Computing Continuous Dynamic Time Warping of Time Series in Polynomial Time

Kevin Buchin
Department of Computer Science, TU Dortmund, Germany

André Nusser
BARC, University of Copenhagen, Denmark

Sampson Wong
School of Computer Science, University of Sydney, Australia

Abstract

Dynamic Time Warping is arguably the most popular similarity measure for time series, where we define a time series to be a one-dimensional polygonal curve. The drawback of Dynamic Time Warping is that it is sensitive to the sampling rate of the time series. The Fréchet distance is an alternative that has gained popularity, however, its drawback is that it is sensitive to outliers.

Continuous Dynamic Time Warping (CDTW) is a recently proposed alternative that does not exhibit the aforementioned drawbacks. CDTW combines the continuous nature of the Fréchet distance with the summation of Dynamic Time Warping, resulting in a similarity measure that is robust to sampling rate and to outliers. In a recent experimental work of Brankovic et al., it was demonstrated that clustering under CDTW avoids the unwanted artifacts that appear when clustering under Dynamic Time Warping and under the Fréchet distance. Despite its advantages, the major shortcoming of CDTW is that there is no exact algorithm for computing CDTW, in polynomial time or otherwise.

In this work, we present the first exact algorithm for computing CDTW of one-dimensional curves. Our algorithm runs in time $\mathcal{O}(n^5)$ for a pair of one-dimensional curves, each with complexity at most n. In our algorithm, we propagate continuous functions in the dynamic program for CDTW, where the main difficulty lies in bounding the complexity of the functions. We believe that our result is an important first step towards CDTW becoming a practical similarity measure between curves.

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

Keywords and phrases Computational Geometry, Curve Similarity, Fréchet distance, Dynamic Time Warping, Continuous Dynamic Time Warping

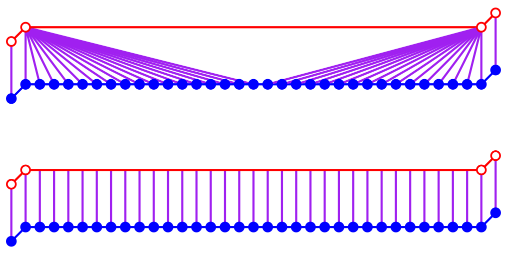
Digital Object Identifier 10.4230/LIPIcs.SoCG.2022.22

Related Version *Full Version*: `https://arxiv.org/abs/2203.04531`

Funding *André Nusser*: Part of this research was conducted while the author was at Saarbrücken Graduate School of Computer Science and Max Planck Institute for Informatics. The author is supported by the VILLUM Foundation grant 16582.

1 Introduction

Time series data arises from many sources, such as financial markets [39], seismology [43], electrocardiography [5] and epidemiology [7]. Domain-specific questions can often be answered by analysing these time series. A common way of analysing time series is by finding similarities. Computing similarities is also a fundamental building block for other analyses, such as clustering, classification, or simplification. There are numerous similarity measures considered in literature [4, 19, 23, 26, 37, 40], many of which are application dependent.


(a) Top: The optimal alignment under a discrete similarity measure, e.g. DTW. Bottom: The optimal alignment under a continuous similarity measure.

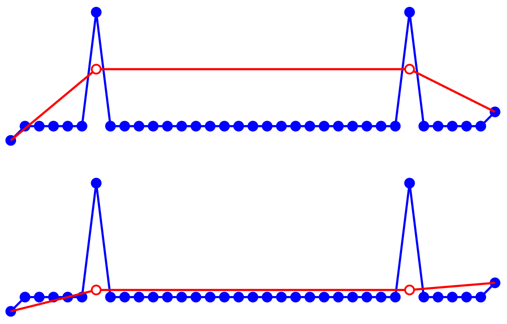

(b) Top: The optimal simplification under a bottleneck measure, e.g., Fréchet distance. Bottom: The optimal simplification under a summation-based similarity measure.

Figure 1 Issues with discrete (left) and bottleneck (right) measures as opposed to continuous, summed measures.

Dynamic Time Warping (DTW) is arguably the most popular similarity measure for time series, and is widely used across multiple communities [2, 24, 30, 32, 33, 34, 38, 41, 42]. Under DTW, a minimum cost discrete alignment is computed between a pair of time series. A discrete alignment is a sequence of pairs of points, subject to the following four conditions: *(i)* the first pair is the first sample from both time series, *(ii)* the last pair is the last sample from both time series, *(iii)* each sample must appear in some pair in the alignment, and *(iv)* the alignment must be a monotonically increasing sequence for both time series. The cost of a discrete alignment, under DTW, is the sum of the distances between aligned points. A drawback of a similarity measure with a discrete alignment is that it is sensitive to the sampling rates of the time series. As such, DTW is a poor measure of similarity between a time series with a high sampling rate and a time series with a low sampling rate. For such cases, it is more appropriate to use a similarity measure with a continuous alignment. In Figure 1a, we provide a visual comparison of a discrete alignment versus a continuous alignment, for time series with vastly different sampling rates.

The Fréchet distance is a similarity measure that has gained popularity, especially in the theory community [3, 13, 20, 36]. To apply the Fréchet distance to a time series, we linearly interpolate between sampled points to obtain a continuous one-dimensional polygonal curve. Under the Fréchet distance, a minimum cost continuous alignment is computed between the pair of curves. A continuous alignment is a simultaneous traversal of the pair of curves that satisfies the same four conditions as previously stated for DTW. The cost of a continuous alignment, under the Fréchet distance, is the maximum distance between a pair of points in the alignment. The Fréchet distance is a bottleneck measure in that it only measures the maximum distance between aligned points. As a result, the drawback of the Fréchet distance is that it is sensitive to outliers. For such cases, a summation-based similarity measure is significantly more robust. In Figure 1b, we illustrate a high complexity curve, and its low complexity "simplification" that is the most similar to the original curve, under either a bottleneck or summation similarity measure. The simplified curve under the Fréchet distance is sensitive to and drawn towards its outlier points.

Continuous Dynamic Time Warping (CDTW) is a recently proposed alternative that does not exhibit the aforementioned drawbacks. It obtains the best of both worlds by combining the continuous nature of the Fréchet distance with the summation of DTW. CDTW was first introduced by Buchin [17], where it was referred to as the average Fréchet distance.

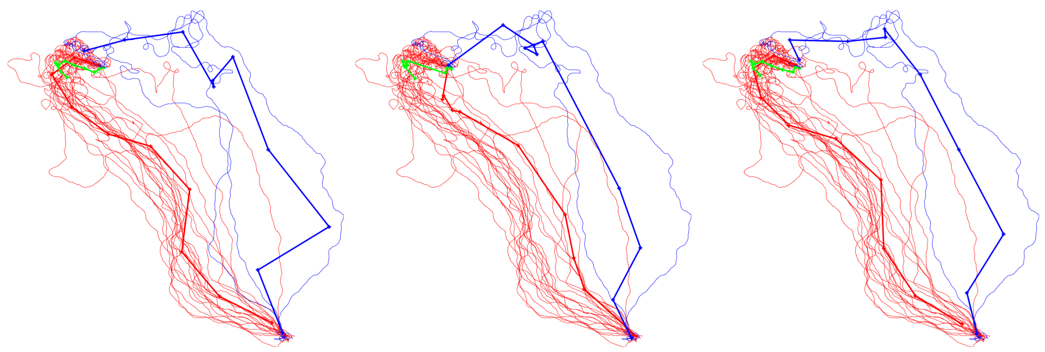


Figure 2 Clustering of the c17 pigeon's trajectories under the DTW (left), Fréchet (middle), and CDTW (right) distances. Figures were provided by the authors of [9].

