Dynamic Time Warping-Based Proximity Problems

Boris Aronov
Department of Computer Science and Engineering, Tandon School of Engineering, New York University, Brooklyn, NY, USA
boris.aronov@nyu.edu

Matthew J. Katz
Department of Computer Science, Ben-Gurion University of the Negev, Beer Sheva, Israel
matya@cs.bgu.ac.il

Elad Sulami
Department of Computer Science, Ben-Gurion University of the Negev, Beer Sheva, Israel
eladsulami@gmail.com

Abstract

Dynamic Time Warping (DTW) is a well-known similarity measure for curves, i.e., sequences of points, and especially for time series. We study several proximity problems for curves, where dynamic time warping is the underlying similarity measure. More precisely, we focus on the variants of these problems, in which, whenever we refer to the dynamic time warping distance between two curves, one of them is a line segment (i.e., a sequence of length two). These variants already reveal some of the difficulties that occur when dealing with the more general ones.

Specifically, we study the following three problems: (i) *distance oracle*: given a curve C in $\mathbb{R}^d$, preprocess it to accommodate distance computations between query segments and C, (ii) *segment center*: given a set $\mathcal{C}$ of curves in $\mathbb{R}^d$, find a segment s that minimizes the maximum distance between s and a curve in $\mathcal{C}$, and (iii) *segment nearest neighbor*: given $\mathcal{C}$, construct a data structure for segment nearest neighbor queries, i.e., return the curve in $\mathcal{C}$ which is closest to a query segment s. We present solutions to these problems in any constant dimension $d \geq 1$, using L_∞ for inter-point distances. We also consider the approximation version of the first problem, using L_1 for inter-point distances. That is, given a length-m curve C in $\mathbb{R}^d$, we construct a data structure of size $O(m \log m)$ that allows one to compute a 2-approximation of the distance between a query segment s and C in $O(\log^3 m)$ time.

Finally, we describe an interesting experimental study that we performed, which is related to the first problem above.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Design and analysis of algorithms

Keywords and phrases dynamic time warping, distance oracle, clustering, nearest-neighbor search

Digital Object Identifier 10.4230/LIPIcs.MFCS.2020.9

Funding *Boris Aronov*: Partially supported by NSF grant CCF-15-40656 and by grant 2014/170 from the US-Israel Binational Science Foundation.
Matthew J. Katz: Partially supported by grant 1884/16 from the Israel Science Foundation and by grant 2014/170 from the US-Israel Binational Science Foundation.

1 Introduction

Dynamic time warping (DTW) is a popular similarity measure for curves, first introduced in [2]. One of its early applications was to speech recognition, and since then DTW has been employed in a wide range of areas such as signature matching, sign language recognition, data mining, information retrieval, signal processing, and protein sequence alignment. DTW can be

applied to almost any time series data, and it is often used for comparing temporal sequences of video, audio and graphics data; see the chapter on DTW in Müller's book [12, Chapter 4] for further information on DTW.

Given two integers m_1, m_2, let $\tau := \langle (i_1, j_1), \ldots, (i_t, j_t) \rangle$ be a sequence of pairs, such that $i_1 = j_1 = 1$, $i_t = m_1$, $j_t = m_2$, and for each $1 < k \leq t$, one of the following conditions holds:

(i) $i_k = i_{k-1} + 1$ and $j_k = j_{k-1}$,
(ii) $i_k = i_{k-1}$ and $j_k = j_{k-1} + 1$, or
(iii) $i_k = i_{k-1} + 1$ and $j_k = j_{k-1} + 1$.

We call such a sequence τ an *alignment* of two curves of lengths m_1 and m_2, respectively (see below).

In this paper, by a *curve* P of length m, we mean a sequence $P = \langle p_1, \ldots, p_m \rangle$ of m points in $\mathbb{R}^d$. Let $P = \langle p_1, \ldots, p_{m_1} \rangle$ and $Q = \langle q_1, \ldots, q_{m_2} \rangle$ be two curves of lengths m_1 and m_2, respectively, in $\mathbb{R}^d$. We say that an alignment τ of P and Q *matches* p_i and q_j if $(i, j) \in \tau$, see Figure 1. The *cost* of an alignment τ of P and Q is $\sigma_{dtw}(\tau(P, Q)) := \sum_{(i,j)\in\tau} \|p_i - q_j\|$,

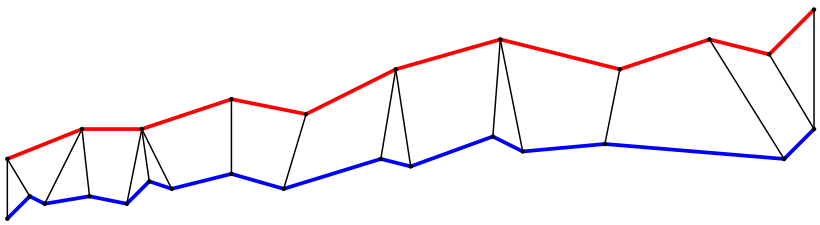


Figure 1 An alignment of two curves.

where $\|\cdot\|$ denotes the inter-point metric being used; in this paper we will focus on L_1 and L_∞ metrics. The *dynamic time warping* distance between P and Q is defined as

$$d_{dtw}(P, Q) := \min_{\tau \in \mathcal{T}} \sigma_{dtw}(\tau(P, Q)),$$

with the minimum taken over the set $\mathcal{T}$ of all alignments τ between P and Q.

