Automatic Detection of Warped Patterns in Time Series: The Caterpillar Algorithm

Conference Paper · November 2018

DOI: 10.1109/ICBK.2018.00063

3 authors, including:

Maximilian Leodolter
AIT Austrian Institute of Technology
21 PUBLICATIONS **40** CITATIONS

SEE PROFILE

Automatic Detection of Warped Patterns in Time Series: The Caterpillar Algorithm

Maximilian Leodolter
Center for Mobility Systems
Austrian Institute of Technology
Vienna, Austria
maximilian.leodolter@ait.ac.at

Norbert Brändle
Center for Mobility Systems
Austrian Institute of Technology
Vienna, Austria
norbert.braendle@ait.ac.at

Claudia Plant
Faculty of Computer Science
ds:UniVie University of Vienna
Vienna, Austria
claudia.plant@univie.ac.at

***Abstract*—Detection of similar representations of a given query time series within longer time series is an important task in many applications such as finance, activity research, text mining and many more. Identifying time warped instances of different lengths but similar shape within longer time series is still a difficult problem. We propose the novel Caterpillar algorithm which fuses the advantages of Dynamic Time Warping (DTW) and the Minimum Description Length (MDL) principle to move a sliding window in a crawling-like way into the future and past of a time series. To demonstrate the wide field of application and validity, we compare our method against state-of-the-art methods on accelerometer time series and synthetic random walks. Our experiments demonstrate that Caterpillar outperforms the comparison methods in detecting accelerometer signals of metro stops.**

***Index Terms*—Minimum Description Length, Dynamic Time Warping, Accelerometer Data**

I. Introduction

Time series data are ubiquitous across many fields such as finance, medicine and activity research. Figure 1a shows an example of a time series, illustrating the accelerometer signal recorded by a person traveling on a metro while carrying a smartphone. Intuitively, the red colored segments resemble the given query pattern in Figure 1b, which represents a typical braking-stopping-accelerating sequence at a metro stop. Further it is obvious that in between the stations the signal has a high variation and follows no distinct pattern. However, it is difficult to detect these vertically shifted and time warped instances of the query pattern.

We propose a novel technique solving the above problem and with (1) a novel modeling approach combining the Dynamic Time Warping (DTW) and Minimum Description Length (MDL) principle and (2) a novel algorithm – the Caterpillar – efficiently detecting time warped fits of model time series within longer time series. The proposed model encodes a time series conditionally on a time warped representation. Using simple matrix transformations the algorithm moves a sliding window, similar to a caterpillar extending and contracting to crawl. Figure 1a illustrates the detected metro stops of different time extension (red numbers) and vertical scale.

When comparing two time series, the length of the query pattern and the time extension of typical shapes often vary,

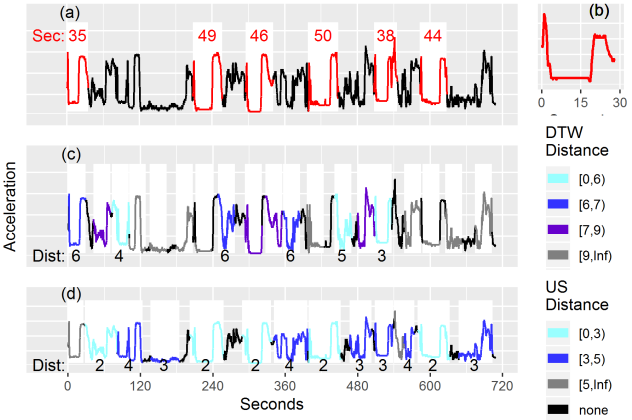


Figure 1: Horizontal accelerometer signal component of a person traveling on a metro while carrying a smartphone. In (a) the Caterpillar algorithm detected the red colored time warped (duration in seconds is plotted) and vertically shifted instances of the query pattern in (b). In (c) the fits found by the baseline method UCR Suite DTW and the respective distances, and the equivalent for UCR Suite US in (d).

such as the duration of the stop or braking in the metro smartphone example above. In many cases it is therefore crucial to consider varying length and time extension when selecting an appropriate distance measure for time series analysis tasks such as pattern detection, classification and clustering. Common distance measures between time series include Euclidean distance, Fréchet distance, SAX [13], Uniform Scaling (US) [15], edit distances, Dynamic Time Warping (DTW) [17], and many more. DTW allows a non-linear alignment of two time series and can thus match them in case of similar shape, yet different time extensions.

Time series data analysis often boils down to the binary question of whether a given time series sample fits to a query time series or not. Traditionally, any distance measure requires setting a threshold parameter. The threshold parameter value is crucial, yet difficult to identify, and in general heavily affects the analysis results. UCR Suite [15] (applying DTW or US) is the state-of-the-art method for scanning longer a time series to detect a query pattern. Given the set of distances to the query pattern of the fits detected by UCR Suite, it is hard to identify

a suitable threshold. Figure 1c and Fig. 1d show the result of matching the query pattern of Figure 1b using UCR Suite with DTW and US distances, respectively. The first segment starting at second 0 with a DTW distance of 6 obviously resembles the query pattern, and the one starting at second 430 with a lower DTW distance clearly not. The same applies to the detected segments and their US distances in Fig 1d starting at second 0 and 130 (more details in Sec. V-C).

The MDL principle is a promising tool which does not require identifying such sensitive distance parameter values, and can be considered as a formalization of Occam's Razor. The main idea is that the best model for a given data set will be the model achieving the highest data compression rate. MDL has been applied in many fields so far (e.g. bioinformatics [20], text mining [5], etc.) for model selection and also for time series analysis [2], [21], [7], [16], [22]. Yet it is still an ongoing process to investigate the potential of MDL for time series analysis. To the best of our knowledge there exists no similar approach combining entropy based MDL and DTW, thus avoiding parameter value setting for distances.

Our main contributions are:

- A novel encoding scheme fusing the MDL principle and DTW to match time warped time series.
- The novel Caterpillar algorithm matching time series in the future and the past to find the best alignment.
- Extensive experiments demonstrate broad applicability on accelerometer time series and synthetic data.

We denote a univariate time series as $\mathbf{x} := \{x_i\}_{i=1\ldots m}$ and repeat the definition of the well known Euclidean distance for two time series $\mathbf{x}$ and $\mathbf{h}$ of same length as: $\sqrt{\sum(x_i - h_i)^2}$.

This paper is organized as follows: The next section discusses different stepwise calculations of DTW. Section III proposes the MDL encoding scheme for time series of different lengths that is applied in the Caterpillar algorithm presented in Sec. IV. Section V demonstrates the broad applicability of our method by extensive experiments before we discuss our method in the context of related work in Sec. VI.