CDTW has also been referred to as the summed, or integral, Fréchet distance. CDTW is similar to the Fréchet distance in that a minimum cost continuous alignment is computed between the pair of curves. The cost of a continuous alignment, under CDTW, is the integral of the distances between pairs of points in the alignment. We provide a formal definition in Section 2. Other definitions were also given under the name CDTW [22, 35], see Section 1.1.

Compared to existing popular similarity measures, CDTW is robust to both the sampling rate of the time series and to its outliers. CDTW has been used in applications where this robustness is desirable. In Brakatsoulas et al. [8], the authors applied CDTW to map-matching of vehicular location data. The authors highlight two common errors in real-life vehicular data, that is, measurement errors and sampling errors. Measurement errors result in outliers whereas sampling errors cause discrepancies in sampling rates between input curves. Their experiments show an improvement in map-matching when using CDTW instead of the Fréchet distance. In a recent paper, Brankovic et al. [9] applied CDTW to clustering of bird migration data and handwritten character data. The authors used (k, ℓ)-center and medians clustering, where each of the k clusters has a (representative) center curve of complexity at most ℓ. Low complexity center curves are used to avoid overfitting. Compared to DTW and the Fréchet distance, Brankovic et al. [9] demonstrated that clustering under CDTW produced centers that were more visually similar to the expected center curve. Under DTW, the clustering quality deteriorated for small values of ℓ, whereas under the Fréchet distance, the clustering quality deteriorated in the presence of outliers.

Brankovic et al.'s [9] clustering of a pigeon data set [28] is shown in Figure 2. The Fréchet distance is paired with the center objective, whereas DTW and CDTW are paired with the medians objective. Under DTW (left), the discretisation artifacts are visible. The blue center curve is jagged and visually dissimilar to its associated input curves. Under the Fréchet distance (middle), the shortcoming of the bottleneck measure and objective is visible. The red center curve fails to capture the shape of its associated input curves, in particular, it misses the top-left "hook" appearing in its associated curves. Under CDTW (right), the center curves are smooth and visually similar to their associated curves.

Despite its advantages, the shortcoming of CDTW is that there is no exact algorithm for computing it, in polynomial time or otherwise. Heuristics were used to compute CDTW in the map-matching [8] and clustering [9] experiments. Maheshwari et al. [27] provided a $(1 + \varepsilon)$-approximation algorithm for CDTW in $\mathcal{O}(\zeta^4 n^3/\varepsilon^2)$ time, for curves of complexity n and spread ζ, where the spread is the ratio between the maximum and minimum interpoint distances. Existing heuristic and approximation methods [8, 9, 27] use a sampled grid on top of the dynamic program for CDTW, introducing an inherent error that depends on the fineness of the sampled grid, which is reflected in the dependency on ζ in [27].

In this work, we present the first exact algorithm for computing CDTW for one-dimensional curves. Our algorithm runs in time $\mathcal{O}(n^5)$ for a pair of one-dimensional curves, each with complexity at most n. Unlike previous approaches, we avoid using a sampled grid and instead devise a propagation method that solves the dynamic program for CDTW exactly. In our propagation method, the main difficulty lies in bounding the total complexity of our propagated functions. Showing that CDTW can be computed in polynomial time fosters hope for faster polynomial time algorithms, which would add CDTW to the list of practical similarity measures for curves.

1.1 Related work

Algorithms for computing popular similarity measures, such as DTW and the Fréchet distance, are well studied. Vintsyuk [41] proposed Dynamic Time Warping as a similarity measure for time series, and provided a simple dynamic programming algorithm for computing the DTW distance that runs in $\mathcal{O}(n^2)$ time, see also [6]. Gold and Sharir [24] improved the upper bound for computing DTW to $\mathcal{O}(n^2/\log\log n)$. For the Fréchet distance, Alt and Godau [3] proposed an $\mathcal{O}(n^2 \log n)$ time algorithm for computing the Fréchet distance between a pair of curves. Buchin et al. [13] improved the upper bound for computing the Fréchet distance to $\mathcal{O}(n^2\sqrt{\log n}(\log\log n)^{3/2})$. Assuming SETH, it has been shown that there is no strongly subquadratic time algorithm for computing the Fréchet distance or DTW [1, 10, 11, 12, 16].

Our definition of CDTW was originally proposed by Buchin [17], and has since been used in several experimental works [8, 9]. We give Buchin's [17] definition formally in Section 2. Other definitions under the name CDTW have also been considered. We briefly describe the main difference between these definitions and the one used in this paper.

To the best of our knowledge, the first continuous version of DTW was by Serra and Berthod [35]. The same definition was later used by Munich and Perona [31]. Although a continuous curve is used in their definition, the cost of the matching is still a discrete summation of distances to sampled points. Our definition uses a continuous summation (i.e. integration) of distances between all points on the curves, and therefore, is more robust to discrepancies in sampling rate. Efrat et al. [22] proposed a continuous version of DTW that uses integration. However, their integral is defined in a significantly different way to ours. Their formulation minimises the change of the alignment and not the distance between aligned points. Thus, their measure is translational invariant and designed to compare the shapes of curves irrespective of their absolute positions in space.

2 Preliminaries

We use $[n]$ to denote the set $\{1, \ldots, n\}$. To continuously measure the similarity of time series, we linearly interpolate between sampled points to obtain a one-dimensional polygonal curve. A one-dimensional polygonal curve P of complexity n is given by a sequence of vertices, $p_1, \ldots, p_n \in \mathbb{R}$, connected in order by line segments. Furthermore, let $||\cdot||$ be the norm in the one-dimensional space $\mathbb{R}$. In higher dimensions, the Euclidean $\mathcal{L}_2$ norm is the most commonly used norm, but other norms such as $\mathcal{L}_1$ and $\mathcal{L}_\infty$ may be used.

Consider a pair of one-dimensional polygonal curves $P = p_1, \ldots, p_n$ and $Q = q_1, \ldots, q_m$. Let $\Delta(n, m)$ be the set of all sequences of pairs of integers $(x_1, y_1), \ldots, (x_k, y_k)$ satisfying $(x_1, y_1) = (1, 1)$, $(x_k, y_k) = (n, m)$ and $(x_{i+1}, y_{i+1}) \in \{(x_i + 1, y_i), (x_i, y_i + 1), (x_i + 1, y_i + 1)\}$.