The running time of the standard dynamic-programming algorithm for computing $d_{dtw}(P, Q)$ is $\Theta(m_1 m_2)$, which has been the asymptotically fastest way to compute it, even in the one-dimensional case. Recently, Gold and Sharir [8] managed to break the quadratic bound in the one-dimensional case by presenting an algorithm with running time $O(m^2 \log\log\log m / \log\log m)$, assuming $m = m_1 = m_2$. They obtain the same bound on the running time in higher (constant) dimensions if the point-wise distance is L_1 or L_∞.

In this paper, we study several fundamental problems for curves, using dynamic time warping as the similarity measure between pairs of curves. More precisely, we are interested in variants of several important problems, in which, whenever we refer to the dynamic time warping distance between two curves, one of them is a line segment (i.e., a sequence of length two). Since line segments are the shortest non-trivial curves, these variants are quite natural and, moreover, they already reveal some of the difficulties that occur in the more general variants of the problems.

Notice that computing the dynamic time warping distance between a curve $C = \langle p_1, \ldots, p_m \rangle$ of length m and a line segment $s = \overline{ab}$ is equivalent to finding the partition of C into two non-empty parts, a prefix and a suffix, such that each point of the prefix is matched to a and each point of the suffix is matched to b (with no point matched to both a and b), and the cost associated with the partition is minimum. Somewhat unsurprisingly computing the dynamic time warping distance between a curve and a segment is an operation requiring $\Theta(m)$ time in absence of preprocessing.

This motivates the first problem that we consider (see Section 2). Given a curve C in $\mathbb{R}^d$, construct a *distance oracle* for C for line segments, which is a data structure supporting queries of the form: given a query segment s in $\mathbb{R}^d$, return $d_{dtw}(s, C)$. We consider two main versions of the problem, where in the first the underlying point-wise distance is L_1 and the answers to queries are approximate, while in the second the underlying point-wise distance is L_∞ and the answers are exact.

In Section 3 we study the *segment center* problem for a set of curves $\mathcal{C}$: Given a set $\mathcal{C} = \{C_1, \ldots, C_n\}$ of n length-m curves in $\mathbb{R}^d$, find a segment s in $\mathbb{R}^d$ that minimizes the distance to the furthest curve, that is $\max\{d_{dtw}(s, C_1), \ldots, d_{dtw}(s, C_n)\}$ is minimum. Here we assume that the point-wise distance is L_∞ and present two solutions to the problem, where the second one is more efficient when $m < n$.

In Section 4 we consider our last algorithmic problem. Given a collection of curves $\mathcal{C}$ in $\mathbb{R}^d$ as above, construct a data structure for segment nearest-neighbor queries, that is, one that returns the curve in $\mathcal{C}$ whose dynamic time warping distance to the query segment is minimum. Here too we assume that the point-wise distance is L_∞. (The general version of this problem, where the queries are also length-m curves was studied by Emiris and Psarros [6] and Filtser et al. [7] in the approximation setting.)

Finally, in Section 5 we describe an experimental study related to the first problem mentioned above. The goal of this study was to get a better idea on the connection between the query segments and the "correct" partition of the given curve C. In particular, is there an underlying Voronoi-diagram–like geometric structure which is simple enough and may prove useful in the construction of a distance oracle for C? Our experiments suggest that the answer to this question may be positive, at least for curves on the line.

2 Distance Oracle

Let $C = \langle p_1, \ldots, p_m \rangle$ be a curve in $\mathbb{R}^d$. In this section, we study the problem of, given C, constructing a data structure supporting queries of the form: given a query segment s, return the dynamic time warping distance between C and s. We consider two main versions of the problem, where in the first the underlying point-wise distance is L_1 and the answers to queries are approximate, while in the second the underlying point-wise distance is L_∞ and the answers are exact. For $d = 1$, $L_\infty = L_1 = L_2$, of course.

2.1 Approximate queries under L_1

In this section, we assume that the distance between two points x and y in $\mathbb{R}^d$ is $\|x - y\|_1 := \sum_{i=1}^{d} |x_i - y_i|$, and we present a data structure that facilitates fast approximate queries. We start with the one-dimensional case.

A solution for $d = 1$

Let $s = \overline{pq}$ be a segment. We define the functions F, G, and H for $i \in \{1, \ldots, m-1\}$:

$$F(i) := \sum_{j=1}^{i} \|p_j - p\|, \quad G(i) := \sum_{j=i+1}^{m} \|p_j - q\|, \quad \text{and} \quad H(i) := F(i) + G(i).$$

That is, $F(i)$ is the sum of distances between p and the first i points of C, $G(i)$ is the sum of distances between q and the last $m - i$ points of C, and $H(i)$ is the dynamic time warping distance between C and s, assuming the first i points of C are matched to p and the last

$m - i$ points are matched to q. (Notice that we can assume that no point in C is matched to both p and q, as such an alignment can be always improved by removing one or more of the pairs.) Thus, $d_{dtw}(C, s) = \min_{1 \le i \le m-1} H(i)$.

Slightly abusing the notation, we view F and G as piecewise-linear functions defined on $[1, m-1]$ rather than just at the values $1, 2, \ldots, m-1$. Notice that F is increasing, namely, $F(i+1) > F(i)$, while G is decreasing, that is, $G(i+1) < G(i)$, for $i = 1, \ldots m-2$.[1] This implies that, unless the graph of F is above that of G (when $F(1) > G(1)$) or the graph of G is above that of F (when $G(m-1) > F(m-1)$), the two graphs intersect.

▶ **Lemma 2.1.** *(i) If the graph of F is above that of G, then $H(1) \le 2d_{dtw}(C, s)$, and (ii) if the graph of G is above that of F, then $H(m-1) \le 2d_{dtw}(C, s)$.*

Proof. We prove the first claim. The proof of the second one is symmetric. Let ℓ be the index for which $d_{dtw}(C, s) = H(\ell)$. Then,

$$\begin{aligned} 2d_{dtw}(C, s) &= 2H(\ell) = 2(F(\ell) + G(\ell)) \\ &\ge 2F(\ell) \ge 2F(1) \ge F(1) + G(1) = H(1)\,. \end{aligned}$$

◀

Now, assume that the graphs of F and G do intersect, and let k, $1 \le k \le m-2$, be the index for which $F(k) \le G(k)$ and $F(k+1) \ge G(k+1)$.

