathcal{G}_1$ must return some cell; we define it to be $c_{1,1}$. Similarly, the $(11\varepsilon\delta)$-segment query with $w_2 w_1$ on $\mathcal{G}_1$ must return some cell; we define it to be $c_{1,2}$. The pair $(c_{k-1,1}, c_{k-1,2})$ are also defined in a similar way as $v_{i,m} \in \mathcal{M}(w_k)$. Consider any $j \in [2, k-1]$. If $v_{i,a} \in \pi_j$ for some $a \in [m]$, then $v_{i,a} \in \mathcal{M}(w_j w_{j+1})$, which implies that $w_j w_{j+1}$ intersects $v_{i,a} \oplus B_\delta$ and hence the interior of the union of cells in $\mathcal{G}_1$. Thus, $(c_{j,1}, c_{j,2})$ can be defined using the $(11\varepsilon\delta)$-segment queries with $w_j w_{j+1}$ and $w_{j+1} w_j$ as before. On the other hand, if $\pi_j = \emptyset$, we define $(c_{j,1}, c_{j,2})$ to be null. Constraint 1 and property (i) in the lemma are thus satisfied.



$z_r \in \mathcal{M}(p')$
$z_s \in \mathcal{M}(q')$
$v_{i,b_r}$
$v_{i,b_r+1}$
$c_{r,2}$
$q'$
$w_s$
$c_{s,1}$
$q$
$w_{s+1}$
$w_r$
$p$
$p'$
$w_{r+1}$
$c_{r,2} \oplus B_{11\varepsilon\delta}$
$c_{s,1} \oplus B_{11\varepsilon\delta}$

**Figure 3** Illustration of $p$, $p'$, $q'$, $q$, $z_r$, and $z_s$.

Next, we define $\mathcal{A}$ and $\mathcal{B}$ to satisfy constraints 2 and 3.

As $v_{i,1} \in \mathcal{M}(w_1)$ and $v_{i,m} \in \mathcal{M}(w_k)$, both $d(w_1, v_{i,1})$ and $d(w_k, v_{i,m})$ are at most $\delta$. So $w_1$ lies in a cell in $G(v_{i,1} \oplus B_\delta) \subset G(v_{i,1} \oplus B_{(2+12\varepsilon)\delta}) \subset \mathcal{G}_2$; we make this cell $\mathcal{B}[1]$. Similarly, we define $\mathcal{A}[k-1]$ to be the cell in $\mathcal{G}_2$ that contains $w_k$. Constraint 2 is thus enforced.

For $j \in [2, k-2]$, if $(c_{j,1}, c_{j,2}) = \text{null}$, let $\mathcal{A}[j]$ and $\mathcal{B}[j]$ be null, satisfying constraint 3(a). $\mathcal{B}[1]$ and $\mathcal{A}[k-1]$ have already been defined, and they belong to $\mathcal{G}_2$. Since $w_1$ lies in a cell in $G(v_{i,1} \oplus B_\delta) \subset \mathcal{G}_1$, we have $w_1 \in c_{1,1} \oplus B_{11\varepsilon\delta}$ by the $(11\varepsilon\delta)$-segment query. Then, $d(c_{1,1}, \mathcal{B}[1]) \leq 11\varepsilon\delta$ as $w_1 \in \mathcal{B}[1]$. Similarly, $d(c_{k-1,2}, \mathcal{A}[k-1]) \leq 11\varepsilon\delta$. So $\mathcal{B}[1]$ and $\mathcal{A}[k-1]$ satisfy constraint 3(b). It remains to discuss $\mathcal{A}[j]$ for $j \in [1, k-2]$ and $\mathcal{B}[j]$ for $j \in [2, k-1]$.

Consider an arbitrary $j_* \in [k-1]$ such that $(c_{j_*,1}, c_{j_*,2}) \neq \text{null}$. Recall that $\mathcal{J}$ is the set of $(r,s) \in [k-1] \times [k-1]$ such that $r < s$, $(c_{r,1}, c_{r,2}) \neq \text{null}$, $(c_{s,1}, c_{s,2}) \neq \text{null}$, and $(c_{j,1}, c_{j,2}) = \text{null}$ for $j \in [r+1, s-1]$. Thus, if $j_* \leq k-2$, it must exist as the first value in exactly one element of $\mathcal{J}$, and if $j_* \geq 2$, it must also exist as the second value in exactly another element of $\mathcal{J}$. As a result, it suffices to define $\mathcal{A}[r]$ and $\mathcal{B}[s]$ for every $(r,s) \in \mathcal{J}$ and verify that constraints 3(b) and 3(c) are satisfied.

Take any $(r,s) \in \mathcal{J}$. If $\pi_r \neq \emptyset$, it is legal to define $b_r = \max\{b : v_{i,b} \in \pi_r\}$. If $\pi_r = \emptyset$, then $r = 1$ because for any $r > 1$, $\pi_r \neq \text{null}$ by (i) as $(c_{r,1}, c_{r,2}) \neq \text{null}$ by the definition of $\mathcal{J}$. In the case that $r = 1$ and $\pi_1 = \emptyset$, $b_1$ is defined to be 1. Therefore, $b_r$ is well defined for all $(r,s) \in \mathcal{J}$. The definition of $b_r$ implies that $b_r = \max\{b : v_{i,b} \in \mathcal{M}(w_r w_{r+1})\}$. Since $(c_{s,1}, c_{s,2}) \neq \text{null}$ and $(c_{j,1}, c_{j,2}) = \text{null}$ for $j \in [r+1, s-1]$, by (i), $\pi_s \neq \emptyset$ and $\pi_j = \emptyset$ for $j \in [r+1, s-1]$. It follows that $v_{i,b_r+1} \in \pi_s$ which is a subset of $\mathcal{M}(w_s w_{s+1})$. Pick any point $p \in w_r w_{r+1}$ and any point $q \in w_s w_{s+1}$ such that $v_{i,b_r} \in \mathcal{M}(p)$ and $v_{i,b_r+1} \in \mathcal{M}(q)$.

We claim that $pw_{r+1} \cap (c_{r,2} \oplus B_{11\varepsilon\delta})$ and $w_s q \cap (c_{s,1} \oplus B_{11\varepsilon\delta})$ are non-empty. Since $c_{r,2}$ is the cell in $\mathcal{G}_1$ returned by the $(11\varepsilon\delta)$-segment query with $w_{r+1}w_r$, for any intersection point $x$ between $w_{r+1}w_r$ and any cell in $\mathcal{G}_1$, we have $xw_{r+1} \cap (c_{r,2} \oplus B_{11\varepsilon\delta}) \neq \emptyset$ by definition. We have $p \in w_r w_{r+1} \cap (v_{i,b_r} \oplus B_\delta)$ by our choice of $p$; it means that $p$ is an intersection point between $w_r w_{r+1}$ and a cell in $G(v_{i,b_r} \oplus B_\delta) \subset \mathcal{G}_1$. We can thus substitute $p$ for $x$ and conclude that $pw_{r+1} \cap (c_{r,2} \oplus B_{11\varepsilon\delta}) \neq \emptyset$. Similarly, we get $w_s q \cap (c_{s,1} \oplus B_{11\varepsilon\delta}) \neq \emptyset$.

By our claim, when we walk from $w_{r+1}$ to $p$, we hit $c_{r,2} \oplus B_{11\varepsilon\delta}$ at some point $p'$, and when we walk from $w_s$ to $q$, we hit $c_{s,1} \oplus B_{11\varepsilon\delta}$ at some point $q'$. Pick two points $z_r \in \mathcal{M}(p')$ and $z_s \in \mathcal{M}(q')$. By definition, $c_{r,2} \in G(v_{i_r,a_r} \oplus B_\delta)$ for some $\tau_{i_r} \in T$ and some index $a_r \in [m]$. The cell width of $c_{r,2}$ is $\varepsilon\delta/\sqrt{d}$, so $c_{r,2} \subset v_{i_r,a_r} \oplus B_{(1+\varepsilon)\delta}$. By triangle inequality, $p' \in v_{i_r,a_r} \oplus B_{(1+12\varepsilon)\delta}$ and hence $z_r \in v_{i_r,a_r} \oplus B_{(2+12\varepsilon)\delta}$, which implies that $z_r$ is contained in a cell in $G(v_{i_r,a_r} \oplus B_{(2+12\varepsilon)\delta}) \subset \mathcal{G}_2$. By a similar reasoning, we can also deduce that $z_s$ is contained in a cell in $\mathcal{G}_2$. We define $\mathcal{A}[r]$ and $\mathcal{B}[s]$ to be the cells in $\mathcal{G}_2$ that contain $z_r$ and $z_s$, respectively. Figure 3 shows an illustration.