II. Fast Stepwise DTW Calculation

The Euclidean distance is not well suited to recognize distorted patterns along the time axis in time series, since it requires a 1-1 alignment between an entry of $\mathbf{x}$ and a single entry of $\mathbf{h}$. The work in [17] introduced the Dynamic Time Warping (DTW) distance to find non-linear alignments between time series of possibly different lengths. To calculate the DTW we define the matrix of differences Δ, ($\Delta_{ij} = x_i - h_j$) and the cost matrix $\mathbf{C}$ ($\mathbf{C}_{ij} = |\Delta_{ij}|$). The cumulative global cost matrix $\mathbf{G}$ is calculated as follows:

$$\mathbf{G}_{i,j} = \begin{cases} \sum_{k \le i} \mathbf{C}_{k,1} & j = 1 \\ \sum_{l \le j} \mathbf{C}_{1,l} & i = 1 \\ \mathbf{C}_{i,j} + \min(\mathbf{G}_{i-1,j}, \ \mathbf{G}_{i,j-1}, \ \mathbf{G}_{i-1,j-1}) & i,j > 1 \end{cases} \quad (1)$$

with $\mathbf{G}_{nm}$ being the DTW distance measure. The matrix $\mathbf{G}$ is calculated simultaneously with the direction matrix $\mathbf{D}$, $\mathbf{D}_{ij} \in \{1,2,3\}$. Each element of $\mathbf{D}$ represents which step is cheapest to take next, either diagonal (1), horizontal (2) or vertical (3). The warping path ω is an excerpt of $\mathbf{D}$ and is the vector of steps (diagonal, vertical, horizontal) taken to find the cheapest path from $\mathbf{G}_{nm}$ back to $\mathbf{G}_{11}$ to align the two time series $\mathbf{x}$ and $\mathbf{h}$ with the lowest cumulative costs. The difference path δ is an excerpt of Δ, and constitutes the vector of differences of the elements of $\mathbf{x}$ and $\mathbf{h}$ that are aligned to each other as described by ω. Finally, to represent $\mathbf{x}$ by a given $\mathbf{h}$, we require information about the deviations (δ) and the alignment (ω) of the time series. Figures 2a-c illustrate the defined matrices and vectors for a simple example.

A. In-/Decremental DTW

Given the result of $DTW(\mathbf{x}, \mathbf{h})$ for $\mathbf{h}$ and $\mathbf{x} = \{x_i\}_{i=1\ldots m}$, suppose new observations are appended at the end of $\mathbf{x}$ denoted as $\mathbf{x}^+ = \{x_i\}_{i=1\ldots m+k}$. To calculate $DTW(\mathbf{x}^+, \mathbf{h})$ we can append the cumulative costs of new observations according to (1) to the already computed $\mathbf{G}$ and receive $\mathbf{G}^+$. Figure 2d shows the incremental step to include a further observation of $\mathbf{x}$, 7 to the alignment to $\mathbf{h}$.

The opposite of the incremental step is to omit potentially irrelevant observations at the end of $\mathbf{x}$ and recalculate the DTW distance given the results of the complete case. The matrices Δ, $\mathbf{C}$ and $\mathbf{G}$ can be updated directly by omitting the respective columns at the end. Figure 2e shows the decremental step to exclude the final observation of $\mathbf{x}$, 7 from the alignment to $\mathbf{h}$.

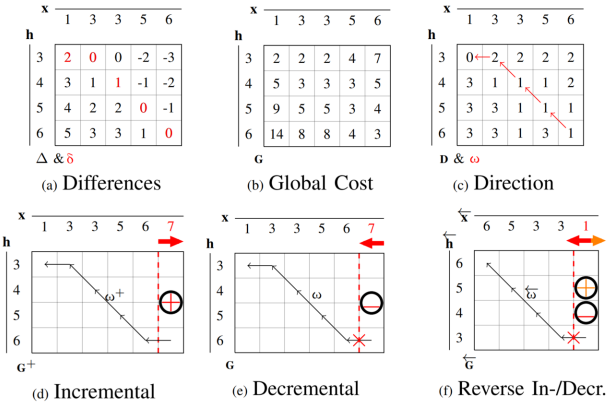


Figure 2: DTW computation: Based on Δ (a) the matrix $\mathbf{G}$ (b) and $\mathbf{D}$ (c) are computed in parallel. Next the paths ω (c) and δ (a) are extracted from the respective matrices by tracking back the given steps in $\mathbf{D}$. Also the incremental inclusion (d) of a new observation of $\mathbf{x}$, 7 (e) the decremental exclusion of the last element, 7 and (f) the reverse-decremental exclusion of the first element, 1.

B. Reverse In-/Decremental DTW

In-/decremental steps allow to react to alterations of the end of $\mathbf{x}$. Can we do the same for alterations at the initial observations of $\mathbf{x}$? The answer is no, because the basic structure of the DTW algorithm and the dynamic calculation

of $\mathbf{G}$ does not allow to alter, add or omit observations at the beginning (initial/past values of $\mathbf{x}$) without changing the entire global cost matrix. However [1] proved the DTW distance calculation to be reversible: $DTW(\mathbf{x}, \mathbf{h}) = DTW(\overleftarrow{\mathbf{x}}, \overleftarrow{\mathbf{h}})$, where $\overleftarrow{\mathbf{x}}$ is $\mathbf{x}$ in reverse order: $\overleftarrow{\mathbf{x}} \coloneqq \mathbf{x}(m : 1)$. We apply this principle and proceed analogously to the incremental (decremental) step to finally add (omit) initial values of $\mathbf{x}$. Figure 2f depicts the reverse time series after omitting 7. To exclude the first element of $\mathbf{x}$, 1 we calculate $DTW(\overleftarrow{\mathbf{x}}, \overleftarrow{\mathbf{h}})$. The main advantage of the reverse-in-/decremental calculation becomes evident when comparing results of adding or omitting more than one initial element of $\mathbf{x}$. Since for each step after the initial step, the results from the previous calculation can be recycled analogously to the in-/decremental DTW calculation, again computation time can be saved (details in section IV-E).

C. Vector based Dynamic Time Warping Implementation

If one is interested in only the DTW distance measure and not in the alignment, one can save computation time by not allocating the matrices $\mathbf{D}$, $\mathbf{C}$ and $\mathbf{G}$ for the DTW calculation. Since the calculation of the j-th column of $\mathbf{G}$, $\mathbf{G}_{.,j}$ only depends on $\mathbf{G}_{.,j-1}$, instead of allocating a new vector, the vector storing information of $\mathbf{G}_{.,j-2}$ is overwritten with $\mathbf{G}_{.,j}$. Incremental calculations for new observations are possible as well, as long as the previously calculated vector $\mathbf{G}_{.,m}$ is saved. For decremental calculations we allocate an additional vector that stores the last entries of each column, $\mathbf{G}_{n,.}$.

We stress this calculation principle (also implemented in the R packages IncDTW [11] and rucrdtw [15] that we also apply) since we apply it for the fusion of DTW and MDL in Sec. IV-D.