▶ **Lemma 2.2.** *If the graphs of F and G intersect,*

$$d_{dtw}(C, s) \le \min(H(k), H(k+1)) \le 2d_{dtw}(C, s).$$

Proof. The left in inequality is obvious, since $d_{dtw}(C, s) = \min_{1 \le i \le m-1} H(i)$. As for the right inequality, let ℓ be the index for which $d_{dtw}(C, s) = H(\ell)$. Assume first that $\ell \le k$, then

$$\begin{aligned} 2d_{dtw}(C, s) &= 2H(\ell) = 2(F(\ell) + G(\ell)) \ge 2G(\ell) \\ &\ge 2G(k) \ge F(k) + G(k) = H(k) \ge \min(H(k), H(k+1))\,. \end{aligned}$$

On the other hand, if $\ell > k$, then

$$\begin{aligned} 2d_{dtw}(C, s) &= 2H(\ell) = 2(F(\ell) + G(\ell)) \ge 2F(\ell) \ge 2F(k+1) \\ &\ge F(k+1) + G(k+1) = H(k+1) \ge \min(H(k), H(k+1))\,. \end{aligned}$$

◀

We obtain the following high-level algorithm for computing a 2-approximation for $d_{dtw}(C, s)$, that is a value v such that $v/2 \le d_{dtw}(C, s) \le v$:

1. If $F(1) \ge G(1)$ return $H(1)$.
2. If $G(m-1) \ge G(m-1)$ return $H(m-1)$.
3. Perform a binary search to find the largest index k for which $F(k) \le G(k)$; return $\min(H(k), H(k+1))$.

Recall that s is not known in advance. To complete the description of the algorithm, we need to show how to preprocess C, so that the values $F(i)$ and $G(i)$ can be computed efficiently. We build a standard two-dimensional orthogonal range tree for the set of points $\{(i, p_i) \mid 1 \le i \le m\}$, where the tree of the first level, T, is built using the x-coordinates of the points, i.e., the indices $1, \ldots, m$. Each node u of T points to a second-level tree, T_u,

[1] Strictly speaking, F is non-decreasing and G non-increasing, as it is possible for a point of C to coincide with p and/or q, but hereafter we will stick with the less cumbersome terminology.

which is built using the y-coordinates of the points whose x-coordinates are stored in the subtree of T rooted at u. In addition, we store at each internal node v of T_u the number and sum of the y-coordinates stored in v's subtree. The total size of the data structure is $O(m \log m)$.

Now, given $s = \overline{pq}$ and i, set $A \coloneqq \sum_{j \le i, p_j \le p} p_j$ and $B \coloneqq \sum_{j \le i, p_j > p} p_j$, and let t be the number of terms in the first sum. Notice that

$$F(i) = \sum_{j \le i, p_j \le p} (p - p_j) + \sum_{j \le i, p_j > p} (p_j - p) = (2t - i)p + B - A\,,$$

and by searching in the orthogonal range tree with (i, p), one can compute A, B, and t (and therefore $F(i)$) in $O(\log^2 m)$ time. Finally, $G(i)$ can be computed similarly within the same time bound. Therefore a single comparison in the binary search of our high-level algorithm can be carried out in $O(\log^2 m)$ time and we obtain the following theorem.

▶ **Theorem 2.3.** *For any length-m curve C on the line, one can construct in $O(m \log m)$ time a data structure of size $O(m \log m)$, which, given a query segment s, returns in $O(\log^3 m)$ time a 2-approximation of $d_{dtw}(C, s)$, i.e., a value v such that $v/2 \le d_{dtw}(C, s) \le v$.*

A solution for $d \ge 2$

The high-level algorithm presented above works in any dimension, so we only need to modify our data structure for computing the values $F(i)$ and $G(i)$ efficiently, given a segment $s = \overline{pq}$ and index i.

Notice that now $F(i) = \sum_{\ell=1}^{d} F_\ell(i)$, where

$$F_\ell(i) \coloneqq \sum_{j \le i, p_j^\ell \le p^\ell} (p^\ell - p_j^\ell) + \sum_{j \le i, p_j^\ell > p^\ell} (p_j^\ell - p^\ell)\,,$$

which, in turn, by letting t_ℓ be the number of terms in the first sum and setting $A_\ell \coloneqq \sum_{j \le i, p_j^\ell \le p^\ell} p_j^\ell$ and $B_\ell \coloneqq \sum_{j \le i, p_j^\ell > p^\ell} p_j^\ell$, is equal to $(2t_\ell - i)p^\ell + B_\ell - A_\ell$; we use superscripts to denote coordinates of points in $\mathbb{R}^d$. We thus modify our data structure, so that each node u of the first-level tree T points to d second-level trees, $T_{u,1}, \ldots, T_{u,d}$, where $T_{u,\ell}$ is built using the ℓ'th coordinates of the points whose indices are stored in the subtree of T rooted at u. The size of the modified data structure is still $O(m \log m)$, assuming d is a constant, and by searching in it with (i, p^ℓ), one can compute A_ℓ, B_ℓ and t_ℓ, for $\ell = 1, \ldots, d$, in $O(\log^2 m)$ time. We obtain the following theorem.