Since $d(c_{r,2}, z_r) \leq d(c_{r,2}, p') + d(p', z_r) \leq (1 + 11\varepsilon)\delta$, we get $d(c_{r,2}, \mathcal{A}[r]) \leq (1 + 11\varepsilon)\delta$. Similarly, $d(c_{s,1}, \mathcal{B}[s]) \leq (1 + 11\varepsilon)\delta$. This takes care of constraint 3(b).

As $v_{i,b_r} \in \mathcal{M}(p)$ and $p \leq_{w_r w_{r+1}} p'$, we have $v_{i,b_r} \leq_{\tau_i} \mathcal{M}(p')$. Similarly, we have $\mathcal{M}(q') \leq_{\tau_i} v_{i,b_{r+1}}$. Therefore, $v_{i,b_r} \leq_{\tau_i} \mathcal{M}(p') \leq_{\tau_i} \mathcal{M}(q') \leq_{\tau_i} v_{i,b_{r+1}}$. As $z_r \in \mathcal{M}(p')$ and $z_s \in \mathcal{M}(q')$, $z_r$ and $z_s$ satisfy property (ii) of the lemma. The distance between $z_r$ and any vertex $x_r$ of $\mathcal{A}[r]$ is at most $\varepsilon\delta$. So is the distance between $z_s$ and any vertex $x_s$ of $\mathcal{B}[s]$. Thus, we can use the linear interpolation $\mathcal{I}$ between $x_r x_s$ and $z_r z_s$ as a matching to get $d_{\mathcal{I}}(x_r x_s, z_r z_s) \leq \varepsilon\delta$. Combining $\mathcal{M}$ and $\mathcal{I}$ shows that there is a matching between $\sigma[p', q']$ and $x_r x_s$ within a distance of $(1+\varepsilon)\delta$. Since $\sigma[w_{r+1}, w_s] \subseteq \sigma[p', q']$, we have thus verified constraint 3(c). ◀

## 2.2 Data structure organization and construction

We construct $\mathcal{G}_1$ and $\mathcal{G}_2$ in $O(mn/\varepsilon^d)$ time and space. We need a point location data structure for $\mathcal{G}_2$ which is organized as a multi-level tree as follows. The top-level tree has leaves corresponding to the intervals of the cells on the first coordinate axis. Each leaf is associated with the cells that project to the interval of that leaf, and these cells are stored in a second-level tree with leaves corresponding to the intervals of these cells on the second coordinate axis. Continuing in this manner yields $d = O(1)$ levels, using $O(|\mathcal{G}_2|) = O(mn/\varepsilon^d)$ space and $O\big((mn/\varepsilon^d)\log\frac{mn}{\varepsilon}\big)$ preprocessing time. A point location takes $O(\log\frac{mn}{\varepsilon})$ time.

The $(1+O(\varepsilon), \delta)$-ANN data structure is a trie $\mathcal{D}$. Each key to be stored in $\mathcal{D}$ is a *candidate* coarse encoding, which is a 3-tuple $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ just like a coarse encoding. For a candidate coarse encoding, constraints 1(a), 2(a), 3(a), and 3(b) must be satisfied, but constraints 1(b), 2(b), and 3(c) are ignored. This difference is necessary because constraints 1(b), 2(b), and 3(c) require the query curve, which is not available in preprocessing. For each candidate coarse encoding $E$, let $T_E$ be the subset of input curves that are within a Fréchet distance of $(1+O(\varepsilon))\delta$ from any query curve that has $E$ as a coarse encoding, we will discuss shortly how to obtain the curves in $T_E$.

Each key $E$ in $\mathcal{D}$ has $O(k)$ size because $E$ stores $O(k)$ cells in $\mathcal{G}_1$ and $\mathcal{G}_2$. As a trie, $\mathcal{D}$ is a rooted tree with as many levels as the length of the key $E$. Searching in $D$ boils down to visiting the appropriate child of an internal node of $\mathcal{D}$. Each component of the key $E$ is an element of $\mathcal{G}_2 \cup \{\text{null}\}$ or $(\mathcal{G}_1 \times \mathcal{G}_1) \cup \{\text{null}\}$; there are $O(m^2n^2/\varepsilon^{2d})$ possibilities. We keep a dictionary at each internal node of $\mathcal{D}$ for finding the appropriate child to visit in $O(\log\frac{mn}{\varepsilon})$ time. Hence, the total search time of $\mathcal{D}$ is $O(k\log\frac{mn}{\varepsilon})$.

To bound the size of $\mathcal{D}$, observe that each key $E$ at a leaf of $\mathcal{D}$ induces $O(k)$ entries in the dictionaries at the ancestors of that leaf. There are $O(\sqrt{d}/\varepsilon)^{4d(k-1)}(mn)^{4(k-1)}$ candidate coarse encodings. So the total space taken by the dictionaries at the internal nodes is $O(\sqrt{d}/\varepsilon)^{4d(k-1)}(mn)^{4(k-1)}k$. We will show that if a query curve has $E$ as a coarse encoding, any curve in $T_E$ is within a Fréchet distance of $(1+O(\varepsilon))\delta$ from that query curve. Therefore, we only need to store one of the curves in $T_E$ at the leaf for $E$, and it suffices to store the index of this curve. Therefore, the total space complexity of $\mathcal{D}$ is $O(\sqrt{d}/\varepsilon)^{4d(k-1)}(mn)^{4(k-1)}k$.

The construction of $\mathcal{D}$ proceeds as follows. We initialize $\mathcal{D}$ to be empty. We enumerate all possible candidate coarse encodings based on constraints 1(a), 2(a), 3(a), and 3(b). Take a possible candidate coarse encoding $E$. The set $T_E$ is initialized to be empty. We go through every input curve $\tau_i$ to determine whether to include $\tau_i$ in $T_E$. If $T_E \neq \emptyset$ in the end, we insert $E$ together with one curve in $T_E$ into $\mathcal{D}$. In the following, we discuss the checking of whether to include $\tau_i$ in $T_E$.

Let $E$ be $(\mathcal{A}, \mathcal{B}, \mathcal{C})$. We generate all possible *partitions* of $\{v_{i,1}, \ldots, v_{i,m}\}$ that satisfy the following properties.

**Partition:** a sequence of $k$ *disjoint* subsets $(\pi_0, \pi_1, \ldots, \pi_{k-1})$ such that $\bigcup_{j=0}^{k-1}\pi_j = \{v_{i,1}, \ldots, v_{i,m}\}$, $v_{i,1} \in \pi_0$, $v_{i,m} \in \pi_{k-1}$, $\pi_j$ may be empty for some $j \in [k-2]$, and for any $v_{i,a} \in \pi_j$ and any $v_{i,b} \in \pi_l$, if $j < l$, then $a < b$.

There are fewer than $m^{k-1}$ partitions. Given a partition $(\pi_0, \ldots, \pi_{k-1})$, the vertices in $\pi_0$ are to be matched with $v_{i,1}$; for $j \in [k-1]$, the vertices in $\pi_j$ are to be matched with points in $w_j w_{j+1} \setminus \{w_j\}$, where $w_j w_{j+1}$ is the $j$-th edge of the query curve; $v_{i,m}$ and possibly other vertices of $\tau_i$ are matched with $w_k$. The reference to $w_j w_{j+1}$ is conceptual; we do not need to know the query curve in preprocessing.

We describe four tests for each partition below. As soon as we come across a partition that passes all four tests, we insert $\tau_i$ into $T_E$. If a partition fails any test, we move on to the next partition. If no partition can pass all four tests in the end, we do not include $\tau_i$ in $T_E$.

The first test is that for $j \in [2, k-1]$, $\pi_j = \emptyset$ if and only if $(c_{j,1}, c_{j,2}) = \text{null}$. This test takes $O(k)$ time. We exclude $\pi_1$ from this test because $(c_{1,1}, c_{1,2}) \neq \text{null}$ by constraint 1(a), whereas $\pi_1$ may be empty or not depending on the partition enumerated.