III. DTW-based description length

To apply the MDL principle we first normalize and discretize a time series $\mathbf{x}^0$ and define the discrete time series $\mathbf{x}$, $\forall j \in 1 \dots m$ (also done in e.g. [16], [2], [7], [21], [22]) as:

$$x_j \coloneqq round\Big(\frac{x_j^0 - \min}{\max - \min} \cdot 2^b - 1\Big) + 1. \tag{2}$$

min and max are the minimum and maximum values of $\mathbf{x}^0$. For discretization of a set of time series, min and max are replaced by global equivalents. The value of b is set to 6 which hardly affects the accuracy of common time series analysis tasks (such as clustering or classification) as discussed in [16].

A. Conditional Description Length

We define the number of required bits to encode a time series of length m as the description length:

$$DL(\mathbf{x}) = -\sum_{i=1}^{m} \log_2\big(P(x_i)\big), \tag{3}$$

where P is the probability function, discussed in detail in Sec. III-B. We apply the two part MDL principle to describe the conditional description length and estimate the description length by the entropy of the time series. According to Krafts inequality, the length of any lossless code is lower bounded by the entropy. In case the hypothesis $\mathbf{h}$ and $\mathbf{x}$ are of the same length, the code length of $\mathbf{x}$ conditional on the hypothesis $\mathbf{h}$ is defined similar to their Euclidean distance (in the following called **E-model**):

$$DL(\mathbf{x}, \mathbf{h}) \coloneqq DL(\mathbf{h}) + DL(\mathbf{x} \mid \mathbf{h}) = DL(\mathbf{h}) + DL(\mathbf{x} - \mathbf{h}). \tag{4}$$

We call $DL(\mathbf{x}, \mathbf{h})$ from (4) the conditional description length (CDL) of $\mathbf{x}$ given $\mathbf{h}$ according to the **E-model**. For time series of different lengths and/or different extensions in time of typical patterns, we propose to incorporate the DTW algorithm into the MDL principle. As discussed in Sec. II, the DTW algorithm not only calculates the DTW distance measure, but also the warping path ω and the path of differences δ. We use these components to incorporate the main advantage of DTW allowing non-linear alignments of time series of different lengths and define the DL for $\mathbf{x}$ conditional on $\mathbf{h}$ (and call it **DTW-model**):

$$\begin{aligned} DL(\mathbf{x}, \mathbf{h}) &\coloneqq DL(\mathbf{h}) + DL(\mathbf{x} \mid \mathbf{h}) \\ DL(\mathbf{x} \mid \mathbf{h}) &\coloneqq DL(\delta) + DL(\omega) = DL(\delta) + |\omega| \lceil \log_2 3 \rceil. \end{aligned} \tag{5}$$

Comparing (4) and (5), $DL(\mathbf{x} - \mathbf{h})$ is the equivalent of $DL(\delta)$. The additional warping costs $DL(\omega)$ trade off the possibly lower DL of differences due to non-linear alignments. As every direction in the path is about equally frequent, we use 2 bits to encode each element.

Given a set of time series $\mathbf{X}$ and a hypothesis $\mathbf{h}$, a time series $\mathbf{x}$ is a candidate-match if $DL(\mathbf{x}) > DL(\mathbf{x}|\mathbf{h})$. If the sum of the saved costs for all candidate-matches is larger than $DL(\mathbf{h})$, then the hypothesis $\mathbf{h}$ is accepted and the candidate-matches are considered as true matches. Otherwise it is cheaper to describe candidate-matches unconditionally.

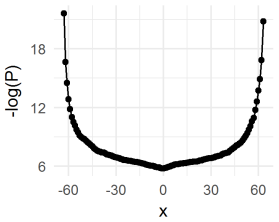
In general, two time series to be compared are not of the same length. Furthermore, depending on the application, it might be preferable not to match the entire time series $\mathbf{x}$ to a given hypothesis $\mathbf{h}$, but to consider only a segment of $\mathbf{x}$ as a valid match to $\mathbf{h}$. Since the logarithm is additive we can easily split a time series $\mathbf{x}(1 : m)$ into disjunct segments and describe different segments independently from each other, either unconditionally or conditionally to $\mathbf{h}$. If multiple fits are found, the saved DL is the difference between DL(x) and the sum of encoding costs for the conditional and unconditional segments. For lossless compression, we also need to encode the information of the starting indices, ι of the detected candidate fits. The index costs are defined as:

$$DL(\text{indices}) \coloneqq \sum_{i \geq 1} \lceil \log_2(m - \xi_{i-1}) \rceil,$$

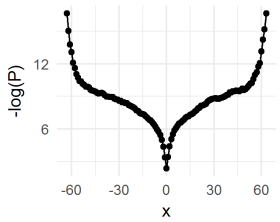
where $\xi_0 = 0$. The index costs for the second fit are smaller than those of the previous, since we make use of $\iota_2 > \xi_1$. If $DL(\text{indices})$ is bigger than the saved DL, the candidate fits are discarded.

B. Probability Density Function

The applied probability density function (PDF) in (3) is supposed to be as general as possible and applicable in many



(a) DL for 1-1 alignment



(b) DL for non-linear alignment

Figure 3: DL of differences: Logarithms of the estimated Probability functions.

different domains and at the same time provide valid approximations of the intrinsic PDFs. In contrast to the approaches of [16], [22], we apply global PDFs for our models instead of estimating P directly from each time series segment, since we intensively stress the additivity of the CDL. In the following we describe how we selected the PDF and Sec. V will demonstrate the broad field of application for our choice. We choose to apply a uniform PDF for the unconditional DL, so that $DL(\mathbf{x}) = |\mathbf{x}| \cdot b = |\mathbf{x}| \cdot 6$. This way the DL depends only on the length of a time series and not on its values, which implicates that the decision if **x** is described best by **h** depends only on the differences and not on the values of **x** itself.

Further we apply different PDFs for linear 1-1 alignments of time series of same lengths and non-linear alignments of time series of possibly unequal lengths. For the E-model (4) we simulate 1000 couples of random walks (**x**, **h**, each of length $= 10^6$) independent from each other, discretize them and count the occurrences of the difference vectors (**x**−**h**). For the DTW-model (5) we generate 1000 couples of random walks (**x**, **h**) independent from each other. The length m of **x** is chosen randomly between 100 and 1000, and the length n of **h** is drawn randomly out of the interval $[0.5{\cdot}m, \dots 2{\cdot}m]$. This dependency between the lengths avoids unreasonable combinations. We then calculate DTW and the path of differences δ and count the frequencies per possible value. Figure 3 shows the negative logarithms (DL for a single element of difference) of the two populations. The non-linear alignment clearly causes a higher density – equivalent to lower encoding costs – around 0 than the linear 1-1 alignment.