▶ **Theorem 2.4.** *For any length-m curve C in $\mathbb{R}^d$, one can construct in $O(m \log m)$ time a data structure of size $O(m \log m)$, which, given a query segment s, returns in $O(\log^3 m)$ time a 2-approximation of $d_{dtw}(C, s)$, i.e., a value v such that $v/2 \le d_{dtw}(C, s) \le v$.*

2.2 Exact queries under L_∞

We now consider the problem of computing $d_{dtw}(C, s)$ exactly, where, as before, $C = \langle p_1, \ldots, p_m \rangle$ is a fixed given curve in $\mathbb{R}^d$ and $s = \overline{pq}$ is a query segment in $\mathbb{R}^d$, and the distance between two points x and y is $\|x - y\|_\infty \coloneqq \max_{i=1}^{d} |x_i - y_i|$. We first describe a solution for the one-dimensional version of the problem.

A solution for $d = 1$

Let $\tau = \langle \tau_1, \ldots, \tau_{m+1} \rangle$ be the partition of the real line into intervals (two of which are unbounded), induced by the points of C (after sorting them from left to right). We denote the left and right endpoints of τ_i by $l(\tau_i)$ and $r(\tau_i)$, respectively, where $l(\tau_1) = -\infty$ and $r(\tau_{m+1}) = \infty$.

For each pair of intervals (τ_a, τ_b), we construct a data structure for segment queries $s = \overline{ab}$, such that $a \in \tau_a$ and $b \in \tau_b$. We now describe the data structure for a fixed pair (τ_a, τ_b).

For each j, $1 \leq j < m$, we define a bivariate function f_j:

$$f_j(a,b) \coloneqq \sum_{p_i \leq l(\tau_a),\, i \leq j} (a - p_i) + \sum_{p_i \geq r(\tau_a),\, i \leq j} (p_i - a) + \sum_{p_i \leq l(\tau_b),\, i > j} (b - p_i) + \sum_{p_i \geq r(\tau_b),\, i > j} (p_i - b)\,.$$

Notice that $f_j(a,b)$ is simply the dynamic time warping distance between C and $s = \overline{ab}$, assuming C is split into a prefix of size j (which is matched to a) and a suffix of size $m - j$ (which is matched to b). Moreover, notice that f_j is a linear bivariate function defined over the rectangular range $R \coloneqq \tau_a \times \tau_b$, and therefore its graph is a plane in $\mathbb{R}^3$ (over R). Finally, given $1 \leq j < m$ and $(a,b) \in R$, one can compute the value $f_j(a,b)$ in $O(1)$ time, after a preprocessing stage of $O(m)$ time. (In the preprocessing stage, we compute, for each $1 \leq j < m$, the sums $S_j^1 = \sum_{p_i \leq l(\tau_a),\, i \leq j} (l(\tau_a) - p_i)$, $S_j^2 = \sum_{p_i \geq r(\tau_a),\, i \leq j} (p_i - r(\tau_a))$, $S_j^3 = \sum_{p_i \leq l(\tau_b),\, i > j} (l(\tau_b) - p_i)$, $S_j^4 = \sum_{p_i \geq r(\tau_b),\, i > j} (p_i - r(\tau_b))$ and record the numbers m_j^1, m_j^2, m_j^3, m_j^4, where m_j^k is the number of terms in the sum S_j^k, for $k = 1, \ldots, 4$. Computing the first two sums and corresponding numbers for $j = 1$ requires $O(1)$ time, and given the sums for some $1 \leq j < m-1$, one can compute the sums for $j+1$ in $O(1)$ time; the last two sums can be similarly computed working backwards in total $O(m)$ time. Now, given j and $(a,b) \in R$, $f_j(a,b) = S_j^1 + m_j^1(a - l(\tau_a)) + S_j^2 + m_j^2(r(\tau_a) - a) + S_j^3 + m_j^3(b - l(\tau_b)) + S_j^4 + m_j^4(r(\tau_b) - b)$.)

Let $S = S(a,b) \coloneqq \min_j f_j(a,b)$ be the lower envelope of the planes $f_1, \ldots, f_{m-1}$ (over R). Then, S is a piecewise-linear function of complexity $O(m)$, and the dynamic time warping distance between C and s is $S(a,b)$. We thus compute S and preprocess its graph for vertical ray-shooting queries: Given $s = \overline{ab}$, return the plane f_j containing the face of the graph of S lying above the point $(a,b) \in R$. This can be done in $O(m \log m)$ time, after which a ray-shooting query can be answered in $O(\log m)$ time (and then $f_j(a,b)$ can be evaluated in $O(1)$ time).

This completes the description of the data structure for the pair (τ_a, τ_b). Since we need $O(m^2)$ such data structures, we obtain the following theorem.

▶ **Theorem 2.5.** *For a length-m curve C on the line, one can construct in $O(m^3 \log m)$ time a data structure of size $O(m^3)$ which, given a query segment s on the line, can compute $d_{dtw}(C, s)$ in $O(\log m)$ time.*

A solution for $d \geq 2$

We generalize our solution for $d = 1$ to any constant dimension $d \geq 2$. Let $p = (p^1, \ldots, p^d)$ be a point of C and consider the partition of space into cells that is obtained by drawing the following $2\binom{d}{2}$ hyperplanes through p. For each $1 \leq i < j \leq d$, draw the hyperplanes $x^i - p^i = x^j - p^j$ and $x^i - p^i = p^j - y^j$. Given a query point q, by the cell of the partition in which it lies, we know the coordinate i that determines $\|q - p\|_\infty$ and whether the distance is $q^i - p^i$ or $p^i - q^i$. We thus draw these $2\binom{d}{2}$ hyperplanes, for each point p_i of C, to obtain an arrangement of $O(m)$ hyperplanes in $\mathbb{R}^d$.