In the second test, for $j \in [k-1]$, if $\pi_j \neq \emptyset$, let $a_j, b_j \in [m]$ be the smallest and largest indices such that $v_{i,a_j}, v_{i,b_j} \in \pi_j$, the intuition is that $v_{i,a_j}, \ldots, v_{i,b_j}$ can be matched to the surrogate $(c_{j,1}, c_{j,2})$ of $w_j w_{j+1}$ within a distance of $(1 + O(\varepsilon))\delta$. The second test checks this property as follows. Observe that $(c_{j,1}, c_{j,2}) \neq \text{null}$: $(c_{1,1}, c_{1,2}) \neq \text{null}$ by constraint 1(a), and for $j \in [2, k-1]$, $(c_{j,1}, c_{j,2}) \neq \text{null}$ by the first test. Pick the smallest vertices $x_j$ of $c_{j,1}$ and $y_j$ of $c_{j,2}$ according to the lexicographical order of their coordinates. If $x_j y_j \cap (v_{i,a_j} \oplus B_{(1+12\varepsilon)\delta})$ or $x_j y_j \cap (v_{i,b_j} \oplus B_{(1+12\varepsilon)\delta})$ is empty, the test fails. Otherwise, compute the minimum point $x'_j$ in $x_j y_j \cap (v_{i,a_j} \oplus B_{(1+12\varepsilon)\delta})$ and the maximum point $y'_j$ in $x_j y_j \cap (v_{i,b_j} \oplus B_{(1+12\varepsilon)\delta})$ with respect to $\leq_{x_j y_j}$. If it is not the case that $x'_j \leq_{x_j y_j} y'_j$, the test fails. Suppose that $x'_j \leq_{x_j y_j} y'_j$. Compute $d_F(x'_j y'_j, \tau_i[v_{i,a_j}, v_{i,b_j}])$ and check whether it is $(1 + 12\varepsilon)\delta$ or less. If all of the above checks succeed for all $j \in [k-1]$, the second test succeeds; otherwise, the test fails. The test takes $O(m \log m)$ time, which is dominated by the computation of $d_F(x'_j y'_j, \tau_i[v_{i,a_j}, v_{i,b_j}])$ over all $j \in [k-1]$.

The third test is that $\mathcal{B}[1] \in G(v_{i,1} \oplus B_\delta)$ and $\mathcal{A}[k-1] \in G(v_{i,m} \oplus B_\delta)$, which boils down to checking whether $d(v_{i,1}, \mathcal{B}[1])$ and $d(v_{i,m}, \mathcal{A}[k-1])$ are at most $\delta$.

The fourth test involves $\mathcal{J}$, the set of $(r, s) \in [k-1] \times [k-1]$ such that $r < s$, $(c_{r,1}, c_{r,2}) \neq \text{null}$, $(c_{s,1}, c_{s,2}) \neq \text{null}$, and $(c_{j,1}, c_{j,2}) = \text{null}$ for $j \in [r+1, s-1]$. Note that $|\mathcal{J}| \leq k-1$ and it can be constructed in $O(k)$ time. For every $(r, s) \in \mathcal{J}$, if $r = 1$ and $\pi_1 = \emptyset$, let $b_1 = 1$; otherwise, let $b_r = \max\{b : v_{i,b} \in \pi_r\}$. It follows that $b_r + 1 = \min\{a : v_{i,a} \in \pi_s\}$. We check if it is the case that $\tau_{i,b_r} \cap \mathcal{A}[r] \neq \emptyset$, $\tau_{i,b_r} \cap \mathcal{B}[s] \neq \emptyset$, and we hit $\mathcal{A}[r]$ no later than $\mathcal{B}[s]$ when we walk from $v_{i,b_r}$ to $v_{i,b_r+1}$. (Recall the intuition that the pair $\mathcal{A}[r]$ and $\mathcal{B}[s]$ serve as the surrogate of the edge $\tau_{i,b_r} = v_{i,b_r} v_{i,b_r+1}$.) If check fails for any $(r, s) \in \mathcal{J}$, the test fails. Otherwise, the test succeeds. This test runs in $O(k)$ time.

The following result summarizes the construction of $\mathcal{D}$ and four properties of each candidate coarse encoding in $\mathcal{D}$.

▶ **Lemma 3.** *The trie $\mathcal{D}$ has $O(\sqrt{d}/\varepsilon)^{4d(k-1)}(mn)^{4(k-1)}k$ size and can be constructed in $O(\sqrt{d}/\varepsilon)^{4d(k-1)}(mn)^{4(k-1)}(k \log \frac{mn}{\varepsilon} + m^k \log m)$ time. We can search $\mathcal{D}$ with a coarse encoding in $O(k \log \frac{mn}{\varepsilon})$ time. For each candidate coarse encoding $E = (\mathcal{A}, \mathcal{B}, \mathcal{C})$, a curve $\tau_i \in T$ belongs to $T_E$ if and only if there exists a partition $(\pi_0, \ldots, \pi_{k-1})$ of the vertices of $\tau_i$ that satisfy the following four properties. For $j \in [k-1]$, if $j = 1$ and $\pi_1 = \emptyset$, let $b_1 = 1$; otherwise, if $\pi_j \neq \emptyset$, let $a_j = \min\{a : v_{i,a} \in \pi_j\}$ and let $b_j = \max\{b : v_{i,b} \in \pi_j\}$.*

**(i)** *For $j \in [2, k-1]$, $\pi_j = \emptyset$ if and only if $(c_{j,1}, c_{j,2}) = \text{null}$.*

**(ii)** *For $j \in [k-1]$, if $\pi_j \neq \emptyset$, let $x_j$ and $y_j$ be the smallest vertices of $c_{j,1}$ and $c_{j,2}$ according to the lexicographical order of their coordinates, there exist $x''_j, y''_j \in x_j y_j$ such that $x''_j \leq_{x_j y_j} y''_j$ and $d_F(x''_j y''_j, \tau_i[v_{i,a_j}, v_{i,b_j}]) \leq (1 + 12\varepsilon)\delta$.*

**(iii)** *$\mathcal{B}[1] \in G(v_{i,1} \oplus B_\delta)$ and $\mathcal{A}[k-1] \in G(v_{i,m} \oplus B_\delta)$.*

**(iv)** *For every $(r, s) \in \mathcal{J}$, $\tau_{i,b_r} \cap \mathcal{A}[r] \neq \emptyset$, $\tau_{i,b_r} \cap \mathcal{B}[s] \neq \emptyset$, and we hit $\mathcal{A}[r]$ no later than $\mathcal{B}[s]$ when we walk from $v_{i,b_r}$ to $v_{i,b_r+1}$.*

## 2.3 Querying

At query time, we are given a curve $\sigma = (w_1, ..., w_k)$. We enumerate all coarse encodings of $\sigma$; for each coarse encoding $E$ enumerated, we search the trie $\mathcal{D}$ for $E$; if $E$ is found, we return the curve in $T_E$ stored with $E$ as the answer of the query; if no coarse encoding of $\sigma$ can be found in $\mathcal{D}$, we return "no".

Each search in $\mathcal{D}$ takes $O(k \log \frac{mn}{\varepsilon})$ time as stated in Lemma 3. The enumeration of the coarse encodings of $\sigma$ require a solution for the $(11\varepsilon\delta)$-segment queries on $\mathcal{G}_1$ as stated in constraint 1(b)(i) in Section 2.1. We will discuss an efficient solution later.

For $j \in [k-1]$, we make two $(11\varepsilon\delta)$-segment queries with $w_jw_{j+1}$ and $w_{j+1}w_j$ on $\mathcal{G}_1$ to obtain $u_{j,1}$ and $u_{j,2}$, respectively. If any of the two queries returns null, define $(u_{j,1}, u_{j,2})$ to be null. If $(u_{j,1}, u_{j,2}) \neq$ null and the minimum point in $w_jw_{j+1} \cap (u_{j,1} \oplus B_{11\varepsilon\delta})$ does not lie in front of the maximum point in $w_jw_{j+1} \cap (u_{j,2} \oplus B_{11\varepsilon\delta})$ with respect to $\leq_{w_jw_{j+1}}$, then constraint 1(b)(ii) is not satisfied. It must be the case that $w_jw_{j+1}$ does not intersect the interior of the union of cells in $\mathcal{G}_1$, and the $(11\varepsilon\delta)$-segment queries just happen to return two cells that violate constraint 1(b)(ii). In this case, the input vertices are too far from $w_jw_{j+1}$ to be matched to any point in $w_jw_{j+1}$ within a distance $\delta$, so we reset $(u_{j,1}, u_{j,2})$ to be null.