IV. CATERPILLAR

Given two time series **x** and **h** of length m and n, with $m > n$, we want to scan the longer time series **x** to recognize possibly warped instances of the hypothesis-pattern **h** as part of it. Before proposing our method – the Caterpillar algorithm – we mention the baseline approach (inspired by [16] and adapted here by applying a global P instead of segmentwise since we make use of the additivity of the DL) applying the E-model from (4) and denote it as PREMDL (Pattern recognition by Euclidean distance and the MDL). For each index $i \in 1 \dots m-n$, we calculate $DL(\mathbf{x}(i:i+n), \mathbf{h})$ and evaluate if the following two conditions $\forall j \in \{i-n, \dots, i+n\}$ hold to find a match.

$$DL(\mathbf{x}(i:i+n),\ \mathbf{h}) < DL(\mathbf{x}(i:i+n))$$
$$DL(\mathbf{x}(i:i+n),\ \mathbf{h}) < DL(\mathbf{x}(j:j+n), \mathbf{h}).$$

A. The Caterpillar Algorithm

We propose the following Caterpillar algorithm to scan a longer time series **x** to detect a warped instance of **h**. Given a couple of starting indices ι (lower) and ξ (upper) the Caterpillar algorithm performs one of the following movement steps:

- Forward: Increase ξ: $\xi^+ > \xi$; evaluate $DL(\mathbf{x}(\iota : \xi^+), \mathbf{h})$
- Back up: Decrease ξ: $\xi^- \leq \xi$; evaluate $DL(\mathbf{x}(\iota : \xi^-), \mathbf{h})$
- Catch up: Increase ι: $\iota^+ \geq \iota$; evaluate $DL(\mathbf{x}(\iota^+ : \xi), \mathbf{h})$
- Backward: Decr. ι: $\iota^- < \iota$; evaluate $DL(\mathbf{x}(\iota^- : \xi), \mathbf{h})$.

The CDL is calculated according to the DTW-model (5). These 4 steps enable the algorithm to move the scanning window $(\iota : \xi)$ along **h** similarly to a caterpillar that is extending to move forward or backward, and can also contract its front to back up, or its back to catch up.

Each of the movement steps requires to compute the DTW. To save computation time we calculate the DTW according to the computation methods proposed in Sec. II: incremental for forward, decremental for back up, reverse-decremental for catch up, and reverse-incremental for backward. Figures 2d-f give examples for the respective calculations. Finally a match is found if:

$$DL(\mathbf{x}(\iota : \xi) \mid \mathbf{h}) < DL(\mathbf{x}(\iota : \xi)). \tag{6}$$

Algorithm 1 gives details on how these movements help the Caterpillar algorithm to detect warped patterns similar to **h** in **x**. Figures 4a-d depict the iterative movements of the Caterpillar for a simple example. The Caterpillar also helps to decide whether two time series should be matched completely (Fig. 4b), or rather partially (Fig. 4d).

B. Find initial lower index ι

Because of the additivity of the DL definition we can initiate the Caterpillar algorithm in parallel on different initial indices ι. To find initials for the Caterpillar algorithm we apply the PREMDL algorithm. The first index of a found match is selected to be an initial lower index ι. Depending on the application, the proposed algorithms can also detect matches in **x** of the same shape as **h**, however vertically shifted for a fixed shift α (compare 6):

$$DL(\mathbf{x}(\iota : \xi), \mathbf{h} + \alpha) + DL(\alpha) < DL(\mathbf{x}(\iota : \xi)). \tag{7}$$

For a given index ι we define α implicitly when the PREMDL algorithm calculates $DL(d)$ where $d = \mathbf{x}(\iota : \iota + n) - \mathbf{h}$. We define $\alpha \coloneqq mean(d)$.

C. Lower Bound for CDL: Find initial upper index ξ

In case $|\mathbf{h}| = n > m = |\mathbf{x}|$ we estimate a lower bound for the CDL for a complete match (**x** is matched from begin to end to **h**):

$$DL(\mathbf{x}, \mathbf{h}) = DL(\omega) + DL(\delta) + DL(\mathbf{h}) \geq DL(\omega) + DL(\delta)$$
$$\geq |\omega| \cdot \lceil \log_2 3 \rceil - |\delta| \cdot \log_2 p(0) \geq n(2 - \log_2 p(0))$$

The first inequality follows from omitting the costs for the hypothesis, which is certainly positive. Also, in case of many matches, the costs are not considered for the calculation of a single match. The second inequality holds, since we suppose the PDF for the warping path to be uniform and the cheapest path for the differences is the one of zeros. The third inequality holds, since the shortest warping path is the one with equal length as the longer time series, **h**. So we can combine:

$$DL(\mathbf{x}) = 6m \overset{?}{\leq} n(2 - \log_2 p(0)) \leq DL(\mathbf{x} \mid \mathbf{h}) \tag{8}$$

As long as inequality (8) holds no CDL for the combination of **x** and **h** needs to be calculated. In case of the proposed distributions of Fig. 3b we abandon calculations as long as $m \leq n\frac{1}{6}(2 - \log_2 p(0)) \approx n \cdot 0.73$. Finally we use this lower bound to determine the initial extension of the Caterpillar, the initial index ξ (which is also the minimum extension of the Caterpillar) for a given ι, respectively:

$$\mu := 0.73 \cdot n + \iota. \tag{9}$$

If **x** is much longer than **h**, nly a segment of the time series (**y**) is matched to the hypothesis and the remaining segment (**z**) that is described unconditionally. The just derived estimation of the lower bound holds also for this case, since:

$$\begin{aligned} DL(\mathbf{x}) = DL(\mathbf{y}_i) + DL(\mathbf{z}_i) &\overset{?}{\leq} DL(\mathbf{y}_i \mid \mathbf{h}) + DL(\mathbf{z}_i) \\ \iff DL(\mathbf{y}_i) &\overset{?}{\leq} DL(\mathbf{y}_i \mid \mathbf{h}) \end{aligned} \tag{10}$$

Due to the additivity of the DL , it does not matter how long the time series **x** is. The lower bound estimation only depends on the segment to be matched , **y**.

D. Accelerate The Caterpillar

According to the DTW-model (5), the DL is determined by these two steps:

1 calculate the DTW, and
2 determine the DL on top of the results of step 1.