The DTW distance between P and Q is defined as

$$d_{DTW}(P,Q) = \min_{\alpha\in\Delta(n,m)} \sum_{(x,y)\in\alpha} ||p_x - q_y||.$$

The discrete Fréchet distance between P and Q is defined as

$$d_{dF}(P,Q) = \min_{\alpha\in\Delta(n,m)} \max_{(x,y)\in\alpha} ||p_x - q_y||.$$

Let p and q be the total arc lengths of P and Q respectively. Define the parametrised curve $\{P(z) : z \in [0,p]\}$ to be the one-dimensional curve P parametrised by its arc length. In other words, $P(z)$ is a piecewise linear function so that the arc length of the subcurve from $P(0)$ to $P(z)$ is z. Define $\{Q(z) : z \in [0,q]\}$ analogously. Let $\Gamma(p)$ be the set of all continuous and non-decreasing functions $\alpha : [0,1] \to [0,p]$ satisfying $\alpha(0) = 0$ and $\alpha(1) = p$. Let $\Gamma(p,q) = \Gamma(p) \times \Gamma(q)$. The continuous Fréchet distance between P and Q is defined as

$$d_F(P,Q) = \inf_{(\alpha,\beta)\in\Gamma(p,q)} \max_{z\in[0,1]} ||P(\alpha(z)) - Q(\beta(z))||,$$

The CDTW distance between P and Q is defined as

$$d_{CDTW}(P,Q) = \inf_{(\alpha,\beta)\in\Gamma(p,q)} \int_0^1 ||P(\alpha(z)) - Q(\beta(z))|| \cdot ||\alpha'(z) + \beta'(z)|| \cdot dz.$$

For the definition of CDTW, we additionally require that α and β are differentiable. The original intuition behind $d_{CDTW}(P,Q)$ is that it is a line integral in the parameter space, which we will define in Section 2.1. The term $||\alpha'(z) + \beta'(z)||$ implies that we are using the $\mathcal{L}_1$ metric in the parameter space, but other norms have also been considered [26, 27].

2.1 Parameter space under CDTW

The parameter space under CDTW is analogous to the free space diagram under the continuous Fréchet distance. Similar to previous work [17, 26, 27], we transform the problem of computing CDTW into the problem of computing a line integral in the parameter space.

Recall that the total arc lengths of P and Q are p and q respectively. The parameter space is defined to be the rectangular region $R = [0,p] \times [0,q]$ in $\mathbb{R}^2$. The region is imbued with a metric $||\cdot||_R$. The $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_\infty$ norms have all been considered, but $\mathcal{L}_1$ is the preferred metric as it is the easiest to work with [26, 27]. At every point $(x,y) \in R$ we define the height of the point to be $h(x,y) = ||P(x) - Q(y)||$.

Next, we provide the line integral formulation of d_{CDTW}, which is the original motivation behind its definition. To make our line integral easier to work with, we parametrise our line integral path γ in terms of its $\mathcal{L}_1$ arc length in R. The following lemma is a consequence of Section 6.2 in [17]. We provide a proof sketch of the result for the sake of self-containment.

▶ **Lemma 1.**

$$d_{CDTW}(P,Q) = \inf_{\gamma\in\Psi(p,q)} \int_0^{p+q} h(\gamma(z)) \cdot dz,$$

where $\Psi(p,q)$ is the set of all functions $\gamma : [0,p+q] \to R$ satisfying $\gamma(0) = (0,0)$, $\gamma(p+q) = (p,q)$, γ is differentiable and non-decreasing in both x- and y-coordinates, and $||\gamma'(z)||_R = 1$.

Proof (Sketch). We provide a full proof in [15]. We summarise the main steps. Recall that the formula for CDTW is $\inf_{(\alpha,\beta)\in\Gamma(p,q)} \int_0^1 ||P(\alpha(z)) - Q(\beta(z))|| \cdot ||\alpha'(z) + \beta'(z)|| \cdot dz$. If we define $\gamma(t) = (\alpha(t), \beta(t))$, then the first term of the integrand is equal to $h(\gamma(t))$. Next, we reparametrise $\gamma(t)$ in terms of its $\mathcal{L}_1$ arc length in R. For our reparametrised γ, we get $||\alpha'(z) + \beta'(z)|| = 1$, so the second term of the integrand is equal to 1. Finally, we prove that the parameter z ranges from 0 to $p+q$, and note that $\gamma(0) = (0,0)$, $\gamma(p+q) = (p,q)$, γ is differentiable and non-decreasing, and $||\gamma'(z)||_R = 1$. This gives us the stated formula. ◀

2.2 Properties of the parameter space

Before providing the algorithm for minimising our line integral, we first provide some structural insights of our parameter space $R = [0,p] \times [0,q]$. Recall that $P : [0,p] \to \mathbb{R}$ maps points on the x-axis of R to points on the one-dimensional curve P, and analogously for Q and the y-axis. Hence, each point $(x,y) \in R$ is associated with a pair of points $P(x)$ and $Q(y)$, so that the height function $h(x,y) = ||P(x) - Q(y)||$ is simply the distance between the associated pair of points. Divide the x-axis of R into $n-1$ segments that are associated with the $n-1$ segments $p_1p_2, \ldots, p_{n-1}p_n$ of P. Divide the y-axis of R into $m-1$ segments analogously. In this way, we divide R into $(n-1)(m-1)$ cells, which we label as follows:

▶ **Definition 2** (cell). *Cell (i,j) is the region of the parameter space associated with segment p_ip_{i+1} on the x-axis, and q_jq_{j+1} on the y-axis, where $i \in [n-1]$ and $j \in [m-1]$.*

For points (x,y) restricted to a single cell (i,j), the functions $P(x)$ and $Q(y)$ are linear. Hence, $P(x) - Q(y)$ is also linear, so $h(x,y) = ||P(x) - Q(y)||$ is a piecewise linear surface with at most two pieces. If $h(x,y)$ consists of two linear surface pieces, the boundary of these two pieces is along a segment where $h(x,y) = 0$. Since we are working with one-dimensional curves, we have two cases for the relative directions of the vectors $\overrightarrow{p_ip_{i+1}}$ and $\overrightarrow{q_jq_{j+1}}$. If the vectors are in the same direction, since $\overrightarrow{p_ip_{i+1}}$ and $\overrightarrow{q_jq_{j+1}}$ are both parametrised by their arc lengths, they must be travelling in the same direction and at the same rate. Therefore, the line satisfying $h(x,y) = 0$ has slope 1 in R. Using a similar argument, if $\overrightarrow{p_ip_{i+1}}$ and $\overrightarrow{q_jq_{j+1}}$ are in opposite direction, then the line satisfying $h(x,y) = 0$ has slope -1 in R.

The line with zero height and slope 1 will play an important role in our algorithm. We call these lines valleys. If a path γ travels along a valley, the line integral accumulates zero cost as long as it remains on the valley, since the valley has zero height.