After defining $(u_{j,1}, u_{j,2})$ for $j \in [k-1]$, we generate the coarse encodings of $\sigma$ as follows. The pairs $(c_{1,1}, c_{1,2})$ and $(c_{k-1,1}, c_{k-1,2})$ are defined to be $(u_{1,1}, u_{1,2})$ and $(u_{k-1,1}, u_{k-1,2})$, respectively. For $j \in [2, k-2]$, we enumerate all possible $\mathcal{C}$ by setting $(c_{j,1}, c_{j,2})$ to be $(u_{j,1}, u_{j,2})$ or null. This gives a total of $2^{k-3}$ possible $\mathcal{C}$'s. We query the point location data structure for $\mathcal{G}_2$ to find the cells $\mathcal{B}[1]$ and $\mathcal{A}[k-1]$ that contain $w_1$ and $w_k$, respectively. Then, for each $\mathcal{C}$ enumerated, we enumerate $\mathcal{A}[j]$ for $j \in [1, k-2]$ and $\mathcal{B}[j]$ for $j \in [2, k-1]$ according to constraints 3(a) and 3(b) in Section 2.1. This enumeration produces $O(\sqrt{d}/\varepsilon)^{2d(k-2)}$ tuples of $(\mathcal{A}, \mathcal{B}, \mathcal{C})$. For each $(\mathcal{A}, \mathcal{B}, \mathcal{C})$ enumerated, we check whether it satisfies constraint 3(c), which can be done in $O(k \log k)$ time as implied by the following result.

▶ **Lemma 4.** *Take any $(r, s) \in \mathcal{J}$. Let $x_r$ and $x_s$ be the smallest vertices of $\mathcal{A}[r]$ and $\mathcal{B}[s]$ by the lexicographical order of their coordinates. We can check in $O((s-r)\log(s-r))$ time whether there are $x'_r, x'_s \in x_rx_s$ such that $x'_r \leq_{x_rx_s} x'_s$ and $d_F(x'_rx'_s, \sigma[w_{r+1}, w_s]) \leq (1+\varepsilon)\delta$.*

▶ **Lemma 5.** *The query time is $O(kQ_{\mathrm{seg}}) + O(\sqrt{d}/\varepsilon)^{2d(k-2)} k \log \frac{mn}{\varepsilon}$, where $Q_{\mathrm{seg}}$ is the time to answer an $(11\varepsilon\delta)$-segment query.*

## 2.4 Approximation guarantee

First, we show that if $\sigma$ is within a Fréchet distance $\delta$ from some input curve, there exists a coarse encoding $E$ of $\sigma$ such that $T_E \neq \emptyset$. Hence, $E$ and a curve in $T_E$ are stored in $\mathcal{D}$.

▶ **Lemma 6.** *If $d_F(\tau_i, \sigma) \leq \delta$, then $\tau_i \in T_E$ for some coarse encoding $E$ of $\sigma$.*

**Proof.** Let $\mathcal{M}$ be a Fréchet matching between $\tau_i$ and $\sigma$. Let $E$ be the coarse encoding specified for $\sigma$ in Lemma 2. For any subcurve $\sigma' \subseteq \sigma$, we use $\mathcal{M}(\sigma')$ to denote the subcurve of $\tau_i$ matched to $\sigma'$ by $\mathcal{M}$. For $j \in [k-1]$, let $\tilde{\pi}_j = \{v_{i,a} : a \in [m-1], v_{i,a} \in \mathcal{M}(w_jw_{j+1}) \setminus \mathcal{M}(w_j)\}$. Define $\pi_j = \tilde{\pi}_j$ for $j \in [k-2]$, $\pi_{k-1} = \{v_{i,m}\} \cup \tilde{\pi}_{k-1}$, and $\pi_0 = \{v_{i,1}, \ldots, v_{i,m}\} \setminus \bigcup_{j=1}^{k-1} \pi_j$. We obtain a partition $(\pi_0, ..., \pi_{k-1})$ of the vertices of $\tau_i$.

We prove that $E$, $\tau_i$, and $(\pi_0, ..., \pi_{k-1})$ satisfy Lemma 3(i)–(iv) which put $\tau_i$ in $T_E$. Lemma 3(i) follows directly from Lemma 2(i),

Take any $j \in [k-1]$ such that $\pi_j \neq \emptyset$. Let $\pi_j$ be $\{v_{i,a}, v_{i,a+1}, \ldots, v_{i,b}\}$. By the definition of $\pi_j$, every vertex in $\pi_j$ belongs to $\mathcal{M}(w_jw_{j+1})$, so $\tau_i[v_{i,a}, v_{i,b}] \subset \mathcal{M}(w_jw_{j+1})$. Then, there must exist two points $p, q \in w_jw_{j+1}$ such that $p \leq_{w_jw_{j+1}} q$ and $d_F(pq, \tau_i[v_{i,a}, v_{i,b}]) \leq \delta$. If $j = 1$, we have $(c_{1,1}, c_{1,2}) \neq$ null by constraint 1(a); if $j \in [2, k-1]$, by Lemma 2(i), $(c_{j,1}, c_{j,2}) \neq$ null as $\pi_j \neq$ null. Therefore, $c_{j,1}$ and $c_{j,2}$ are cells in $\mathcal{G}_1$ returned by the $(11\varepsilon\delta)$-segment queries with $w_j w_{j+1}$ and $w_{j+1} w_j$, respectively. We have shown that $d_F(pq, \tau_i[v_{i,a}, v_{i,b}]) \leq \delta$; therefore, $p$ is contained in a cell in $G(v_{i,a} \oplus B_\delta) \subset \mathcal{G}_1$. As $c_{j,1}$ is the cell returned by the $(11\varepsilon\delta)$-segment query with $w_j w_{j+1}$, there must be a point $z_p \in w_j w_{j+1} \cap (c_{j,1} \oplus B_{11\varepsilon\delta})$ such that $z_p \leq_{w_j w_{j+1}} p$. In a similar way, we can conclude that there must be a point $z_q \in w_j w_{j+1} \cap (c_{j,2} \oplus B_{11\varepsilon\delta})$ such that $q \leq_{w_j w_{j+1}} z_q$. That is, $z_p \leq_{w_j w_{j+1}} p \leq_{w_j w_{j+1}} q \leq_{w_j w_{j+1}} z_q$. Let $x_j$ and $y_j$ be the smallest vertices of $c_{j,1}$ and $c_{j,2}$ according to the lexicographical order of their coordinates. Both $d(z_p, x_j)$ and $d(z_q, y_j)$ are at most $12\varepsilon\delta$. A linear interpolation from $z_p z_q$ to $x_j y_j$ maps $p$ and $q$ to two points $x''_j$ and $y''_j$ on $x_j y_j$, respectively, such that $x''_j \leq_{x_j y_j} y''_j$. Also, the linear interpolation adds a distance $12\varepsilon\delta$ or less, which gives $d_F(x''_j y''_j, \tau_i[v_{i,a}, v_{i,b}]) \leq d_F(x''_j y''_j, pq) + d_F(pq, \tau_i[v_{i,a}, v_{i,b}]) \leq (1 + 12\varepsilon)\delta$. Hence, Lemma 3(ii) is satisfied.

The grid cells $\mathcal{B}[1]$ and $\mathcal{A}[k-1]$ are defined to contain $w_1$ and $w_k$, respectively. Also, $v_{i,1} \in \mathcal{M}(w_1)$ and $v_{i,m} \in \mathcal{M}(w_k)$. Hence, $\mathcal{B}[1] \in G(v_{i,1} \oplus B_\delta)$ and $\mathcal{A}[k-1] \in G(v_{i,m} \oplus B_\delta)$, satisfying Lemma 3(iii).

For any pair $(r, s) \in \mathcal{J}$, by Lemma 2(ii), there exist two points $z \in \mathcal{A}[r] \cap \tau_{i,b_r}$ and $z' \in \mathcal{B}[s] \cap \tau_{i,b_r}$ such that $z \leq_{\tau_{i,b_r}} z'$. Since $\mathcal{A}[r]$ and $\mathcal{B}[s]$ are interior-disjoint unless they are equal, we must hit $\mathcal{A}[r]$ no later than $\mathcal{B}[s]$ when we walk from $v_{i,b_r}$ to $v_{i,b_r+1}$, satisfying Lemma 3(iv). ◀

We show that if $E$ is a coarse encoding of $\sigma$, each curve in $T_E$ is close to $\sigma$.