We fuse these two components to enhance the DTW-model and the Caterpillar algorithm. That is, the DL is used to define the cost matrix **C** for the DTW algorithm. In detail:

$$\mathbf{C}_{i,j} := DL(x_i - h_j) = -\log_2 P(x_i - h_j) \tag{11}$$

Then **C** is used as input for the DTW algorithm to calculate **G** as usual and the calculated DTW distance equals DL(δ). To take care of the additional costs for the warping path, we could trace back **G** to get the warping path ω. However, since we are only interested in the costs of warping, but not ω itself, we skip the interim stage of calculating ω and add the warping costs in the calculation of the global cost matrix. This way we save computation time by not backtracking the global cost matrix to find the warping path. We define the global cost matrix as follows (compare 1):

$$\mathbf{G}_{i,j} = \begin{cases} \sum_{k \leq i} \mathbf{C}_{k,1} + i \cdot 2 & j = 1 \\ \sum_{l \leq j} \mathbf{C}_{1,l} + j \cdot 2 & i = 1 \\ \mathbf{C}_{i,j} + 2 + \min(\mathbf{G}_{i-1,j},\ \mathbf{G}_{i,j-1},\ \mathbf{G}_{i-1,j-1}) & i,j > 1 \end{cases} \tag{12}$$

We call the DL(**x**, **h**) calculated by the DTW distance between **x** and **h** based on (11) and (12) the fusion-DTW-model (**fDTW-model**).

Since we are not interested in **G** itself, but only the last entry in the last row, we apply the principle of the vector based DTW implementation II-C and additionally save the last column for incremental steps and the last row for decremental steps.

Algorithm 1 sketches the Caterpillar algorithm. As long as the end (ξ^{max}) is not reached and one of the four movements achieves to decrease the CDL, the Caterpillar moves. Each forward while loop increases ξ until the CDL increases. Then the back up step potentially decreases ξ in case the last forward step was too far and the current CDL value is below the DL(x). Next, if no catch up has been performed yet, the Caterpillar has the option to move backward until the start is reached (ι^{min}) or the CDL increases. Finally, the Caterpillar catches up. Without any prior knowledge, we set the initial indices $\iota^{min} = \iota = 1$, $\xi^{max} = |\mathbf{x}|$ (or restrict them by neighboring initials, respectively) and $\xi = \mu$ (see 9).

Algorithm 1 Caterpillar

```
PREMDL defines the tuples (ι, ξ, ι^min, ξ^max)
for each tuple call Caterpillar:
procedure CATERPILLAR(x, h, ι, ξ, ι^min, ξ^max)
    CatchUpSuccess ← FALSE
    CDL ← DL(x)
    while ξ < ξ^max ∧ CDL ↘ do          ▷ as long as CDL decreases
        repeat
            (CDL, ι, ξ) ← Forward(ι, ξ)  ▷ increase ξ to decrease CDL
        until ξ = ξ^max ∨ CDL ↗
        if CDL < DL(x) then
            (CDL, ι, ξ) ← BackUp (ι, ξ) ▷ decrease ξ to decrease CDL
        end if
        if !CatchUpSuccess then
            repeat
                (CDL, ι, ξ) ← Backward (ι, ξ) ▷ decrease ι to decrease CDL
            until ι = ι^min ∨ CDL ↗
        end if
        (CDL, ι, ξ) ← CatchUp (ι, ξ)     ▷ increase ι to decrease CDL
        if CatchUp successful then
            CatchUpSuccess ← TRUE
        end if
    end while
    return (ι, ξ)                        ▷ return the indices of smallest CDL
end procedure
```

Fusing DTW and MDL (12) saves computation time by omitting the backtracking step to find the warping path, and enables fast versions of the two Caterpillar movements: back up step and catch up step. To find the optimal index, we compare all possible values since these are stored in the already computed vectors storing information of the last row and last column of the global cost matrix. In detail:

a) Back Up: **G** is already computed[1] for $\mathbf{x}(\iota : \xi)$ and $\mathbf{h}(1 : n) = \mathbf{h}$, and w.l.o.g say $\iota = 1$. Suppose the last forward movement of the Caterpillar increases the total costs. To evaluate possible back up step widths $j \in \{0, \ldots \xi - \mu\}$, we compare $\mathbf{G}(n,\ \iota - \xi + 1 - j) + 6j$ = $\mathbf{G}(n,\ \xi - j) + 6j$.

[1]Actually we never compute the whole matrix **G**, but only the last column and row, but for simplicity we stick to this notation.

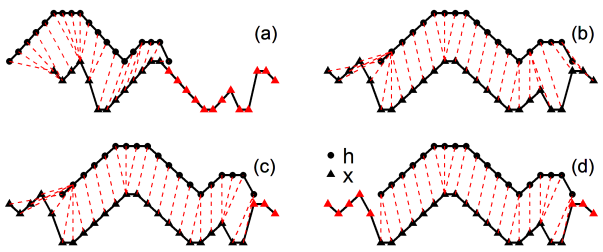


Figure 4: The Caterpillar: After the initial alignment (a) the forward movement concludes in the complete matching (b). Backing up helps to exclude the final 3 observations of **x** (c), and finally the catch up in (d) shows the best alignment.

In Fig. 4a the Caterpillar algorithm first aligns the time series **x** and **h** for the indices $\iota = 1$ and $\xi = \iota + 0.73 \cdot n$. Next the Caterpillar moves forward until $\xi = m$, so the end is reached (Fig. 4b), before it decides to back up three steps ($j = 3$) and represent the last three elements of **x** unconditionally (Fig. 4c).

b) Backward/ Catch Up: To evaluate possible catch up and backward steps, we calculate $DTW(\overleftarrow{\mathbf{x}}, \overleftarrow{\mathbf{h}})$. Then we evaluate possible back up step widths for the reverse time series $\overleftarrow{\mathbf{x}}$ and $\overleftarrow{\mathbf{h}}$ – which is equivalent to the catch up of the original time series **x** and **h** – by comparing the last row of $\overleftarrow{\mathbf{G}}$ analogously to the back up step of **x** and **h**. Further we can evaluate possible backward steps in parallel to the catch up step by calculating an incremental step of $\overleftarrow{\mathbf{G}}$ and comparing the last row. In Fig. 4d the Caterpillar algorithm decides to catch up and excludes the initial 5 observations of **x** from alignment to **h** and finally concludes with the optimal fit.

E. Complexity

The runtime complexity of the E-model and fDTW-model are identical to the calculation of the Euclidean distance $O(m)$, DTW distance $O(n \cdot m)$, respectively. One forward step of the Caterpillar algorithm has $O(n)$ complexity which is much less than computing it again from scratch at the costs of $O(n \cdot (m + 1))$. The decremental movement has $O((1 - 0.73) \cdot n)$ complexity according to (9). The same is valid for the reverse movements (except for the initial reverse movements).

V. Results

To demonstrate that our proposed method detects time warped instances of a hypothesis time series **h**, we compare (1) the proposed encoding scheme isolated from scanning algorithms with traditional distance based methods and similar state of the art encoding schemes for linear and non-linear alignments on synthetic data (Sections V-A and (2) the Caterpillar, PREMDL and UCR Suite [15] on synthetic data and accelerometer data (Sections V-B and V-C).