▶ **Definition 3** (valley). *In a cell, the set of points (x,y) satisfying $h(x,y) = 0$ forms a line, moreover, the line has slope 1 or -1. If the line has slope 1, we call it a valley.*

3 Algorithm

Our approach is a dynamic programming algorithm over the cells in the parameter space, which we defined in Section 2.1. To the best of our knowledge, all the existing approximation algorithms and heuristics [8, 9, 26] use a dynamic programming approach, or simply reduce the problem to a shortest path computation [27]. Next, we highlight the key difference between our approach and previous approaches.

In previous algorithms, sampling is used along cell boundaries to obtain a discrete set of grid points. Then, the optimal path between the discrete set of grid points is computed. The shortcoming of previous approaches is that an inherent error is introduced by the grid points, where the error depends on the fineness of the grid that is used.

In our algorithm, we consider all points along cell boundaries, not just a discrete subset. However, this introduces a challenge whereby we need to compute optimal paths between continuous segments of points. To overcome this obstacle, we devise a new method of propagating continuous functions across a cell. The main difficulty in analysing the running time of our algorithm lies in bounding the total complexity of the propagated continuous functions, across all cells in the dynamic program.

Our improvement over previous approaches is in many ways similar to previous algorithms for the weighted region problem [29], and the partial curve matching problem [14]. In all three problems, a line integral is minimised over a given terrain, and an optimal path is computed instead of relying on a sampled grid. However, our problem differs from that of [29] and [14] in two important ways. First, in both [29] and [14], the terrain is a piecewise constant function, whereas in our problem, the terrain is a piecewise linear function. Second, our main difficulty is bounding the complexity of the propagated functions. In [29], a different technique is used that does not propagate functions. In [14], the propagated functions are concave, piecewise linear and their complexities remain relatively low. In our algorithm, the propagated functions are piecewise quadratic and their complexities increase at a much higher, albeit bounded, rate.

3.1 Dynamic program

Our dynamic program is performed with respect to the following cost function.

▶ **Definition 4** (cost function). *Let $(x, y) \in R$, we define*

$$cost(x, y) = \inf_{\gamma \in \Psi(x,y)} \int_0^{x+y} h(\gamma(z)) \cdot dz.$$

Recall from Lemma 1 that $d_{CDTW}(P, Q) = \inf_{\gamma \in \Psi(p,q)} \int_0^{p+q} h(\gamma(z)) \cdot dz$, which implies that $cost(p, q) = d_{CDTW}(P, Q)$. Another way of interpreting Definition 4 is that $cost(x, y)$ is equal to $d_{CDTW}(P_x, Q_y)$, where P_x is the subcurve from $P(0)$ to $P(x)$, and Q_y is the subcurve from $Q(0)$ to $Q(y)$.

Recall from Section 2.2 that the parameter space is divided into $(n-1)(m-1)$ cells. Our dynamic program solves cells one at a time, starting from the bottom left cell and working towards the top right cell. A cell is considered solved if we have computed the cost of every point on the boundary of the cell. Once we solve the top right cell of R, we obtain the cost of the top right corner of R, which is $cost(p, q) = d_{CDTW}(P, Q)$, and we are done.

In the base case, we compute the cost of all points lying on the lines $x = 0$ and $y = 0$. Note that if $x = 0$ or $y = 0$, then the function $cost(x, y)$ is simply a function in terms of y or x respectively. In general, the function along any cell boundary – top, bottom, left or right – is a univariate function in terms of either x or y. We call these boundary cost functions.

▶ **Definition 5** (boundary cost function). *A boundary cost function is $cost(x, y)$, but restricted to a top, bottom, left or right boundary of a cell. If it is restricted to a top or bottom (resp. left or right) boundary, the boundary cost function is univariate in terms of x (resp. y).*

In the propagation step, we use induction to solve the cell (i, j) for all $1 \le i \le n-1$ and $1 \le j \le m-1$. We assume the base case. We also assume as an inductive hypothesis that, if $i \ge 2$, then the cell $(i-1, j)$ is already solved, and if $j \ge 2$, then the cell $(i, j-1)$ is already solved. Our assumptions ensure that we receive as input the boundary cost function along the bottom and left boundaries of the cell (i, j). In other words, we use the boundary cost functions along the input boundaries to compute the boundary cost functions along the output boundaries.

▶ **Definition 6** (input/output boundary). *The input boundaries of a cell are its bottom and left boundaries. The output boundaries of a cell are its top and right boundaries.*

We provide details of the base case in Section 3.2, and the propagation step in Section 3.3.

3.2 Base case

The base case is to compute the cost of all points along the x-axis. The y-axis can be handled analogously. Recall that $cost(x, y) = \inf_{\gamma\in\Psi(x,y)} \int_0^{x+y} h(\gamma(z)) \cdot dz$. Therefore, for points $(x, 0)$ on the x-axis, we have $cost(x, 0) = \inf_{\gamma\in\Psi(x,0)} \int_0^{x} h(\gamma(z)) \cdot dz$. Since $\gamma(z)$ is non-decreasing in x- and y-coordinates, and $||\gamma'(z)|| = 1$, we must have that $\gamma'(z) = (1, 0)$. By integrating from 0 to z, we get $\gamma(z) = (z, 0)$, which implies that $cost(x, 0) = \int_0^x h(z, 0) \cdot dz$.

Consider, for $1 \leq i \leq n-1$, the bottom boundary of the cell $(i, 1)$. The height function $h(z)$ is a piecewise linear function with at most two pieces, so its integral $cost(x, 0) = \int_0^x h(z, 0) \cdot dz$ is a continuous piecewise quadratic function with at most two pieces. Similarly, since the height function along $x = 0$ is a piecewise linear function with at most $2(n - 1)$ pieces, the boundary cost function along $x = 0$ is a continuous piecewise quadratic function with at most $2(n - 1)$ pieces. For boundaries not necessarily on the x- or y-axis, we claim that the boundary cost function is still a continuous piecewise-quadratic function.

▶ **Lemma 7.** *The boundary cost function is a continuous piecewise-quadratic function.*

We defer the proof of Lemma 7 to the full version of this paper [15]. Although the boundary cost function has at most two pieces for cell boundaries on the x- or y-axis, in the general case it may have more than two pieces. As previously stated, the main difficulty in bounding our running time analysis in Section 3.4 is to bound complexities of the boundary cost functions.

3.3 Propagation step

First, we define optimal paths in the parameter space. We use optimal paths to propagate the boundary cost functions across cells in the parameter space. Note that the second part of Definition 8 is a technical detail to ensure the uniqueness of optimal paths. Intuitively, the optimal path from s to t is the path minimising the path integral, and if there are multiple such paths, the optimal path is the one with maximum y-coordinate.

▶ **Definition 8** (optimal path). *Given $t = (x_t, y_t) \in R$, its optimal path is a path $\gamma \in \Psi(x_t, y_t)$ minimising the integral $\int_0^{x_t+y_t} h(\gamma(z)) \cdot dz$. If there are multiple such curves that minimise the integral, the optimal path is the one with maximum y-coordinate (or formally, the one with maximum integral of its y-coordinates).*

Suppose t is on the output boundary of the cell (i, j). Consider the optimal path γ that starts at $(0, 0)$ and ends at t. Let s be the first point where γ enters the cell (i, j). We consider the subpath from s to t, which is entirely contained in the cell (i, j). In the next lemma, we show that the shape of the subpath from s to t is restricted, in particular, there are only three types of paths that we need to consider.