▶ **Lemma 7.** *Let $E$ be a coarse encoding of $\sigma$. For every $\tau_i \in T_E$, $d_F(\tau_i, \sigma) \leq (1 + 24\varepsilon)\delta$.*

**Proof.** (Sketch) Suppose that $T_E \neq \emptyset$ as the lemma statement is vacuous otherwise. Take any $\tau_i \in T_E$. We construct a matching $\mathcal{M}$ between $\tau_i$ and $\sigma$ such that $d_{\mathcal{M}}(\tau_i, \sigma) \leq (1+24\varepsilon)\delta$. Since $T_E \neq \emptyset$, there exists a partition $(\pi_0, \ldots, \pi_{k-1})$ of the vertices of $\tau_i$ that satisfy Lemma 3(i)–(iv). Using these properties, we can match the vertices of $\tau_i$ to points on $\sigma$ and then the vertices of $\sigma$ to points on $\tau_i$. Afterwards, $\sigma$ and $\tau_i$ divided into line segments by their vertices and images of their matching partners. We use linear interpolations to match the corresponding line segments. More details can be found in the appendix. ◀

In two and three dimensions, the ray shooting data structure for boxes in [8] can be used as the $(11\varepsilon\delta)$-segment query data structure. It has an $O(\log |\mathcal{G}_1|) = O(\log \frac{mn}{\varepsilon})$ query time and an $O(|\mathcal{G}_1|^{2+\mu}) = O((mn)^{2+\mu}/\varepsilon^{d(2+\mu)})$ space and preprocessing time for any fixed $\mu \in (0, 1)$. If the query segment does not intersect any cell in $\mathcal{G}_1$, we return null. In four and higher dimensions, we will prove the following result in Section 4.

▶ **Lemma 8.** *We can construct a data structure in $O(\sqrt{d}/\varepsilon)^{O(d/\varepsilon^2)} \cdot \tilde{O}((mn)^{O(1/\varepsilon^2)})$ space and preprocessing time such that given any oriented edge $e$ of the query curve $\sigma$, the data structure either discovers that $\min_{\tau_i \in T} d_F(\sigma, \tau_i) > \delta$, or reports a correct answer for the $(11\varepsilon\delta)$-segment query with $e$ on $\mathcal{G}_1$. The query time is $\tilde{O}((mn)^{0.5+\varepsilon}/\varepsilon^d)$.*

Combining the results in this section with the ray shooting result in [8], Lemma 8, and Theorem 1 gives $(1+\varepsilon)$-ANN data structures. Theorem 1 uses the deletion cost of a $(1+\varepsilon, \delta)$-ANN data structure. We perform each deletion by reconstructing the data structure from scratch because we do not have a more efficient solution.

▶ **Theorem 9.** *For any $\varepsilon \in (0, 0.5)$, there is a $(1 + O(\varepsilon))$-ANN data structure for $T$ under the Fréchet distance with the following performance guarantees:*

- $d \in \{2,3\}$ :
  *query time* $= O\big(\frac{1}{\varepsilon}\big)^{2d(k-2)} k \log \frac{mn}{\varepsilon} \log n$,
  *space* $= O\big(\frac{1}{\varepsilon}\big)^{4d(k-1)+1} (mn)^{4(k-1)} k \log^2 n$,
  *expected preprocessing time* $= O\big(\frac{1}{\varepsilon}\big)^{4d(k-1)} (mn)^{4(k-1)} \big(k \log \frac{mn}{\varepsilon} + m^k \log m\big) n \log n$.
- $d \geq 4$ :
  *query time* $= \tilde{O}\big(\frac{1}{\varepsilon^d} k (mn)^{0.5+\varepsilon}\big) + O\big(\frac{\sqrt{d}}{\varepsilon}\big)^{2d(k-2)} k \log \frac{mn}{\varepsilon} \log n$,
  *space* $= O\big(\frac{\sqrt{d}}{\varepsilon}\big)^{4d(k-1)} (mn)^{4(k-1)} k \cdot \frac{1}{\varepsilon} \log^2 n + O\big(\frac{\sqrt{d}}{\varepsilon}\big)^{O(d/\varepsilon^2)} \cdot \tilde{O}((mn)^{O(1/\varepsilon^2)})$,
  *expected preprocessing time* $= O\big(\frac{\sqrt{d}}{\varepsilon}\big)^{4d(k-1)} (mn)^{4(k-1)} \big(k \log \frac{mn}{\varepsilon} + m^k \log m\big) n \log n +$
  $O\big(\frac{\sqrt{d}}{\varepsilon}\big)^{O(d/\varepsilon^2)} \cdot \tilde{O}((mn)^{O(1/\varepsilon^2)})$.

## 3 $(3 + O(\varepsilon), \delta)$-ANN

Given a query curve $\sigma = (w_1, w_2, \ldots, w_k)$, for $j \in [k-1]$, we solve the $(11\varepsilon\delta)$-segment queries with $w_j w_{j+1}$ and $w_{j+1} w_j$ on $\mathcal{G}_1$ as before. Let $((c_{j,1}, c_{j,2}))_{j\in[k-1]}$ denote the results of the queries. Recall that each $(c_{j,1}, c_{j,2})$ belongs to $(\mathcal{G}_1 \times \mathcal{G}_1) \cup \{\text{null}\}$.

Suppose that there are $k_0 \leq k-1$ non-null pairs in $((c_{j,1}, c_{j,2}))_{j\in[k-1]}$. Extract these non-null pairs to form the sequence $((c_{j_r,1}, c_{j_r,2}))_{r\in[k_0]}$. Note that $j_1 = 1$ and $j_{k_0} = k-1$ by constraint 1(a). We construct a polygonal curve $\sigma_0$ by connecting the centers of $c_{j_r,1}$ and $c_{j_r,2}$ for $r \in [k_0]$ and the centers of $c_{j_r,2}$ and $c_{j_{r+1},1}$ for $r \in [k_0 - 1]$. The polygonal curve $\sigma_0$ acts as a surrogate of $\sigma$. It has at most $2k-2$ vertices. We will use $\sigma_0$ as the key to search a trie at query time to obtain an answer for a $(3 + O(\varepsilon), \delta)$-ANN query. As a result, no enumeration is needed which avoids the exponential dependence of the query time on $k$.

In preprocessing, we enumerate all sequences of $2l$ cells in $\mathcal{G}_1$ for $l \in [2, k-1]$. For each sequence, we construct the polygonal curve $\sigma'$ that connects the centers of the cells in the sequence, and we find the nearest input curve $\tau_i$ to $\sigma'$. If $d_F(\sigma', \tau_i) \leq (1 + 12\varepsilon)\delta$, we store $(\sigma', i)$ in a trie $\mathcal{D}$. There are $O(\sqrt{d}/\varepsilon)^{2d(k-1)}(mn)^{2(k-1)}$ entries in $\mathcal{D}$. We organize the trie $\mathcal{D}$ in the same way as described in Section 2.2. The space required by $\mathcal{D}$ is $O(\sqrt{d}/\varepsilon)^{2d(k-1)}(mn)^{2(k-1)}k$. The search time of $\mathcal{D}$ is $O(k \log \frac{mn}{\varepsilon})$. The preprocessing time is $O(\sqrt{d}/\varepsilon)^{2d(k-1)}(mn)^{2(k-1)}(k \log \frac{mn}{\varepsilon} + kmn \log(km)) = O(\sqrt{d}/\varepsilon)^{2d(k-1)}(mn)^{2k-1} k \log \frac{mn}{\varepsilon}$ due to the computation of the nearest input curve for each sequence enumerated.

At query time, we construct $\sigma_0$ from $\sigma$ in $O(kQ_{\text{seg}})$ time, where $Q_{\text{seg}}$ is the time to answer a $(11\varepsilon\delta)$-segment query. We compute $d_F(\sigma, \sigma_0)$ in $O(k^2 \log k)$ time. If $d_F(\sigma, \sigma_0) > (2 + 12\varepsilon)\delta$, we report "no". Otherwise, we search $\mathcal{D}$ with $\sigma_0$ in $O(k \log \frac{mn}{\varepsilon})$ time. If the search fails, we report "no". Otherwise, the search returns $(\sigma_0, i)$ for some $i \in [n]$.