Our comparative evaluation includes two additional models from the literature which describe a time series' DL and CDL based only on the cardinality c of a time series. The first model introduced by [2] defines $DL(\mathbf{x} \mid \mathbf{h})$ by counting the mismatches of **x** and **h**, and for each mismatch the costs for one observation plus the index costs are saved (we call it **C-Model**). The second additional model proposed by [21] is an extension of the C-Model for time series of possibly different lengths by incorporating DTW and the coding scheme of [2]. They propose to count the mismatches of **x** and **h** along the warping path of these two time series (we call it **cDTW-Model**). Both models depend on the number of mismatches, however are independent of the magnitude of the mismatches.

Since the Caterpillar scans longer time series to detect shorter query patterns, we compare the Caterpillar against this task's state-of-the-art method **UCR Suite** [15] (applying DTW or US). UCR Suite compares a given query pattern with segments of the same length of a longer time series via a sliding window approach. The DTW distance is calculated for each comparison (unless lower bounding avoids it), and early abandoning aborts the calculation. The original algorithm returns the segment with the lowest DTW distance and the distance itself. We apply UCR Suite multiple times to return multiple fits in time series (see Figures 1c and d). UCR Suite US is a combination of pruning and lower bounding techniques to accelerate calculations and a generalization of the Euclidean Distance – Uniform Scaling – such that fits of different lengths (we allowed scaling factors between 0.5 and 2) can be detected.

For runtime comparisons we used a standard laptop computer with 2.8 GHz and 16GB RAM.

A. Determine Matches in Synthetic Time Series

To investigate under which conditions our proposed model **fDTW** and others perform best, we test them independently from heuristic pattern matching algorithms, that is we only compare complete alignments of two time series **x** and **h** for models incorporating DTW and selected the lowest CDL for linear alignments (which is similar to PREMDL). With the help of the MDL framework we decide if two time series are either 'not close' or 'close' and assign them to each other. For traditional DTW and UCR Suite we need to set a threshold that separates 'not close' from 'close'. We simulate couples of discretized N(0,1) Markov Chain random walk time series (**x**, **h**), where $x_0 = 0$, $x_i = x_{i-1} + z$ and $z \sim N(0, 1)$. The hypothesis **h** is either independently simulated from **x**, or simulated as a noisy and warped instance of **x**. We varied the simulation parameters of $n = |\mathbf{h}|$, $\sigma =$ standard deviation of the overlaid noise and the amount of warp. In this synthetic case, we do not consider the costs of the hypothesis, since the models are supposed to be applied to find multiple matches to one hypothesis. The costs of each hypothesis would be compared to the total savings of all found fits per hypothesis.

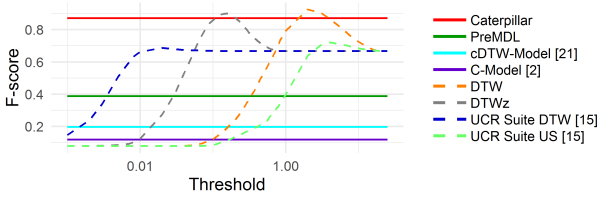


Figure 5: F-scores per model and threshold for detecting whether two time series are similar or not.

Figure 5 shows different models and the according F-scores of the classification task whether one **x** matches the according **h** or not. The Caterpillar (the fDTW-model respectively) performs almost as good as a classifier applying DTW and setting the best threshold (DTWz performs z-normalisation before the distance calculation). The low levels of the F1-scores of PREMDL and the cardinality based models – C-model [2] and cDTW-model [21] – are due to a low recall. The models UCR Suite DTW [15], PREMDL and C-Model cannot fit time series of different lengths, so they scan the longer time series for the best fit. Figure 6 plots the F-scores per

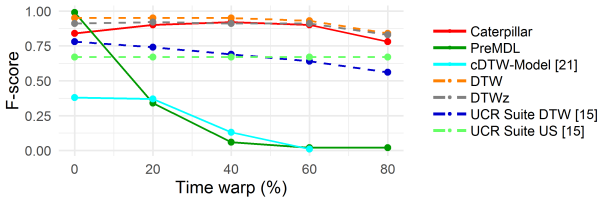


Figure 6: F-scores per model and degree of warping for complete alignments of synthetic time series

amount of time warp. For the methods requiring identifying a distance threshold we set the threshold equal to values showing best performance in Fig. 5. In case of no time warp, the PREMDL performs best, however already at a low level of 10% warping the performance drops from 96% to 72%. As expected, the models relying on DTW perform much better with F1-scores above 90%, however the cDTW-model cannot compete with DTW, Suite DTW and the Caterpillar. The C-Model does not identify any matches in case of warped time series, leading to F1-scores of NaN, due to division by 0. The Caterpillar performs almost as good as the basic DTW (with the best threshold set) with F1-scores between 90% and 96%. The performance of UCR Suite US is indendent from the warping extend which is because the nature of US. The accepted percentage of time warp certainly depends on the application. As Figure 1 demonstrates a warping to the extent of 56% ($= 28/50$ sec) is reasonable.

B. Scan for multiple Matches in Synthetic Time Series

We simulate **x** as a concatenation of multiple warped and noisy instances of **h** and noise-like random walks. Since we know the points in time when the patterns start and end, we evaluate the techniques by counting mismatches and measuring the relative distances of detected fits to their ground truth instance. Figure. 7 and Fig. 8 show the performance and computation time. The Caterpillar (8.7% deviation and 6.7% misses) outperforms the other methods, where PREMDL is the second best (23.2% deviation and 18.5% misses).

Since UCR Suite is designed to find the (single) segment in a long time series best matching a given query pattern, we need to invoke it multiple times to find multiple fits. It must be stressed that UCR Suite (DTW and US) could be much faster for finding multiple fits if it were adapted to detect the k nearest segments instead of one, or alternatively all fits beyond a predefined *threshold*. However, such an adaption still would

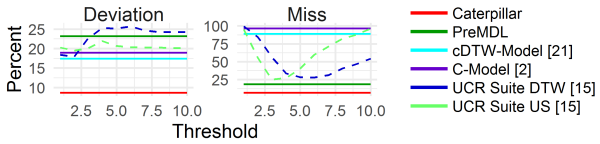


Figure 7: Errors for finding multiple fits in synthetic data

leave the open question of identifying suitable parameters. In addition, while the computation time would be lower, the found fits would be the same. We leave it for future work to test computation times of such a non-trivial adapted version of UCR Suite. Caterpillar is the most accurate method for accurately detecting multiple fits and their extensions. The cDTW- Model is the second best method in terms of deviation, but misses many matches. As for PreMDL, the opposite holds. Our algorithm clearly outperforms both methods at a minor runtime overhead. For comparison we implemented the brute force method of UCR Suite US [15], so the original algorithm applying enveloping techniques would be even faster, however the error rates are worse than most other methods here.