▶ **Lemma 9.** *Let t be a point on the output boundary of a cell. Let s be the first point where the optimal path to t enters the cell. There are only three types of paths from s to t:*

(A) *The segments of the cell are in opposite directions. Then all paths between s and t have the same cost.*

(B) *The segments of the cell are in the same direction and the optimal path travels towards the valley, then along the valley, then away from the valley.*

(C) *The segments of the cell are in the same direction and the optimal path travels towards the valley, then away from the valley.*

For an illustration of these three types of paths, see Figure 3.

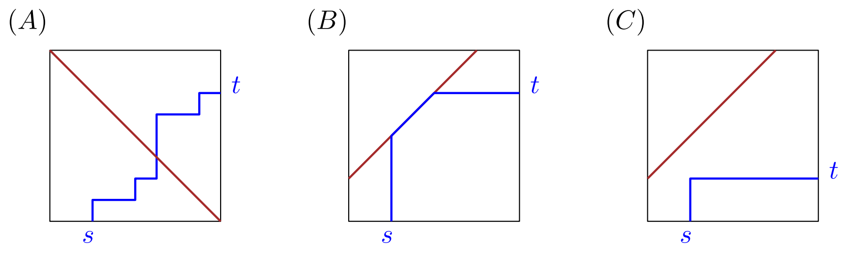


Figure 3 The three types of optimal paths through a cell.

Proof (Sketch). A similar proof can be found in Lemma 4 of Maheswari et al. [27]. Nonetheless, due to slight differences, we provide a full proof in the full version [15]. Specifically, we consider one-dimensional curves, and use the $\mathcal{L}_1$ norm in parameter space to obtain a significantly stronger statement for type (A) paths.

We summarise the main steps here. Define γ_1 to be an optimal path to s, followed by any path from s to t. Define γ_2 to be an optimal path to s, followed by either a type (A), (B) or (C) path from s to t. If the segments are in opposite directions, we use a type (A) path, whereas if the segments are in the same direction, we use either a type (B) or type (C) path. The main step is to show that $h(\gamma_1(z)) \geq h(\gamma_2(z))$, as this would imply that γ_2 is an optimal path from s to t. In fact, if the segments are in the opposite directions, we get that $h(\gamma_1(z)) = h(\gamma_2(z))$, implying that all type (A) paths from s to t have the same cost. ◀

We leverage Lemma 9 to propagate the boundary cost function from the input boundaries to the output boundaries of a cell. We provide an outline of our propagation procedure in one of the three cases, that is, for type (B) paths. These paths are the most interesting to analyse, and looking at this special case provides us with some intuition for the other cases. For type (B) paths, we compute the cost function along the output boundary in three consecutive steps. We first list the steps, then we describe the steps in detail.

1. We compute the cost function along the valley in a restricted sense.
2. We compute the cost function along the valley in general.
3. We compute the cost function along the output boundary.

In the first step, we restrict our attention only to paths that travel from the input boundary towards the valley. This is the first segment in the type (B) path as defined in Lemma 9. We call this first segment a type $(B1)$ path, see Figure 4. Define the type $(B1)$ cost function to be the cost function along the valley if we can only use type $(B1)$ paths from the input boundary to the valley. The type $(B1)$ cost function is simply the cost function along the bottom or left boundary plus the integral of the height function along the type $(B1)$ path. The height function along the type $(B1)$ path is a linear function, so the integral is a quadratic function. To obtain the type $(B1)$ cost function, we add the quadratic function for the type $(B1)$ path to the cost function along an input boundary. We combine the type $(B1)$ cost functions along the bottom and the left boundaries by taking their lower envelope.

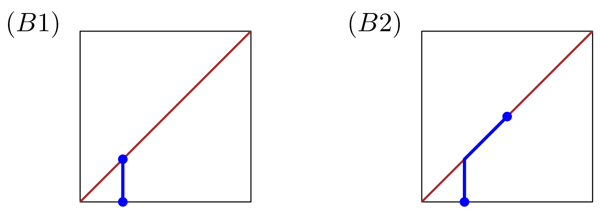


Figure 4 The type $(B1)$ and type $(B2)$ paths from the bottom boundary to the valley.

In the second step, we compute the cost function along the valley in general. It suffices to consider paths that travel from the input boundary towards the valley, and then travel along the valley. This path is the first two segments in a type (B) path as defined in Lemma 9. We call these first two segments a type $(B2)$ path, see Figure 4. Since the height function is zero along the valley, if we can reach a valley point with a particular cost with a type $(B1)$ path, then we can reach all points on the valley above and to the right of it with a type $(B2)$ path with the same cost. Therefore, the type $(B2)$ cost function is the cumulative minimum of the type $(B1)$ cost function, see Figure 5. Note that the type $(B2)$ cost function may have more quadratic pieces than the type $(B1)$ cost function. For example, in Figure 5, the type $(B2)$ cost function has twice as many quadratic pieces as the type $(B1)$ cost function, since each quadratic piece in the type $(B1)$ cost function splits into two quadratic pieces in the type $(B2)$ cost function – the original quadratic piece plus an additional horizontal piece.

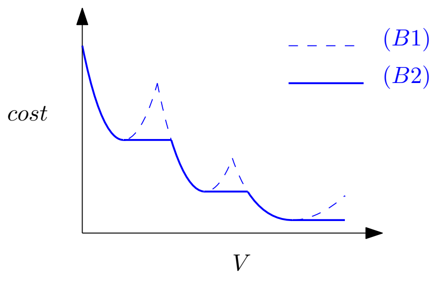


Figure 5 The type $(B2)$ cost function plotted over its position along the valley V. The type $(B2)$ cost function is the cumulative minimum of the type $(B1)$ cost function.

In the third step, we compute the cost function along the output boundary, given the type $(B2)$ cost function along the valley. A type (B) path is a type $(B2)$ path appended with a horizontal or vertical path from the valley to the boundary. The height function of the appended path is a linear function, so its integral is a quadratic function. We add this quadratic function to the type $(B2)$ function along the valley to obtain the output function. This completes the description of the propagation step in the type (B) paths case.

Using a similar approach, we can compute the cost function along the output boundary in the type (A) and type (C) paths as well. The propagation procedure differs slightly for each of the three path types, for details see the full version [15]. Recall that due to the second step of the type (B) propagation, each quadratic piece along the input boundary may propagate to up to two pieces along the output boundary. In general, we claim that each quadratic piece along the input boundary propagates to at most a constant number of pieces along the output boundary. Moreover, given a single input quadratic piece, this constant number of output quadratic pieces can be computed in constant time.