▶ **Lemma 10.** *If $d_F(\sigma, \sigma_0) \leq (2 + 12\varepsilon)\delta$ and the search in $\mathcal{D}$ with $\sigma_0$ returns $(\sigma_0, i)$, then $d_F(\sigma, \tau_i) \leq (3 + 24\varepsilon)\delta$. Otherwise, $\min_{\tau_i \in T} d_F(\sigma, \tau_i) > \delta$.*

Combining the results in this section with the ray shooting results in two and three dimensions [8], Lemma 8, and Theorem 1, we obtain the following theorem.

▶ **Theorem 11.** *For any $\varepsilon \in (0, 0.5)$, there is a $(3 + O(\varepsilon))$-ANN data structure for $T$ under the Fréchet distance with the following performance guarantees:*

- $d \in \{2,3\}$*:*
  *query time* $= O(k \log \frac{mn}{\varepsilon} \log n)$,
  *space* $= O\big(\frac{1}{\varepsilon}\big)^{2d(k-1)+1} (mn)^{2(k-1)} k \log^2 n$,
  *expected preprocessing time* $= O\big(\frac{1}{\varepsilon}\big)^{2d(k-1)} (mn)^{2k-1} kn \log \frac{mn}{\varepsilon} \log n$.

- $d \geq 4:$
  *query time* $= \tilde{O}\big(\frac{1}{\varepsilon^d} k(mn)^{0.5+\varepsilon}\big)$,
  *space* $= O\big(\frac{\sqrt{d}}{\varepsilon}\big)^{2d(k-1)}(mn)^{2(k-1)}k \cdot \frac{1}{\varepsilon}\log^2 n + O\big(\frac{\sqrt{d}}{\varepsilon}\big)^{O(d/\varepsilon^2)} \cdot \tilde{O}((mn)^{O(1/\varepsilon^2)})$,
  *expected preprocessing time* $= O\big(\frac{\sqrt{d}}{\varepsilon}\big)^{2d(k-1)}(mn)^{2k-1}kn\log\frac{mn}{\varepsilon}\log n +$
  $O\big(\frac{\sqrt{d}}{\varepsilon}\big)^{O(d/\varepsilon^2)} \cdot \tilde{O}((mn)^{O(1/\varepsilon^2)})$.

## 4 $(11\varepsilon\delta)$-segment queries and proof of Lemma 8

We describe the $(11\varepsilon\delta)$-segment query data structure in Lemma 8. We first present the main ideas before giving the details. Let $w_j w_{j+1}$ be a query segment, which is unknown at preprocessing. There are three building blocks.

First, the intuition is to capture the support lines of all possible query segments using pairs of cells in $\mathcal{G}_1$. It would be ideal to retrieve a pair of cells intersected by $\mathrm{aff}(w_j w_{j+1})$, but this seems to be as difficult as the ray shooting problem. For a technical reason, we need to use more grid cells in a larger neighborhood of the input vertices than in $\mathcal{G}_1$, so define $\mathcal{G}_3 = \bigcup_{i\in[n],a\in[m]} G(v_{i,a} \oplus B_{(1+6\varepsilon)\delta})$.

We find a grid vertex $x$ of $\mathcal{G}_1$ that is a $(1+\varepsilon)$-approximate nearest grid vertex to $\mathrm{aff}(w_j w_{j+1})$. We will show that if $d(x, \mathrm{aff}(w_j w_{j+1})) > (1+\varepsilon)\varepsilon\delta$, the answer to the $(11\varepsilon\delta)$-segment query is null; otherwise, we can find a cell $\gamma \in \mathcal{G}_3$ near $x$ that intersects $\mathrm{aff}(w_j w_{j+1})$. We can use any other cell $c \in \mathcal{G}_1$ to form a pair with $\gamma$ that acts as a surrogate for the support lines of query segments that pass near $c$ and $\gamma$.

Second, given $w_j w_{j+1}$ at query time, among all possible choices of $c$, we need to find the right one(s) efficiently so that $(c, \gamma)$ is a surrogate for $\mathrm{aff}(w_j w_{j+1})$. We explain the ideas using the case that $w_{j+1}$ lies between $w_j$ and $\mathrm{aff}(w_j w_{j+1}) \cap \gamma$. Note that $w_j$ may not be near any cell in $\mathcal{G}_1$. In order that $\min_{\tau_i \in T} d(\sigma, \tau_i) \leq \delta$, $w_j$ must be within a distance $\delta$ from some input edge $\tau_{i,a}$. We find a maximal packing of $\mathrm{aff}(\tau_{i,a}) \oplus B_{O(\delta)}$ using lines that are parallel to $\tau_{i,a}$ and are at distance $\Theta(\varepsilon\delta)$ or more apart. There are $O(\varepsilon^{1-d})$ lines in the packing, and every point in $\mathrm{aff}(\tau_{i,a}) \oplus B_\delta$ is within a distance $O(\varepsilon\delta)$ from some line in the packing. The projection of $w_j$ to the approximately nearest line approximates the location of $w_j$. Hence, we should seek to divide the lines in the packing into appropriate segments so that, given $w_j$ and its approximately nearest line in the packing, we can efficiently find the segment that contains the projection of $w_j$ and retrieve some precomputed information for that segment.

Third, let $\ell$ be a line in the packing mentioned above, for each possible cell $c \in \mathcal{G}_1$, we use the geometric construct $F(c, \gamma) = \{x \in \mathbb{R}^d : \exists\, y \in \gamma \,\text{s.t.}\; xy \cap c \neq \emptyset\}$ defined in [6] which can be computed in $O(1)$ time. The projection of $(\ell \oplus B_{2\varepsilon\delta}) \cap F(c, \gamma)$ in $\ell$ is the set of points on $\ell$ such that if the projection of $w_j$ is in it, then $(c, \gamma)$ is a surrogate for $\mathrm{aff}(w_j w_{j+1})$. As a result, the endpoints of the projections of $(\ell \oplus B_{2\varepsilon\delta}) \cap F(c, \gamma)$ over all possible choices of $c$ divide $\ell$ into segments that we desire. Each segment may stand for several choices of $c$'s. For each segment, we store the cell $c'$ close to that segment because the ideal choice is the cell that we hit first as we walk from $w_j$ to $w_{j+1}$.

As described above, we use two approximate nearest neighbor data structures that involve lines. The first one is due to Andoni et al. [4] which stores a set of points $P$ such that given a query line, the $(1+\varepsilon)$-approximate nearest point to the query line can be returned in $\tilde{O}(d^3|P|^{0.5+\varepsilon})$ time. It uses $\tilde{O}\big(d^2|P|^{O(1/\varepsilon^2)}\big)$ space and preprocessing time. The second result is due to Agarwal et al. [2] which stores a set $L$ of lines such that given a query point, the 2-approximate nearest line to the query point can be returned in $\tilde{O}(1)$ time. It uses $\tilde{O}(|L|^2)$ space and expected preprocessing time.

## 4.1 Data structure organization

We restrict $\varepsilon$ to be chosen from $(0, 0.5)$. We construct the data structure of Andoni et al. [4] for the grid vertices of $\mathcal{G}_1$ so that for any query line, the $(1+\varepsilon)$-approximate nearest grid vertex can be returned in $\tilde{O}((mn)^{0.5+\varepsilon}/\varepsilon^d)$ time. We denote this data structure by $D_{\text{anp}}$. It takes $O(\sqrt{d}/\varepsilon)^{O(d/\varepsilon^2)} \cdot \tilde{O}((mn)^{O(1/\varepsilon^2)})$ space and preprocessing time.

For each input edge $\tau_{i,a}$, define a set of lines $L_{i,a}$ as follows. Let $H$ be the hyperplane through $v_{i,a}$ orthogonal to $\text{aff}(\tau_{i,a})$. Take a $(d-1)$-dimensional grid in $H$ with $v_{i,a}$ as a grid vertex and cell width $\varepsilon\delta/\sqrt{d-1}$. The set $L_{i,a}$ includes every line that is orthogonal to $H$ and passes through a vertex of this grid in $H$ at distance within $(1+2\varepsilon)\delta$ from $v_{i,a}$. The set $L_{i,a}$ has $O(\varepsilon^{1-d})$ size, and it can be constructed in $O(\varepsilon^{1-d})$ time. Moreover, every point in the cylinder $\text{aff}(\tau_{i,a}) \oplus B_\delta$ is within a distance $\varepsilon\delta$ from some line in $L_{i,a}$.