Now, for each pair of cells (τ_a, τ_b) of this arrangement, we construct a data structure for segment queries $s = \overline{ab}$, such that $a \in \tau_a$ and $b \in \tau_b$. As above, we define $(m-1)$ $2d$-variate linear functions, $f_j(a,b)$, over the box $B := \tau_a \times \tau_b$. The graphs of these functions are hyperplanes in $\mathbb{R}^{2d+1}$ and we compute their lower envelope $S = S(a,b)$ (over B). The total complexity of S is $O(m^{\lfloor \frac{2d+1}{2} \rfloor}) = O(m^d)$ [11], it can be computed in time $O(m^d)$ [4], and we can preprocess the graph of S for logarithmic-time vertical ray-shooting queries in $O(m^d)$ time [10] (see also [3] for a more recent randomized algorithm).

Since we need $O(m^{2d})$ such data structures, we obtain the following theorem.

▶ **Theorem 2.6.** *For a length-m curve C in $\mathbb{R}^d$, one can construct in $O(m^{3d} \log m)$ time a data structure of size $O(m^{3d})$ which, given a query segment s in $\mathbb{R}^d$, can compute $d_{dtw}(C, s)$ in $O(\log m)$ time.*

Proof. Given a query segment $s = \overline{ab}$, we need to describe how the pair (τ_a, τ_b), such that $a \in \tau_a$ and $b \in \tau_b$, is determined. This can be done by performing $2\binom{d}{2}$ binary searches for endpoint of s (after presorting the points $p_1, \ldots, p_m$ in the appropriate $2\binom{d}{2}$ directions). Each pair (τ_a, τ_b) is associated with a unique tuple of $4\binom{d}{2}$ indices in these orders, so a suitable pair can be identified using a standard search structure.[2] ◀

3 Segment Center

Let $\mathcal{C} = \{C_1, \ldots, C_n\}$ be a set of n length-m curves in $\mathbb{R}^d$. In this section, we consider the *segment center* problem for $\mathcal{C}$ under L_∞: Given $\mathcal{C}$, find a segment s in $\mathbb{R}^d$ minimizing $\max\{d_{dtw}(s, C_1), \ldots, d_{dtw}(s, C_n)\}$. We present two solutions to the problem, where the second one is more efficient when m is significantly smaller than n. For each of the solutions, we begin with a detailed description for the one-dimensional version, which we then generalize to higher dimensions.

3.1 First solution

The case of $d = 1$

Let $\tau = \langle \tau_1, \ldots, \tau_{mn+1} \rangle$ be the partition of the line into intervals (of which two are unbounded), induced by the nm points of the curves of $\mathcal{C}$ (after sorting them from left to right). For each pair of intervals (τ_a, τ_b), we compute the optimal segment $s = \overline{ab}$ with $a \in \tau_a$ and $b \in \tau_b$. Then, we return the segment whose corresponding distance is minimum, among these optimal segments.

We now describe how to find the optimal segment for a given pair (τ_a, τ_b).

1. Let C_i be a curve in $\mathcal{C}$. For each j, $1 \leq j < m$, we define the function $f_j(a,b)$ as in Section 2.2. Recall that $f_j(a,b)$ is the dynamic time warping distance between C_i and s, assuming C_i is split into a prefix of size j and a suffix of size $m-j$. It is a bivariate linear function, so its graph is a plane in $\mathbb{R}^3$ (over $R := \tau_a \times \tau_b$).
2. Let S_i denote the lower envelope of the planes $f_1, \ldots, f_{m-1}$. Then, S_i is a piecewise-linear function of complexity $O(m)$, or more precisely, its graph is the boundary of an unbounded convex polyhedron with at most m facets whose projection to the xy-plane is R, and the dynamic time warping distance between C_i and s is $S_i(a,b)$.

[2] A similar issue arises again several times below and can be handled by analogous means; we omit the details.

3. Compute the lower envelopes $S_1, \ldots, S_n$ corresponding to $C_1, \ldots, C_n$. Let U denote the upper envelope of $S_1, \ldots, S_n$. Then $U(a, b)$ is the maximum of the dynamic time warping distances between the curves of $\mathcal{C}$ and s. Since the lower envelopes S_i are piecewise-linear, so is U. As for the complexity of U, since each graph of C_i can be triangulated without increasing its asymptotic complexity, the graph of U can be viewed as the upper envelope of $O(nm)$ triangles, so its complexity is bounded by $O((nm)^2\alpha(nm))$, where α is the inverse of Ackermann's function. Moreover, U, clipped to within the rectangle R, can be computed using, e.g., divide-and-conquer in $\tilde{O}(n^2m^2)$ time; see [5, 13].
   A slightly more careful argument gives a better bound: In [9], it was shown that the complexity of the upper envelope of S piecewise-linear bivariate *concave* functions with a total of T facets is $O(ST\alpha(ST))$ and that it can be computed in time $\tilde{O}(ST)$. In our case $S = n$ and $T = O(nm)$, so we conclude that the complexity of U over R is at most $O(n^2m\alpha(n^2m))$ and that it can be constructed in time $\tilde{O}(n^2m)$.
4. Finally, we observe that the minimum of U (clipped to within R) is obtained at a vertex of U (which may occur at a point on the boundary of R), so we return the lowest vertex $v^* = (a^*, b^*, z^*)$, and the optimal segment with endpoints in τ_a and τ_b is $s^* := \overline{a^*b^*}$ and its corresponding distance is z^*.

Since there are $O(n^2m^2)$ pairs (τ_a, τ_b), we obtain the following lemma.