▶ **Lemma 10.** *Each quadratic piece in the input boundary cost function propagates to at most a constant number of pieces along the output boundary. Propagating a quadratic piece takes constant time.*

We prove Lemma 10 in the full version [15]. We can now state our propagation step in general. Divide the input boundaries into subsegments, so that for each subsegment, the cost function along that subsegment is a single quadratic piece. Apply Lemma 10 to a subsegment to compute in constant time a piecewise quadratic cost function along the output boundary. Apply this process to all subsegments to obtain a set of piecewise quadratic cost functions along the output boundary. Combine these cost functions by taking their lower envelope. Return this lower envelope as the boundary cost function along the output boundary. This completes the statement of our propagation step. Its correctness follows from construction.

3.4 Running time analysis

We start the section with a useful lemma. Essentially the same result is stated without proof as Observation 3.3 in [14]. For the sake of completeness, we provide a proof sketch.

▶ **Lemma 11.** *Let γ_1, γ_2 be two optimal paths. These paths cannot cross, i.e., there are no z_1, z_2 such that $\gamma_1(z_1)$ is below $\gamma_2(z_1)$ and $\gamma_1(z_2)$ is above $\gamma_2(z_2)$.*

Proof (Sketch). We provide a full proof in [15]. We summarise the main steps. Let u be the first point where a pair of optimal paths crosses. We show that the crossing paths, up to u, must have been identical, so the paths cannot cross at u. ◀

Define N to be the total number of quadratic pieces in the boundary cost functions over all boundaries of all cells. We will show that the running time of our algorithm is $\mathcal{O}(N)$.

▶ **Lemma 12.** *The running time of our dynamic programming algorithm is $\mathcal{O}(N)$.*

Proof. The running time of the dynamic program is dominated by the propagation step. Let $I_{i,j}$ denote the input boundaries of the cell (i, j). Let $|I_{i,j}|$ denote the number of quadratic functions in the input boundary cost function. By Lemma 10, each piece only propagates to a constant number of new pieces along the output boundary, and these pieces can be computed in constant time. The final piecewise quadratic function is the lower envelope of all the new pieces, of which there are $\mathcal{O}(|I_{i,j}|)$ many.

We use Lemma 11 to speed up the computation of the lower envelope, so that this step takes only $\mathcal{O}(|I_{i,j}|)$ time. Since optimal paths do not cross, it implies that the new pieces along the output boundary appear in the same order as their input pieces. We perform the propagation in order of the input pieces. We maintain the lower envelope of the new pieces in a stack. For each newly propagated piece, we remove the suffix that is dominated by the new piece and then add the new piece to the stack. Since each quadratic piece can be added to the stack at most once, and removed from the stack at most once, the entire operation takes $\mathcal{O}(|I_{i,j}|)$ time. Summing over all cells, we obtain an overall running time of $\mathcal{O}(N)$. ◀

Note that Lemma 12 does not yet guarantee that our algorithm runs in polynomial time as we additionally need to bound N by a polynomial. Lemma 10 is of limited help. The lemma states that each piece on the input boundary propagates to at most a constant number of pieces on the output boundary. Recall that in Section 3.3, we illustrated a type (B) path that resulted in an output boundary having twice as many quadratic pieces as its input boundary. The doubling occurred in the second step of the propagation of type (B) paths, see Figure 5. If this doubling behaviour were to occur for all our cells in our dynamic program, we would get up to $N = \Omega(2^{n+m})$ quadratic pieces in the worst case, where n and m are the complexities of the polygonal curves P and Q. To obtain a polynomial running time, we show that although this doubling behaviour may occur, it does not occur *too often*.

3.5 Bounding the cost function's complexity

Our bound comes in two parts. First, we subdivide the boundaries in the parameter space into subsegments and show, in Lemma 13, that there are $\mathcal{O}((n+m)^3)$ subsegments in total. Second, in Lemma 15, we show that each subsegment has at most $\mathcal{O}((n+m)^2)$ quadratic pieces. Putting this together in Theorem 16 gives $N = \mathcal{O}((n+m)^5)$.

We first define the $\mathcal{O}((n+m)^3)$ subsegments. The intuition behind the subsegments is that for any two points on the subsegment, the optimal path to either of those two points is structurally similar. We can deform one of the optimal paths to the other without passing through any cell corner, or any points where a valley meets a boundary.

Formally, define A_k to be the union of the input boundaries of the cells (i, j) such that $i + j = k$. Alternatively, A_k is the union of the output boundaries of the cells (i, j) such that $i + j + 1 = k$. Next, construct the partition $\mathcal{A}_k := \{A_{k,1}, A_{k,2}, \dots, A_{k,L}\}$ of A_k into subsegments. Define the subsegment $A_{k,\ell}$ to be the segment between the ℓ^{th} and $(\ell+1)^{th}$ *critical point* along A_k. We define a critical point to be either *(i)* a cell corner, *(ii)* a point where the valley meets the boundary, or *(iii)* a point where the optimal path switches from passing through a subsegment $A_{k-1,\ell'}$ to a different subsegment $A_{k-1,\ell''}$.

Let $|\mathcal{A}_k|$ denote the number of piecewise quadratic cost functions $A_{k,\ell}$ along A_k. Let $|A_{k,\ell}|$ denote the number of pieces in the piecewise quadratic cost function along the subsegment $A_{k,\ell}$. Thus, we can rewrite the total number of quadratic functions N as:

$$N = \sum_{k=2}^{n+m-1} \sum_{\ell=1}^{|\mathcal{A}_k|} |A_{k,\ell}|.$$

We first show that the number of subsegments $|\mathcal{A}_k|$ is bounded by $\mathcal{O}(k^2)$ and then proceed to show that $|A_{k,\ell}|$ is bounded by $\mathcal{O}(k^2)$ for all k, ℓ.

▶ **Lemma 13.** *For any $k \in [n+m]$, we have $|\mathcal{A}_k| \leq 2k^2$.*

Proof. We prove the lemma by induction. Since the cell $(1, 1)$ has at most one valley, and since the input boundary A_2 has one cell corner, we have $|\mathcal{A}_2| \leq 3$. For the inductive step, note that there are at most $2k$ cell corners on A_k, and there are at most k points where a valley meets a boundary on A_k. By the inductive hypothesis, there are at most $2(k-1)^2$ subsegments on A_{k-1}. And as optimal paths do not cross by Lemma 11, each subsegment of A_{k-1} contributes at most once to the optimal path switching from one subsegment to a different one on A_k. Thus, for $k \geq 3$, we obtain $|\mathcal{A}_k| \leq 2(k-1)^2 + 2k + k + 1 = 2k^2 - k + 3 \leq 2k^2$. ◀

Next, we show that $|A_{k,\ell}|$ is bounded by $\mathcal{O}(k^2)$ for all k, ℓ. We proceed by induction. Recall that, due to the third type of critical point, all optimal paths to $A_{k,\ell}$ pass through the same subsegment of A_{k-1}, namely $A_{k-1,\ell'}$ for some ℓ'. Our approach is to assume the inductive hypothesis for $|A_{k-1,\ell'}|$, and bound $|A_{k,\ell}|$ relative to $|A_{k-1,\ell'}|$. We already have a bound of this form, specifically, Lemma 10 implies that $|A_{k,\ell}| \leq c \cdot |A_{k-1,\ell'}|$, for some constant $c > 1$. Unfortunately, this bound does not rule out an exponential growth in the cost function complexity. We instead prove the following improved bound:

▶ **Lemma 14.** *Let $|A_{k,\ell}|$ be a subsegment on A_k, and suppose all optimal paths to $|A_{k,\ell}|$ pass through subsegment $|A_{k-1,\ell'}|$ on A_{k-1}. Then*

$$\begin{aligned} |A_{k,\ell}| &\leq |A_{k-1,\ell'}| + D(A_{k-1,\ell'}), \\ D(A_{k,\ell}) &\leq D(A_{k-1,\ell'}) + 1, \end{aligned}$$

where $D(\cdot)$ counts, for a given subsegment, the number of distinct pairs (a,b) over all quadratics ax^2+bx+c in the boundary cost function for that subsegment.