Define $\mathcal{L} = \bigcup_{i\in[n],a\in[m-1]} L_{i,a}$. The size of $\mathcal{L}$ is $O(mn/\varepsilon^{d-1})$, and $\mathcal{L}$ can be constructed in $O(mn/\varepsilon^{d-1})$ time. We construct the data structure of Agarwal et al. [2] for $\mathcal{L}$ so that for any query point, a 2-approximate nearest line in $\mathcal{L}$ can be returned in $\tilde{O}(1)$ time. We denote this data structure by $D_{\text{anl}}$. It uses $\tilde{O}((mn)^2/\varepsilon^{2d-2})$ space and expected preprocessing time.

Recall that $\mathcal{G}_3 = \bigcup_{i\in[n],a\in[m]} G(v_{i,a} \oplus B_{(1+6\varepsilon)\delta})$.

For every $\gamma \in \mathcal{G}_3$ and every $c \in \mathcal{G}_1$, we construct $F(c,\gamma) = \{x \in \mathbb{R}^d : \exists\, y \in \gamma \text{ s.t. } xy \cap c \neq \emptyset\}$, which is empty or an unbounded convex polytope of $O(1)$ size that can be constructed in $O(1)$ time as a Minkowski sum [6]. The total time needed is $O((mn)^2/\varepsilon^{2d})$.

For every $\gamma \in \mathcal{G}_3$, every $c \in \mathcal{G}_1$, and every line $\ell \in \mathcal{L}$, compute the intersection $(\ell \oplus B_{2\varepsilon\delta}) \cap F(c,\gamma)$ and project it orthogonally to a segment in $\ell$. Take any line $\ell \in \mathcal{L}$. The resulting segment endpoints in $\ell$ divide $\ell$ into *canonical segments*. There are $O((mn)^2/\varepsilon^{2d})$ canonical segments in $\ell$. For every cell $\gamma \in \mathcal{G}_3$ and every canonical segment $\xi \subseteq \ell$, compute the set $C_{\gamma,\xi}$ of every cell $c \in \mathcal{G}_1$ such that $\xi$ is contained in the projection of $(\ell \oplus B_{2\varepsilon\delta}) \cap F(c,\gamma)$ onto $\ell$. Fix an arbitrary point in $\xi$ and denote it by $p_\xi$. Each $C_{\gamma,\xi}$ has $O(mn/\varepsilon^d)$ size. The total time needed over all cells in $\mathcal{G}_3$ and all canonical segments in all lines in $\mathcal{L}$ is $\tilde{O}((mn)^5/\varepsilon^{5d-1})$.

Let $p_\gamma$ be the center of the cell $\gamma$. Define $c_{\gamma,\xi}$ to be the cell in $C_{\gamma,\xi}$ such that $p_\xi p_\gamma \cap (c_{\gamma,\xi} \oplus B_{5\varepsilon\delta})$ is nearest to $p_\xi$ among $\{p_\xi p_\gamma \cap (c \oplus B_{5\varepsilon\delta}) : c \in C_{\gamma,\xi}\}$. The total time to compute $c_{\gamma,\xi}$ over all cells in $\mathcal{G}_3$ and all canonical segments in all lines in $\mathcal{L}$ is $O((mn)^5/\varepsilon^{5d-1})$.

Finally, for every line $\ell \in \mathcal{L}$, we store the canonical segments in $\ell$ in an interval tree $T_\ell$ [7]. It uses linear space and preprocessing time. For any query point in $\ell$, one can search $T_\ell$ in $O(\log \frac{mn}{\varepsilon})$ time to find the canonical segment in $\ell$ that contains the query point. For each canonical segment $\xi$ stored in $T_\ell$, we keep a dictionary $T_\xi$ that stores the set $\{(\gamma, c_{\gamma,\xi}) : \gamma \in \mathcal{G}_3\}$ with $\gamma$ as the key. For any cell $\gamma \in \mathcal{G}_3$, we can search $T_\xi$ in $O(\log \frac{mn}{\varepsilon})$ time to report $c_{\gamma,\xi}$. These interval trees and dictionaries have a total size of $O((mn)^4/\varepsilon^{4d-1})$, and they can be constructed in $\tilde{O}((mn)^4/\varepsilon^{4d-1})$ time.

The data structures $D_{\text{anp}}$, $D_{\text{anl}}$, $T_\ell$ for $\ell \in \mathcal{L}$, and $T_\xi$ for all canonical segments $\xi$'s are what we need to support the $(11\varepsilon\delta)$-segment queries on $\mathcal{G}_1$.

▶ **Lemma 12.** *We can construct $D_{\text{anp}}$, $D_{\text{anl}}$, $T_\ell$ for $\ell \in \mathcal{L}$, and $T_\xi$ for every $\ell \in \mathcal{L}$ and every canonical segment $\xi \subset \ell$ in $O(\sqrt{d}/\varepsilon)^{O(d/\varepsilon^2)} \cdot \tilde{O}((mn)^{O(1/\varepsilon^2)})$ space and preprocessing time.*

In the definition of $c_{\gamma,\xi}$, one may ask what if $p_\xi p_\gamma$ does not intersect $c \oplus B_{5\varepsilon\delta}$ for some $c \in C_{\gamma,\xi}$. We prove that this cannot happen. We also establish some other properties.

▶ **Lemma 13.** *Let $\gamma$ be a cell in $\mathcal{G}_3$. Let $\xi$ be a canonical segment. Let $L_\xi$ be the cylinder with $\xi$ as the axis and radius $2\varepsilon\delta$.*

**(i)** *For every cell $c \in \mathcal{G}_1$, if $c \cap xy \neq \emptyset$ for some points $x \in L_\xi$ and $y \in \gamma$, then $c \in C_{\gamma,\xi}$.*

**(ii)** *For every point $x \in L_\xi$, every point $y \in \gamma$ and every cell $c \in C_{\gamma,\xi}$, $xy \cap (c \oplus B_{5\varepsilon\delta}) \neq \emptyset$.*

**(iii)** *Let $\lambda$ be any value greater than or equal to $11\varepsilon\delta$. When we walk from a point $x \in L_\xi$ to a point $y \in \gamma$, we cannot hit any $c \in C_{\gamma,\xi}$ earlier than $c_{\gamma,\xi} \oplus B_\lambda$ irrespective of the choices of $x$ and $y$.*

## 4.2 Answering a query

Given an oriented segment $w_j w_{j+1}$ of the query curve $\sigma$, we answer the $(11\varepsilon\delta)$-segment query with $w_j w_{j+1}$ on $\mathcal{G}_1$ by the following steps.