▶ **Lemma 3.1.** *Given a set $\mathcal{C}$ of n curves, where each curve consists of m points on the line, one can find the segment center of $\mathcal{C}$ in $\tilde{O}(n^4m^3)$ time.*

A solution for $d \geq 2$

We generalize our solution for $d = 1$ to any constant dimension $d \geq 2$. We consider the arrangement of hyperplanes obtained by drawing, through each point of each curve in $\mathcal{C}$, $2\binom{d}{2}$ hyperplanes, as in Section 2.2. Now, for each pair of cells (τ_a, τ_b) of the arrangement, we compute the optimal segment $s = \overline{ab}$ with $a \in \tau_a$ and $b \in \tau_b$. The process is repeated for every pair of cells.

For a fixed pair of cells (τ_a, τ_b) we do the following. For each curve $C_i \in \mathcal{C}$, we compute the lower envelope S_i of the functions $f_1, \ldots, f_{m-1}$ which correspond to hyperplanes in $\mathbb{R}^{2d+1}$. The complexity of S_i is $O(m^d)$ [11] and the graph can be viewed as a union of $O(m^d)$ pairwise non-overlapping simplices, see Section 2.2. Finally, we compute the upper envelope U of $S_1, \ldots, S_n$. We may think of U as the upper envelope of $O(nm^d)$ $2d$-simplices in $\mathbb{R}^{2d+1}$, so its complexity is $O((nm^d)^{2d}\alpha(nm^d))$ [13] and all its vertices can be computed in $\tilde{O}(n^{2d}m^{2d^2})$ expected time [1].

Since there are $O((nm)^{2d})$ pairs (τ_a, τ_b) we obtain the following lemma.

▶ **Lemma 3.2.** *Given a set $\mathcal{C}$ of n curves in $\mathbb{R}^d$, for $d \geq 2$, where each curve consists of m points, one can find the segment center of $\mathcal{C}$ in $\tilde{O}(n^{4d}m^{2d^2+2d})$ time.*

3.2 Second solution

We describe an alternative solution, which is more efficient when m is significantly smaller than n. Again, we first consider the one-dimensional version of the problem.

The case of $d = 1$

Let C_i be a curve in $\mathcal{C}$, and let $\tau = \langle \tau_1, \ldots, \tau_{m+1} \rangle$ be the partition of the line into intervals, induced by the points of C_i. For each pair of intervals (τ_a, τ_b), we define the bivariate functions $f_1, \ldots, f_{m-1}$ as in Section 2.2 and compute their lower envelope S_{τ_a,τ_b} over the

rectangle $R \coloneqq \tau_a \times \tau_b$. Recall that S_{τ_a,τ_b} is a piecewise-linear function of complexity $O(m)$, and that for any segment $s = \overline{ab}$ with $a \in \tau_a$ and $b \in \tau_b$, the dynamic time warping distance between C_i and s is $S_{\tau_a,\tau_b}(a,b)$.

Let S_i be the function representing the dynamic time warping distance from s to C_i, for an arbitrary segment $s = \overline{ab}$. Namely, $S_i(a,b) \coloneqq S_{\tau_a,\tau_b}(a,b)$, whenever $a \in \tau_a$ and $b \in \tau_b$. Then S_i is a piecewise-linear function of complexity $O(m^3)$.

We compute the segment center of $\mathcal{C}$ in three steps. In the first step, we compute the functions S_i, for all curves $C_i \in \mathcal{C}$. In the second step, we compute the upper envelope U of these n functions, whose complexity is bounded by $O(K^2\alpha(K))$, where $K = O(nm^3)$ is the total complexity of the functions S_i. Finally, we return the lowest vertex $v^* = (a^*, b^*, z^*)$ of U. The segment $s^* \coloneqq \overline{a^*b^*}$ is the desired one and its corresponding distance is z^*. We obtain the following lemma.

▶ **Lemma 3.3.** *Given a set $\mathcal{C}$ of n curves, where each curve consists of m points on the line, one can find the segment center of $\mathcal{C}$ in $\tilde{O}(n^2m^6)$ time.*

A solution for $d \geq 2$

We generalize our solution for $d = 1$ to any constant dimension $d \geq 2$. Now, S_i is a combination, over $O(m^{2d})$ pairs of cells (τ_a, τ_b), of the functions S_{τ_a,τ_b}, each of complexity $O(m^d)$, so the complexity of S_i is $O(m^{3d})$. We compute U, the upper envelope of $S_1, \ldots, S_n$, whose complexity is $O((nm^{3d})^{2d}\alpha(nm^{3d}))$, in $\tilde{O}((nm^{3d})^{2d})$ time [13] (more precisely, it is sufficient to enumerate the vertices of U by an algorithm of [1]), and return the lowest vertex of U. We thus obtain the following lemma.

▶ **Lemma 3.4.** *Given a set $\mathcal{C}$ of n curves in $\mathbb{R}^d$, for $d \geq 2$, where each curve consists of m points, one can find the segment center of $\mathcal{C}$ in $\tilde{O}(n^{2d}m^{6d^2})$ time.*

The following theorem summarizes our segment center results.

▶ **Theorem 3.5.** *Given a set $\mathcal{C}$ of n curves in $\mathbb{R}^d$, for $d \geq 1$, where each curve consists of m points, one can find the segment center of $\mathcal{C}$ in time (i) $\tilde{O}(n^2m^6)$, if $d = 1$ and $m \leq n^{2/3}$, (ii) $\tilde{O}(n^3m^4)$, if $d = 1$ and $m \geq n^{2/3}$, (iii) $\tilde{O}(n^{2d}m^{6d^2})$, if $d \geq 2$ and $m \leq n^{\frac{1}{2d-1}}$, and (iv) $\tilde{O}(n^{4d}m^{2d^2+2d})$, if $d \geq 2$ and $m \geq n^{\frac{1}{2d-1}}$.*

4 Nearest Neighbor

Let $\mathcal{C} = \{C_1, \ldots, C_n\}$ be a set of n curves in $\mathbb{R}^d$, where each curve consists of m points. In this section, we consider the *nearest neighbor* problem for $\mathcal{C}$ under L_∞. That is, construct a data structure that, given a query segment s in $\mathbb{R}^d$, returns the curve $C \in \mathcal{C}$ that minimizes the dynamic time warping distance between s and a curve of $\mathcal{C}$ (if there is more than one such curve, return and arbitrary one). We begin by presenting a solution for the one-dimensional version, which we then generalize to higher dimensions.