We prove Lemma 14 in [15]. The lemma obtains a polynomial bound on the growth of the number of quadratic pieces by showing, along the way, a polynomial bound on the growth of the number of distinct (a,b) pairs over the quadratics ax^2+bx+c.

As we consider this lemma to be one of the main technical contributions of the paper, we will briefly outline its intuition. It is helpful for us to revisit the doubling behaviour of type (B) paths. Recall that in our example in Figure 5, we may have $|A_{k,\ell}| = 2|A_{k-1,\ell'}|$. This doubling behaviour does not contradict Lemma 14, so long as all quadratic functions along $A_{k-1,\ell'}$ have distinct (a,b) pairs. In fact, for $|A_{k,\ell}| = 2|A_{k-1,\ell'}|$ to occur, each quadratic function in $|A_{k-1,\ell'}|$ must create a new horizontal piece in the cumulative minimum step. But for any two quadratic functions with the same (a,b) pair, only one of them can to create a new horizontal piece, since the horizontal piece starts at the x-coordinate $-\frac{b}{2a}$. Therefore, we must have had that all quadratic functions along $A_{k-1,\ell'}$ have distinct (a,b) pairs. In [15], we generalise this argument and prove $|A_{k,\ell}| \leq |A_{k-1,\ell'}| + D(A_{k-1,\ell'})$.

We perform a similar analysis in the special case of type (B) paths to give the intuition behind $D(A_{k,\ell}) \leq D(A_{k-1,\ell'}) + 1$. For type (B) paths, the number of distinct (a,b) pairs changes only in the cumulative minimum step. All pieces along $A_{k,\ell}$ can either be mapped to a piece along $A_{k-1,\ell'}$, or it is a new horizontal piece. However, all new horizontal pieces have an (a,b) pair of $(0,0)$, so the number of distinct (a,b) pairs increases by only one. For the full proof of Lemma 14 for all three path types, refer to the full version [15].

With Lemma 14 in mind, we can now prove a bound on $|A_{k,\ell}|$ by induction.

▶ **Lemma 15.** *For any $k \in [n+m]$ and $A_{k,\ell} \in A_k$ we have $|A_{k,\ell}| \leq k^2$.*

Proof. Note that in the base case $D(A_{2,\ell}) \leq 2$ and $|A_{2,\ell}| \leq 4$ for any $A_{2,\ell} \in A_2$. By Lemma 14, we get $D(A_{k,\ell}) \leq D(A_{k-1,\ell'}) + 1$, for some subsegment $A_{k-1,\ell'}$ on A_{k-1}. By a simple induction, we get $D(A_{k,\ell}) \leq k$ for any $k \in [n+m]$. Similarly, assuming $|A_{k-1,\ell}| \leq (k-1)^2$ for any $A_{k-1,\ell} \in A_{k-1}$, we use Lemma 14 to inductively obtain $|A_{k,\ell}| \leq |A_{k-1,\ell'}| + D(A_{k-1,\ell'}) + 1 \leq |A_{k-1,\ell'}| + k \leq (k-1)^2 + k - 1 + 1 \leq k^2$ for any $A_{k,\ell} \in A_k$. ◀

Using our lemmas, we can finally bound N, and thereby the overall running time.

▶ **Theorem 16.** *The Continuous Dynamic Time Warping distance between two 1-dimensional polygonal curves of length n and m, respectively, can be computed in time $\mathcal{O}((n+m)^5)$.*

Proof. Using Lemmas 13 and 15, we have

$$N = \sum_{k=2}^{n+m-1} \sum_{\ell=1}^{|A_k|} |A_{k,\ell}| \leq \sum_{k=2}^{n+m} \sum_{\ell=1}^{|A_k|} k^2 \leq \sum_{k=2}^{n+m} 2k^4 \leq 2(n+m)^5.$$

Thus, the overall running time of our algorithm is $\mathcal{O}((n+m)^5)$, by Lemma 12. ◀

4 Conclusion

We presented the first exact algorithm for computing CDTW of one-dimensional curves, which runs in polynomial time. Our main technical contribution is bounding the total complexity of the functions which the algorithm propagates, to bound the total running time of the algorithm. One direction for future work is to improve the upper bound on the total complexity of the propagated functions. Our $O(n^5)$ upper bound is pessimistic, for example, we do not know of a worst case instance. Another direction is to compute CDTW in higher dimensions. In two dimensions, the Euclidean $\mathcal{L}_2$ norm is the most commonly used norm, however, this is likely to result in algebraic issues similar to that for the weighted region problem [18]. One way to avoid these algebraic issues is to use a polyhedral norm, such as the $\mathcal{L}_1$, $\mathcal{L}_\infty$, or an approximation of the $\mathcal{L}_2$ norm [21, 25]. This would result in an approximation algorithm similar to [27], but without a dependency on the spread.

References

1 Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Tight hardness results for LCS and other sequence similarity measures. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015*, pages 59–78. IEEE Computer Society, 2015.

2 Pankaj K. Agarwal, Kyle Fox, Jiangwei Pan, and Rex Ying. Approximating dynamic time warping and edit distance for a pair of point sequences. In Sándor P. Fekete and Anna Lubiw, editors, *32nd International Symposium on Computational Geometry, SoCG 2016*, volume 51 of *LIPIcs*, pages 6:1–6:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

3 Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *Int. J. Comput. Geom. Appl.*, 5:75–91, 1995.

4 Gowtham Atluri, Anuj Karpatne, and Vipin Kumar. Spatio-temporal data mining: A survey of problems and methods. *ACM Comput. Surv.*, 51(4):83:1–83:41, 2018.

5 Selcan Kaplan Berkaya, Alper Kursat Uysal, Efnan Sora Gunal, Semih Ergin, Serkan Gunal, and M Bilginer Gulmezoglu. A survey on ECG analysis. *Biomedical Signal Processing and Control*, 43:216–235, 2018.

6 Donald J. Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In Usama M. Fayyad and Ramasamy Uthurusamy, editors, *Knowledge Discovery in Databases: Papers from the 1994 AAAI Workshop, Seattle, Washington, USA, July 1994. Technical Report WS-94-03*, pages 359–370. AAAI Press, 1994.