- Step 1: We query $D_{\mathrm{anp}}$ with $\mathrm{aff}(w_j w_{j+1})$ to report a grid vertex $x$ of $\mathcal{G}_1$. This takes $\tilde{O}((mn)^{0.5+\varepsilon}/\varepsilon^d)$ time.
- Step 2: We check the distance $d(x, \mathrm{aff}(w_j w_{j+1}))$. If $d(x, \mathrm{aff}(w_j w_{j+1})) > (1+\varepsilon)\varepsilon\delta$, then $\mathrm{aff}(w_j w_{j+1})$ is at distance more than $\varepsilon\delta$ from the closest grid vertex of $\mathcal{G}_1$, which implies that $\mathrm{aff}(w_j w_{j+1})$ does not intersect any cell in $\mathcal{G}_1$. In this case, we report null.
  We also check the distances $d(w_j, \mathcal{L})$ and $d(w_{j+1}, \mathcal{L})$. We query $D_{\mathrm{anl}}$ with $w_j$ in $\tilde{O}(1)$ time to find a line $\ell_j \in \mathcal{L}$. If $d(w_j, \ell_j) > 2\varepsilon\delta$, then $d(w_j, \mathcal{L}) > \varepsilon\delta$, which implies that $w_j$ is at distance farther than $\delta$ from $\mathrm{aff}(\tau_{i,a})$ for any $\tau_i \in T$ and any $a \in [m-1]$. As a result, $d_F(\sigma, \tau_i) > \delta$ for all $\tau_i \in T$, so we report "no" for the $(\kappa, \delta)$-ANN query. Analogously, we query $D_{\mathrm{anl}}$ with $w_{j+1}$ in $\tilde{O}(1)$ time to find a line $\ell_{j+1} \in \mathcal{L}$. If $d(w_{j+1}, \ell_{j+1}) > 2\varepsilon\delta$, we report "no" for the $(\kappa, \delta)$-ANN query.
- Step 3: Suppose that $d(x, \mathrm{aff}(w_j w_{j+1})) \le (1+\varepsilon)\varepsilon\delta$, $d(w_j, \ell_j) \le 2\varepsilon\delta$, and $d(w_{j+1}, \ell_{j+1}) \le 2\varepsilon\delta$. Then, we check the cells in $G(x \oplus B_{2\varepsilon\delta})$ in $O(\varepsilon^{-d})$ time to find one that intersects $\mathrm{aff}(w_j w_{j+1})$. Let $\gamma$ be this cell. We do not know if $\gamma$ belongs to $\mathcal{G}_1$ or not. Nevertheless, since $x$ is a grid vertex of $\mathcal{G}_1$, $\gamma$ is within a distance $(1+3\varepsilon)\delta$ from some input curve vertex. Therefore, $\gamma$ must be a cell in $\mathcal{G}_3$. There are three cases depending on the relative positions of $w_j$ and $\gamma$.
  - Step 3(a): $w_j \in \gamma \cap \mathrm{aff}(w_j w_{j+1})$. We claim that $\mathcal{G}_1 \cap G(w_j \oplus B_{7\varepsilon\delta})$ is non-empty, and we report an arbitrary cell in it as the answer for the $(11\varepsilon\delta)$-segment query. This step takes $O(\varepsilon^{-d})$ time.
  - Step 3(b): $w_j$ precedes $\gamma \cap \mathrm{aff}(w_j w_{j+1})$ along $\mathrm{aff}(w_j w_{j+1})$ oriented from $w_j$ to $w_{j+1}$. We query $T_{\ell_j}$ to find the canonical segment $\xi \subset \ell_j$ that contains the projection of $w_j$ in $\ell_j$. Then, we query $T_\xi$ with $\gamma$ to return $c_{\gamma,\xi}$ as the answer for the $(11\varepsilon\delta)$-segment query. The time needed is $O(\log \frac{mn}{\varepsilon})$.
  - Step 3(c): $\gamma \cap \mathrm{aff}(w_j w_{j+1})$ precedes $w_j$ along $\mathrm{aff}(w_j w_{j+1})$ oriented from $w_j$ to $w_{j+1}$. We query $T_{\ell_{j+1}}$ to find the canonical segment $\xi \subset \ell_{j+1}$ that contains the projection of $w_{j+1}$ in $\ell_{j+1}$. Then, we query $T_\xi$ with $\gamma$ to obtain $c_{\gamma,\xi}$. We claim that $G(c_{\gamma,\xi} \oplus B_{5\varepsilon\delta}) \subset \mathcal{G}_3$ and some cell in $G(c_{\gamma,\xi} \oplus B_{5\varepsilon\delta})$ intersects $w_j w_{j+1}$. Pick one such cell $\hat{\gamma}$ in $O(\varepsilon^{-d})$ time. Either step 3(a) or 3(b) is applicable with $\gamma$ replaced by $\hat{\gamma}$. Whichever case is applicable, we jump to that case with $\gamma$ replaced by $\hat{\gamma}$ to return an answer for the $(11\varepsilon\delta)$-segment query. The time needed is $\tilde{O}(\varepsilon^{-d})$.

► **Lemma 14.** *It takes $\tilde{O}((mn)^{0.5+\varepsilon}/\varepsilon^d)$ time to answer a $(11\varepsilon\delta)$-segment query.*

Lemmas 12 and 14 gives the performance of the $(11\varepsilon\delta)$-segment query data structure in Lemma 8. The proof of the query output correctness in Lemma 8 can be found in the full version.

## 5 Conclusion

We present $(1+\varepsilon)$-ANN and $(3+\varepsilon)$-ANN data structures that achieve sublinear query times without having space complexities that are proportion to $\min\{m^{\Omega(d)}, n^{\Omega(d)}\}$ or exponential in $\min\{m, n\}$. The query times are $\tilde{O}(k(mn)^{0.5+\varepsilon}/\varepsilon^{O(d)} + k(d/\varepsilon)^{O(dk)})$ for $(1+\varepsilon)$-ANN and $\tilde{O}(k(mn)^{0.5+\varepsilon}/\varepsilon^{O(d)})$ for $(3+\varepsilon)$-ANN. In two and three dimensions, the query times can be improved to $\tilde{O}(k/\varepsilon^{O(k)})$ for $(1+\varepsilon)$-ANN and $\tilde{O}(k)$ for $(3+\varepsilon)$-ANN. It is an open problem is to lower the exponential dependence on $d$ and $k$.

## References

1 P.K. Agarwal and J. Matoušek. Ray shooting and parametric search. *SIAM Journal on Computing*, 22(4):794–806, 1993.

2 P.K. Agarwal, N. Rubin, and M. Sharir. Approximate nearest neighbor search amid higher-dimensional flats. In *Proceedings of the European Symposium on Algorithms*, pages 4:1–4:13, 2017.

3 H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry and Applications*, 5:75–91, 1995.

4 A. Andoni, P. Indyk, R. Krauthgamer, and H.L. Nguyen. Approximate line nearest neighbor in high dimensions. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 293–301, 2009.

5 K. Bringmann, A. Driemel, A. Nusser, and I. Psarros. Tight bounds for approximate near neighbor searching for time series under Fréchet distance. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 517–550, 2022.

6 S.-W. Cheng and H. Huang. Curve simplification and clustering under Fréchet distance. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 1414–1432, 2023.

7 T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, second edition, 2001.

8 M. de Berg, D. Halperin, M. Overmars, J. Snoeyink, and M. van Kreveld. Efficient ray shooting and hidden surface removal. *Algorithmica*, 12:30–53, 1994.

9 A. Driemel and I. Psarros. $(2+\epsilon)$-ANN for time series under the Fréchet distance. *arXiv preprint v5*, 2021. arXiv:2008.09406.

10 A. Driemel and F. Silvestri. Locality-sensitive hashing of curves. In *Proceedings of the International Symposium on Computational Geometry*, pages 37:1–37:16, 2017.

11 T. Eiter and H. Mannila. Computing discrete Fréchet distance. Technical Report CD-TR 94/64, Christian Doppler Laboratory for Expert Systems, TU Vienna, Austria, 1994.

12 I.Z. Emiris and I. Psarros. Products of Euclidean metrics, applied to proximity problems among curves: Unified treatment of discrete fréchet and dynamic time warping distances. *ACM Transactions on Spatial Algorithms and Systems*, 6(4):1–20, 2020.

13 A. Filtser, O. Filtser, and M.J. Katz. Approximate nearest neighbor for curves: simple, efficient, and deterministic. In *Proceedings of the International Colloquium on Automata, Languages, and Programming*, pages 48:1–48:19, 2020.

14 S. Har-Peled. A replacement for Voronoi diagrams of near linear size. In *Proceedings of the Annual IEEE Symposium on Foundations of Computer Science*, pages 94–103, 2001.

15 S. Har-Peled, P. Indyk, and R. Motwani. Approximate nearest neighbor: towards removing the curse of dimensionality. *Theory of Computing*, 8:321–350, 2012.

16 P. Indyk. Approximate nearest neighbor algorithms for Fréchet distance via product metrics. In *Proceedings of the Annual Symposium on Computational Geometry*, pages 102–106, 2002.

17 P. Indyk and R. Motwani. Approximate nearest neighbor: towards removing the curse of dimensionality. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, pages 604–613, 1998.

18 Mirzanezhad M. On the approximate nearest neighbor queries among curves under the Fréchet distance. *arXiv preprint*, 2020. `arXiv:2004.08444`.

19 C. Shahabi, M. Kolahdouzan, and M. Sharifzadeh. A road network embedding technique for $k$-nearest neighbor search in moving object databases. *GeoInformatica*, 7:255–273, 2003.

20 Z. Song and N. Roussopoulos. $K$-nearest neighbor search for moving query point. In *Proceedings of the International Symposium on Spatial and Temporal Databases*, pages 79–96, 2001.

21 Y. Tao and D. Papadias. Parameterized queries in spatio-temporal databases. In *Proceedings of ACM International Conference on Management of Data*, pages 334–345, 2002.