A solution for $d = 1$

Let $\tau = \langle \tau_1, \ldots, \tau_{mn+1} \rangle$ be the partition of the line into intervals (and two halflines), induced by the nm points of the curves of $\mathcal{C}$ (after sorting them from left to right).

For each pair of intervals (τ_a, τ_b), we construct a data structure for segment queries $s = \overline{ab}$, such that $a \in \tau_a$ and $b \in \tau_b$. We now describe the data structure for a fixed pair (τ_a, τ_b).

As in Section 3.1, let S_i be the lower envelope computed for C_i, $i = 1, \ldots, n$. Recall that S_i is a piecewise-linear function of complexity $O(m)$, and the dynamic time warping distance between $s = \overline{ab}$ and C_i, where $a \in \tau_a$ and $b \in \tau_b$, is $S_i(a, b)$. Let L denote the lower envelope of $S_1, \ldots, S_n$. Since L is the lower envelope of $O(nm)$ facets, its complexity is clearly bounded by $O(n^2m^2\alpha(nm))$, where α is the inverse of Ackermann's function, and (as above) it can be computed in $\tilde{O}(n^2m^2)$ time; see [5, 13]. The envelope L constitutes a compact representation of the set of all answers to segment nearest neighbor queries over $\tau_a \times \tau_b$; that is, if S_i is the envelope determining the face of L lying above (a, b), then the curve of $\mathcal{C}$ nearest to $s = \overline{ab}$ is C_i and $d_{dtw}(s, C_i) = S_i(a, b) = L(a, b)$.

We thus preprocess the graph of L for vertical ray-shooting queries, i.e., given $s = \overline{ab}$, return the envelope S_i determining the face of L lying above (a, b). This can be done in $\tilde{O}(n^2m^2)$ time, after which a ray-shooting query can be answered in $O(\log(nm))$ time.

This completes the description of the data structure for the pair (τ_a, τ_b). Since we need $O((nm)^2)$ such data structures, we obtain the following theorem.

▶ **Theorem 4.1.** *Given a set $\mathcal{C}$ of n curves, where each curve consists of m points on the line, one can construct in $\tilde{O}(n^4m^4)$ time a data structure of size $\tilde{O}(n^4m^4))$ that supports nearest neighbor segment queries in $O(\log(nm))$ time.*

A solution for $d \geq 2$

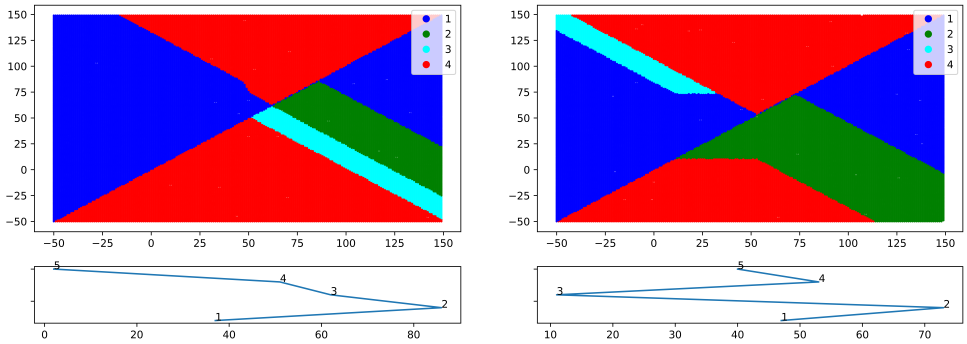
We generalize our solution for $d = 1$ to any constant dimension $d \geq 2$. As in Section 3.1, the complexity of each of the lower envelopes S_i is $O(m^d)$, and we compute their lower envelope L and preprocess it for logarithmic-time vertical ray-shooting queries in $\tilde{O}(n^{2d}m^{2d^2})$ time. Since we build $O((nm)^{2d})$ such data structures, we obtain the following theorem.

▶ **Theorem 4.2.** *Given a set $\mathcal{C}$ of n curves, where each curve consists of m points in $\mathbb{R}^d$, one can construct in $\tilde{O}(n^{4d}m^{2d^2+2d})$ time a data structure of size $\tilde{O}(n^{4d}m^{2d^2+2d})$ that supports nearest segment queries in $O(\log(nm))$ time.*