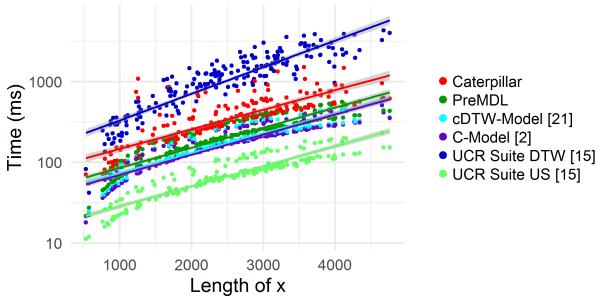


Figure 8: Run times for finding multiple fits in synthetic data.

C. Smartphone Accelerometer Data

To demonstrate the broad applicability of the Caterpillar we use it to detect transport mode specific patterns. We use data collected in an urban area by ten participants equipped with a mobile application on their private smartphones to collect acceleration sensor data. In total the participants collected about 50 hours of unimodal trips of different transport modes (bicycle 7 hours, bus 1, car 3, metro 14, train 4, tram 7, walk 15) and annotated their trips live via the app. For preprocessing, we extracted the horizontal and vertical component of the collected signal similar to [6], [12], [14] and concentrate further only on the horizontal signal. Figure 1b depicts the hypothesis signal **h** we extracted from a scripted trip that represents a typical pattern of a metro that brakes, stops and accelerates again. As Figure 10 shows, such patterns vary vertically and in time extension. Depending on the circumstances (the metro driver, the distance between two stations, the number of passengers, etc.), the metro's acceleration and braking pattern may last longer or shorter. Also, metros do not stop for exactly the same time. Our proposed method still finds these patterns in the acceleration time series.

To evaluate our proposed method, we need a setup with known ground truth. A trip is classified as a metro trip

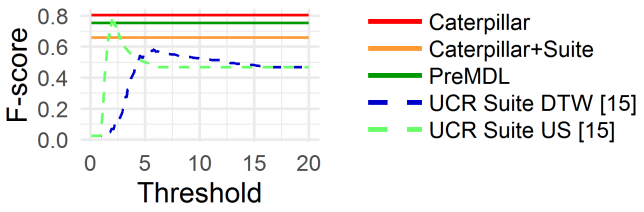


Figure 9: F-scores for classifying accelerometer time series as metro or non-metro

if it contains at least one fit to the hypothesis. Figure 9 demonstrates that the Caterpillar performs best in detecting these metro-typical patterns and achieves an F-score of 80%, which is 5% better than PREMDL and 22% better than UCR Suite DTW (setting the best threshold). Especially UCR Suite DTW has a high recall of 96%, and a low precision of 41%. Setting the ideal threshold UCR Suite US is almost as good as the Caterpillar with an F1-score of 77%. The performance of the UCR Suite methods (dashed lines) highly depends on setting the right threshold. We used UCR Suite in its original form to detect single fits and in the described adapted form to detect multiple fits (see Sec. V-B), both on the original data and discretized time series. The F1-scores for these four variations vary for about 2%, and Fig. 9 shows the best, the multiple Suite DTW on the discrete data with an F1-score of 58%. We also used Suite to detect initials for the Caterpillar (Caterpillar+Suite in Fig. 9), and achieved an F1-score of 66%. We also used (not plotted) only those fits found by Suite below the threshold that achieves the best performance for Suite itself (5.8), and achieved an F1-score of 62%.

To illustrate the dilemma of setting a valid threshold, Figure 10 zooms in the first few minutes of the trip data introduced in Fig. 1. The initial grey segment in Fig. 10d with a US distance of 5 to the query pattern resembles the query pattern obviously much better than the following three segments with US distances of 2, 4, and 3 which show rather random patterns. The two light blue segments starting at seconds 200 and 280 show low distances and visually reasonable matches. Also for the fits found by DTW in Fig. 10c, the distances show a counterintuitive picture. The purple segment at second 280 has a DTW distance of 8 and resembles the query pattern better than the purple segment at second 40 (distance 8) or the light blue second 70 (distance 4). We conclude that it is difficult to set a threshold that separates good from bad fits. The Caterpillar fuses DTW and the MDL principle to avoid such sensitive parameters and performs much better on these data. Moreover the Caterpillar is capable to detect fits of different lengths, whereas those of PREMDL and UCR Suite DTW all have the same length as the query pattern. UCR Suite US also detects fits of different lengths, and performs almost as good as the Caterpillar on the accelerometer data, but as Sec. V-B shows worse on random walk-like synthetic data.

VI. Related Work and Discussion

The MDL principle has been used for controlling the model complexity in different areas (e.g. bioinformatics [20], text

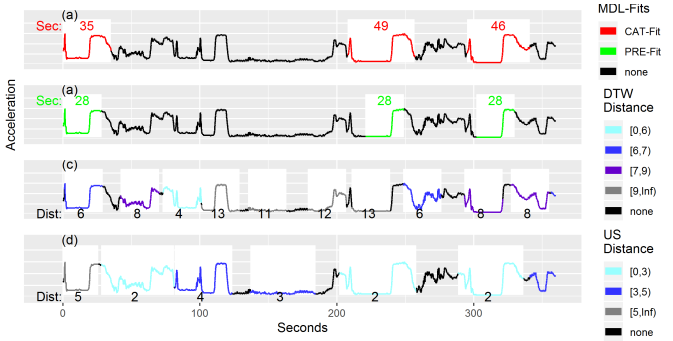


Figure 10: Zoomed version of Fig. 1. Detected fits of Caterpillar (a), PREMDL (b), UCR Suite DTW (c) and UCR Suite US (d)

mining [5], etc.) In time series analysis [2], [21], [7], [16], [22], the MDL principle is useful to answer the question 'Is a given time series **x** close enough to another **h** to consider them as a match?'. To answer this question, the DL of one time series is compared to its CDL given another time series (as applied in Section V-A). In general two strategies exist in the literature to formulate encoding models for the DL and CDL based on MDL. The first approach is to use the cardinality of time series to model the DL [2], [21], [7]. Here the CDL only depends on the number of deviations of two time series, but not the magnitude of the deviations. The second approach is to model the DL and CDL via entropy with Huffman coding [16], [22]. In this work we combine both approaches and demonstrate our method to outperform similar approaches and to be widely applicable.

DTW has been used in many works for time series analysis [3], [9], [4]. There are different approaches how to speed up or early abandon the DTW calculation ([9], [1], [18], [10], [19]) which are all combinable with the here proposed fDTW-model for complete matches of time series (as in Sec. V-A). We leave it for future work to investigate if the Caterpillar algorithm could further benefit from early abandoning or lower bounding.