7 Krishnan Bhaskaran, Antonio Gasparrini, Shakoor Hajat, Liam Smeeth, and Ben Armstrong. Time series regression studies in environmental epidemiology. *International Journal of Epidemiology*, 42(4):1187–1195, 2013.

8 Sotiris Brakatsoulas, Dieter Pfoser, Randall Salas, and Carola Wenk. On map-matching vehicle tracking data. In *Proceedings of the 31st International Conference on Very Large Data Bases, VLDB 2005*, pages 853–864. ACM, 2005.

9 Milutin Brankovic, Kevin Buchin, Koen Klaren, André Nusser, Aleksandr Popov, and Sampson Wong. (k, ℓ)-medians clustering of trajectories using continuous dynamic time warping. In *SIGSPATIAL '20: 28th International Conference on Advances in Geographic Information Systems*, pages 99–110. ACM, 2020.

10 Karl Bringmann. Why walking the dog takes time: Fréchet distance has no strongly subquadratic algorithms unless SETH fails. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014*, pages 661–670. IEEE Computer Society, 2014.

11 Karl Bringmann and Marvin Künnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015*, pages 79–97. IEEE Computer Society, 2015.

12 Karl Bringmann and Wolfgang Mulzer. Approximability of the discrete Fréchet distance. *J. Comput. Geom.*, 7(2):46–76, 2016.

13 Kevin Buchin, Maike Buchin, Wouter Meulemans, and Wolfgang Mulzer. Four soviets walk the dog: Improved bounds for computing the Fréchet distance. *Discret. Comput. Geom.*, 58(1):180–216, 2017.

14 Kevin Buchin, Maike Buchin, and Yusu Wang. Exact algorithms for partial curve matching via the fréchet distance. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009*, pages 645–654. SIAM, 2009.

15 Kevin Buchin, André Nusser, and Sampson Wong. Computing continuous dynamic time warping of time series in polynomial time. *CoRR*, abs/2203.04531, 2022.

16 Kevin Buchin, Tim Ophelders, and Bettina Speckmann. SETH says: Weak Fréchet distance is faster, but only if it is continuous and in one dimension. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019*, pages 2887–2901. SIAM, 2019.

17 Maike Buchin. *On the computability of the Fréchet distance between triangulated surfaces*. PhD thesis, Freie Universität Berlin, 2007.

18 Jean-Lou De Carufel, Carsten Grimm, Anil Maheshwari, Megan Owen, and Michiel H. M. Smid. A note on the unsolvability of the weighted region shortest path problem. *Comput. Geom.*, 47(7):724–727, 2014.

19 Ian R Cleasby, Ewan D Wakefield, Barbara J Morrissey, Thomas W Bodey, Steven C Votier, Stuart Bearhop, and Keith C Hamer. Using time-series similarity measures to compare animal movement trajectories in ecology. *Behavioral Ecology and Sociobiology*, 73(11):1–19, 2019.

20 Anne Driemel, Amer Krivosija, and Christian Sohler. Clustering time series under the Fréchet distance. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016*, pages 766–785. SIAM, 2016.

21 Richard M Dudley. Metric entropy of some classes of sets with differentiable boundaries. *Journal of Approximation Theory*, 10(3):227–236, 1974.

22 Alon Efrat, Quanfu Fan, and Suresh Venkatasubramanian. Curve matching, time warping, and light fields: New algorithms for computing similarity between curves. *J. Math. Imaging Vis.*, 27(3):203–216, 2007.

23 Philippe Esling and Carlos Agón. Time-series data mining. *ACM Comput. Surv.*, 45(1):12:1–12:34, 2012.

24 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.

25 Sariel Har-Peled and Mitchell Jones. Proof of Dudley's convex approximation. *arXiv preprint*, 2019. `arXiv:1912.01977`.

26 Koen Klaren. Continuous dynamic time warping for clustering curves. Master's thesis, Eindhoven University of Technology, 2020.

27 Anil Maheshwari, Jörg-Rüdiger Sack, and Christian Scheffer. Approximating the integral Fréchet distance. *Comput. Geom.*, 70-71:13–30, 2018.

28 Jessica Meade, Dora Biro, and Tim Guilford. Homing pigeons develop local route stereotypy. *Proceedings of the Royal Society B: Biological Sciences*, 272(1558):17–23, 2005.

29 Joseph S. B. Mitchell and Christos H. Papadimitriou. The weighted region problem: Finding shortest paths through a weighted planar subdivision. *J. ACM*, 38(1):18–73, 1991.

30 Meinard Müller. Dynamic time warping. In *Information Retrieval for Music and Motion*, pages 69–84. Springer, 2007.

31 Mario E. Munich and Pietro Perona. Continuous dynamic time warping for translation-invariant curve alignment with applications to signature verification. In *Proceedings of the International Conference on Computer Vision, 1999*, pages 108–115. IEEE Computer Society, 1999.

32 Cory Myers, Lawrence Rabiner, and Aaron Rosenberg. Performance tradeoffs in dynamic time warping algorithms for isolated word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(6):623–635, 1980.

33 Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978.

34 Pavel Senin. Dynamic time warping algorithm review. *Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA*, 855(1-23):40, 2008.

35 Bruno Serra and Marc Berthod. Subpixel contour matching using continuous dynamic programming. In *Conference on Computer Vision and Pattern Recognition, CVPR 1994*, pages 202–207. IEEE, 1994.

36 E. Sriraghavendra, K. Karthik, and Chiranjib Bhattacharyya. Fréchet distance based approach for searching online handwritten documents. In *9th International Conference on Document Analysis and Recognition, ICDAR 2007*, pages 461–465. IEEE Computer Society, 2007.

37 Yaguang Tao, Alan Both, Rodrigo I Silveira, Kevin Buchin, Stef Sijben, Ross S Purves, Patrick Laube, Dongliang Peng, Kevin Toohey, and Matt Duckham. A comparative analysis of trajectory similarity measures. *GIScience & Remote Sensing*, pages 1–27, 2021.

38 Charles C. Tappert, Ching Y. Suen, and Toru Wakahara. The state of the art in online handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(8):787–808, 1990.

39 Stephen J Taylor. *Modelling financial time series.* World Scientific, 2008.

40 Kevin Toohey and Matt Duckham. Trajectory similarity measures. *ACM SIGSPATIAL Special*, 7(1):43–50, 2015.

41 Taras K Vintsyuk. Speech discrimination by dynamic programming. *Cybernetics*, 4(1):52–57, 1968.

42 Xiaoyue Wang, Abdullah Mueen, Hui Ding, Goce Trajcevski, Peter Scheuermann, and Eamonn J. Keogh. Experimental comparison of representation methods and distance measures for time series data. *Data Min. Knowl. Discov.*, 26(2):275–309, 2013.

43 Öz Yilmaz. *Seismic data analysis: Processing, inversion, and interpretation of seismic data.* Society of exploration geophysicists, 2001.