5 An Experimental Study

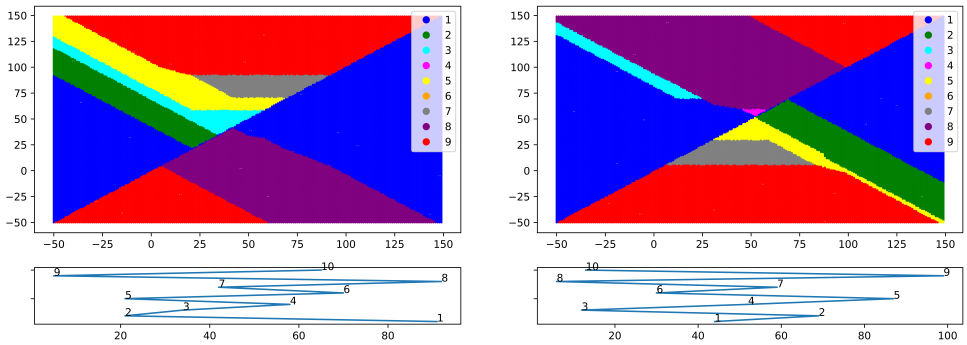
This section is related to the problem studied in Section 2.2: Given a curve C, preprocess it for segment distance queries. We focus on the one-dimensional version, namely, given a curve $C = \langle p_1, \ldots, p_m \rangle$ on the line, construct a data structure that facilitates queries of the form: given a segment $s = \overline{ab}$ on the line, return the dynamic time warping distance, $d_{dtw}(s, C)$, between s and C. For a query segment, the main difficulty in computing $d_{dtw}(s, C)$ is to determine the right partition of C into a prefix, which is matched to a, and a suffix, which is matched to b. Our goal is therefore to get a better idea of the relationship between the query segment and the right partition of C. In particular, is there an underlying geometric structure which is simple enough and may prove useful?

To this end, we have written a program that generates a random curve C consisting of m points in the range $[0, 100]$, where m is a parameter of the program. Next, the program generates a large number of random segments in the range $[-50, 150]$. For each such segment $s = \overline{ab}$, the program first finds the index j, $1 \leq j \leq m - 1$, such that, using the notation of Section 2.2, $d_{dtw}(s, C) = f_j(a, b) = \sum_{i=1}^{j} |p_i - a| + \sum_{i=j+1}^{m} |p_i - b|$, and then assigns the color j to the point (a, b) of the square $S := [-50, 150] \times [-50, 150]$. The output of the program is thus a coloring of a large number of points in S, see Figures 2 and 3; 300,000 random segments were generated for each diagram and thus 300,000 points are colored.



(a) $C = \langle 37, 86, 62, 51, 2 \rangle$. **(b)** $C = \langle 47, 73, 11, 53, 40 \rangle$.

Figure 2 The maps obtained from two runs with $m = 5$.



(a) $C = \langle 91, 21, 34, 58, 21, 70, 42, 92, 5, 65 \rangle$. **(b)** $C = \langle 44, 69, 12, 52, 87, 30, 59, 6, 99, 13 \rangle$.

Figure 3 The maps obtained from two runs with $m = 10$.

Consider, for example, Figure 2a, which shows the map that was obtained for the curve $C = \langle 37, 86, 62, 51, 2 \rangle$. Based on this map, it appears that for the query segment $s = (25, 60)$ the color that would be assigned to s is blue, which corresponds to $j = 1$, and indeed in this case $d_{dtw}(s, C) = f_1(25, 60)$.

Our experiments suggest that there may exist some underlying Voronoi-diagram–like structure, with some nice properties, that can be used for efficient handling of segment queries. It would therefore be interesting to continue the research in this direction. A moderately brave conjecture would be that the complexity of the resulting diagram is near linear in m; a much larger $O(m^3)$ bound follows from our argument in Section 2.2.

References

1 Pankaj K. Agarwal, Boris Aronov, and Micha Sharir. Computing envelopes in four dimensions with applications. *SIAM J. Comput.*, 26(6):1714–1732, 1997. doi:10.1137/S0097539794265724.

2 Richard Bellman and Robert E. Kalaba. On adaptive control processes. *IRE Trans. Automatic Control*, 4:1–9, 1959.

3 Timothy M. Chan. Optimal partition trees. *Discrete & Computational Geometry*, 47(4):661–690, 2012. doi:10.1007/s00454-012-9410-z.

4 Bernard Chazelle. An optimal convex hull algorithm in any fixed dimension. *Discrete & Computational Geometry*, 10:377–409, 1993. doi:10.1007/BF02573985.

5 Herbert Edelsbrunner, Leonidas J. Guibas, and Micha Sharir. The upper envelope of piecewise linear functions: Algorithms and applications. *Discrete & Computational Geometry*, 4:311–336, 1989. doi:10.1007/BF02187733.

6 Ioannis Z. Emiris and Ioannis Psarros. Products of Euclidean metrics and applications to proximity questions among curves. In *34th International Symposium on Computational Geometry, SoCG 2018, June 11-14, 2018, Budapest, Hungary*, volume 99 of *LIPIcs*, pages 37:1–37:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.SoCG.2018.37.

7 Arnold Filtser, Omrit Filtser, and Matthew J. Katz. Approximate nearest neighbor for curves - simple, efficient, and deterministic. In *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 48:1–48:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ICALP.2020.48.

8 Omer Gold and Micha Sharir. Dynamic time warping and geometric edit distance: Breaking the quadratic barrier. *ACM Trans. Algorithms*, 14(4):50:1–50:17, 2018. doi:10.1145/3230734.

9 Daniel P. Huttenlocher, Klara Kedem, and Micha Sharir. The upper envelope of voronoi surfaces and its applications. *Discrete & Computational Geometry*, 9:267–291, 1993. doi:10.1007/BF02189323.

10 Jiří Matoušek and Otfried Schwarzkopf. On ray shooting in convex polytopes. *Discrete & Computational Geometry*, 10:215–232, 1993. doi:10.1007/BF02573975.

11 P. McMullen. The maximum numbers of faces of a convex polytope. *Mathematika*, 17:179–184, 1970. doi:10.1112/S0025579300002850.

12 Meinard Müller. *Information Retrieval for Music and Motion*. Springer, 2007. doi:10.1007/978-3-540-74048-3.

13 János Pach and Micha Sharir. The upper envelope of piecewise linear functions and the boundary of a region enclosed by convex plates: Combinatorial analysis. *Discrete & Computational Geometry*, 4:291–309, 1989. doi:10.1007/BF02187732.