One of the most relevant works in the field of time series matching is the presentation of UCR Suite [15], that facilitates fast comparisons of many time series or scanning of longer time series for query patterns. However UCR Suite DTW only detects matches of the same length as the query pattern, in contrast to our proposed method, the Caterpillar algorithm, and UCR Suite US that we also benchmark against. Due to selecting initials at the cost of O(n), incremental DTW calculation and recycling of former results we save computation time. The Caterpillar efficiently and reliably detects multiple warped fits in a long time series. Relying on the MDL Principle our algorithm automatically fulfils this task without requring the user to specify input parameters which are difficult to set.

The work in [1] applies the incremental calculation of DTW and also proves it to be reversible to use it for anticipatory pruning. We use these insights and complete the ways of piecewise DTW calculations for alterations of past observations: reverse in-/decremental calculations.

In contrast to [8], [16], [15] we normalize and discretize time series as a whole (see 2), because of the different nature of methods relying on sliding windows of equal lengths and the here proposed incremental comparison of segments.

More specifically, for an incremental step of the Caterpillar, the parameters for normalization (mean, standard deviation, min, max) either would need to be redefined for $\mathbf{x}^+$, but normalizing $\mathbf{x}^+$ with different parameters than $\mathbf{x}$ implicates that former results of the DTW calculations can not be recycled, and DTW (or the fDTW-model) needs to be recalculated from scratch. This would make the problem of finding multiple fits of different lengths almost unsolvable in reasonable time. On the other, hand if the parameters are reused for $\mathbf{x}^+$, we could run out of range: e.g. let $\mathbf{h}$=(2,2,2,4) and $\mathbf{x}$=(1,1,1,2,4), and we start with an initial warping of $\mathbf{h}$ and the first four elements of $\mathbf{x}$, $\mathbf{x}_{1:4}$. For the incremental step we test to align the last element of $\mathbf{x}$, 4. If we normalize $\mathbf{x}_{1:5}$ according to (2) with the same parameters as $\mathbf{x}_{1:4}$, the cost matrix of DTW has entries beyond the range of P , and cannot be encoded with the here proposed encoding scheme. We leave it for future work to investigate possible adjustments of the Caterpillar with running normalization for streaming data that follows no physical constraints (as e.g. accelerometer records of smartphones).

Further we demonstrate to outperform UCR Suite, relying on segmentwise normalization, on accelerometer data and synthetic data. The Caterpillar also allows vertical shifts and can compensate a wandering baseline, such that the two patterns (1,2,3,3,2,4) and (11,12,13,13,12,14) have the same CDL to a query pattern (2,3,4,4,3,5), due to (7).

VII. Conclusion and outlook

In this work we presented a novel encoding scheme for time series applying the MDL framework. We compared the proposed model against state of the art methods on synthetic data and further we applied the model for our proposed Caterpillar algorithm that enables to identify vertically shifted and time warped matches of different lengths of hypothesis time series. Finally we demonstrated the Caterpillar to identify metro stops in accelerometer time series. For future work we plan to investigate the potential of DTW pruning methods for the Caterpillar algorithm and test adjustments of the Caterpillar for streaming data as e.g. financial time series.

References

[1] I. Assent, M. Wichterich, R. Krieger, H. Kremer, and T. Seidl. Anticipatory dtw for efficient similarity search in time series databases. *Proc. VLDB Endow.*, 2(1):826–837, Aug. 2009.

[2] N. Begum, B. Hu, T. Rakthanmanon, and E. Keogh. Towards a minimum description length based stopping criterion for semi-supervised time series classification. In *2013 IEEE 14th International Conference on Information Reuse Integration (IRI)*, pages 333–340, Aug 2013.

[3] D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, AAAIWS'94, pages 359–370. AAAI Press, 1994.

[4] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proceedings of the VLDB Endowment*, 1(2):1542–1552, 2008.

[5] P. D. Grnwald, I. J. Myung, and M. A. Pitt. *Advances in Minimum Description Length: Theory and Applications (Neural Information Processing)*. The MIT Press, 2005.

[6] S. Hemminki, P. Nurmi, and S. Tarkoma. Accelerometer-based transportation mode detection on smartphones. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, SenSys '13, pages 13:1–13:14, New York, NY, USA, 2013. ACM.

[7] B. Hu, T. Rakthanmanon, Y. Hao, S. Evans, S. Lonardi, and E. Keogh. Discovering the intrinsic cardinality and dimensionality of time series using mdl. In *2011 IEEE 11th International Conference on Data Mining*, pages 1086–1091, Dec 2011.

[8] E. Keogh and J. Lin. Clustering of time-series subsequences is meaningless: implications for previous and future research. *Knowledge and information systems*, 8(2):154–177, 2005.

[9] E. Keogh and C. A. Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and Information Systems*, 7(3):358–386, Mar 2005.

[10] E. Keogh, L. Wei, X. Xi, S.-H. Lee, and M. Vlachos. Lb_keogh supports exact indexing of shapes under rotation invariance with arbitrary representations and distance measures. In *Proceedings of the 32nd international conference on Very large data bases*, pages 882–893. VLDB Endowment, 2006.

[11] M. Leodolter. R-package for incremental dynamic time warping, July 2017. https://cran.r-project.org/web/packages/IncDTW/index.html.

[12] M. Leodolter, P. Widhalm, C. Plant, and N. Brandle. Semi-supervised segmentation of accelerometer time series for transport mode classification. In *2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, pages 663–668, June 2017.

[13] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, DMKD '03, pages 2–11, New York, NY, USA, 2003. ACM.

[14] H. Lu, J. Yang, Z. Liu, N. D. Lane, T. Choudhury, and A. T. Campbell. The jigsaw continuous sensing engine for mobile phone applications. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, SenSys '10, pages 71–84, New York, NY, USA, 2010. ACM.

[15] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh. Addressing big data time series: Mining trillions of time series subsequences under dynamic time warping. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 7(3):10, 2013.

[16] T. Rakthanmanon, E. J. Keogh, S. Lonardi, and S. Evans. Mdl-based time series clustering. *Knowledge and Information Systems*, 33(2):371–399, 2012.

[17] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, Feb 1978.

[18] Y. Sakurai, M. Yoshikawa, and C. Faloutsos. Ftw: fast similarity search under the time warping distance. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 326–337. ACM, 2005.

[19] S. Salvador and P. Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.

[20] W. Szpankowski, W. Ren, and L. Szpankowski. An optimal dna segmentation based on the mdl principle. *International Journal of Bioinformatics Research and Applications*, 1(1):3–17, 2005. PMID: 18048118.

[21] V. Thanh Vinh and D. Tuan Anh. *Some Novel Improvements for MDL-Based Semi-supervised Classification of Time Series*, pages 483–493. Springer International Publishing, Cham, 2014.

[22] V. T. Vinh and D. T. Anh. Constraint-based mdl principle for semi-supervised classification of time series. In *2015 Seventh International Conference on Knowledge and Systems Engineering (KSE)*, pages 43–48, Oct 2015.