Pattern Recognition 74 (2018) 171–184

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

shapeDTW: Shape Dynamic Time Warping

Jiaping Zhao[*], Laurent Itti

University of Southern California, USA

ARTICLE INFO

Article history:
Received 17 October 2016
Revised 5 August 2017
Accepted 12 September 2017
Available online 14 September 2017

Keywords:
Dynamic Time Warping
Sequence alignment
Time series classification

ABSTRACT

Dynamic Time Warping (DTW) is an algorithm to align temporal sequences with possible local non-linear distortions, and has been widely applied to audio, video and graphics data alignments. DTW is essentially a point-to-point matching method under some boundary and temporal consistency constraints. Although DTW obtains a global optimal solution, it does not necessarily achieve locally sensible matchings. Concretely, two temporal points with entirely dissimilar local structures may be matched by DTW. To address this problem, we propose an improved alignment algorithm, named shape Dynamic Time Warping (shapeDTW), which enhances DTW by taking point-wise local structural information into consideration. shapeDTW is inherently a DTW algorithm, but additionally attempts to pair locally similar structures and to avoid matching points with distinct neighborhood structures. We apply shapeDTW to align audio signal pairs having ground-truth alignments, as well as artificially simulated pairs of aligned sequences, and obtain quantitatively much lower alignment errors than DTW and its two variants. When shapeDTW is used as a distance measure in a nearest neighbor classifier (NN-shapeDTW) to classify time series, it beats DTW on 64 out of 84 UCR time series datasets, with significantly improved classification accuracies. By using a properly designed local structure descriptor, shapeDTW improves accuracies by more than 10% on 18 datasets. To the best of our knowledge, shapeDTW is the first distance measure under the nearest neighbor classifier scheme to significantly outperform DTW, which had been widely recognized as the best distance measure to date. Our code is publicly accessible at: https://github.com/jiapingz/shapeDTW.

1. Introduction

Dynamic Time Warping (DTW) is an algorithm to align temporal sequences, which has been widely used in speech recognition [1], human motion animation [2], human activity recognition [3] and time series classification [4]. DTW allows temporal sequences to be locally shifted, contracted and stretched, and under some boundary and monotonicity constraints, it searches for a global optimal alignment path. DTW is essentially a point-to-point matching algorithm, but it additionally enforces temporal consistencies among matched point pairs. If we distill the matching component from DTW, the matching is executed by checking the similarity of two points based on their Euclidean distance. Yet, matching points based solely on their coordinate values is unreliable and prone to error, therefore, DTW may generate perceptually nonsensible alignments, which wrongly pair points with distinct local structures (see Fig. 1 (c)). This partially explains why the nearest neighbor classifier under the DTW distance measure is less interpretable than the shapelet classifier [5]: although DTW does achieve a global minimal score, the alignment process itself takes no local structural information into account, possibly resulting in an alignment with little semantic meaning. In this paper, we propose a novel alignment algorithm, named shape Dynamic Time Warping (shapeDTW), which enhances DTW by incorporating point-wise local structures into the matching process. As a result, we obtain perceptually interpretable alignments: similarly-shaped structures are preferentially matched based on their degree of similarity. We further quantitatively evaluate alignment paths against the ground-truth, and shapeDTW achieves much lower alignment errors than DTW. An alignment example by shapeDTW is shown in Fig. 1 (d).

Point matching is a well studied problem in the computer vision community, widely known as image matching. In order to search corresponding points from two distinct images taken from the same scene, a quite naive way is to compare their pixel values. But pixel values at a point lacks spatial neighborhood context, making it less discriminative for that point; e.g., a tree leaf pixel from one image may have exactly the same RGB values as a grass pixel from the other image, but these two pixels are not corresponding pixels and should not be matched. Therefore, a routine for image matching is to describe points by their surrounding image patches, and then compare the similarities of point descriptors.

[*] Corresponding author.
E-mail addresses: jiapingz@usc.edu, justinzo@hotmail.com (J. Zhao), itti@usc.edu (L. Itti).

http://dx.doi.org/10.1016/j.patcog.2017.09.020
0031-3203/© 2017 Elsevier Ltd. All rights reserved.

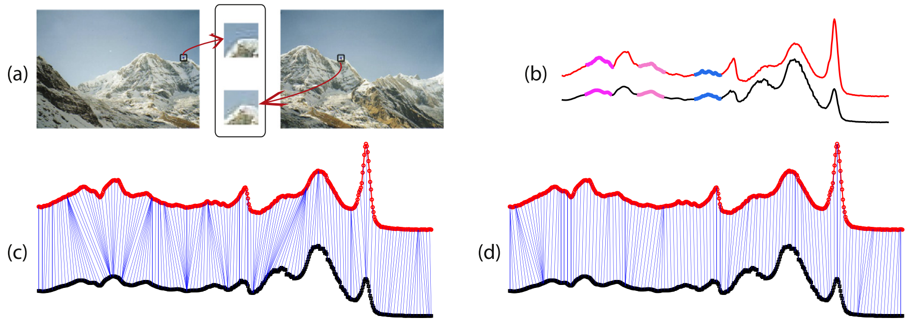


Fig. 1. Motivation to incorporate temporal neighborhood structural information into the sequence alignment process. (a) an image matching example: two corresponding points from the image pairs are boxed out and their local patches are shown in the middle. Local patches encode image structures around spatial neighborhoods, and therefore are discriminative for points, while it is hard to match two points solely by their pixel values. (b) two time series with several similar local structures, highlighted as bold segments. (c) DTW alignment: DTW fails to align similar local structures. (d) shapeDTW alignment: we achieve a more interpretable alignment, with similarly-shaped local structures matched.

Since point descriptors designed in this way encode image structures around local neighborhoods, they are more distinctive and discriminative than single pixel values. In early days, raw image patches were used as point descriptors [6], and now more powerful descriptors like SIFT [7] are widely adopted since they capture local image structures very well and are invariant to image scale and rotation.

Intuitively, local neighborhood patches make points more discriminative from other points, while matching based on RGB pixel values is brittle and results in high false positives. However, the matching component in the traditional DTW bears the same weakness as image matching based on single pixel values, since similarities between temporal points are measured by their coordinates, instead of by their local neighborhoods. An analogous remedy for temporal matching hence is: first encode each temporal point by some descriptor, which captures local subsequence structural information around that point, and then match temporal points based on the similarity of their descriptors. If we further enforce temporal consistencies among matchings, then comes the algorithm proposed in the paper: shapeDTW.

shapeDTW is a temporal alignment algorithm, which consists of two sequential steps: (1) represent each temporal point by some shape descriptor, which encodes structural information of local subsequences around that point; in this way, the original time series is converted into a sequence of descriptors. (2) use DTW to align two sequences of descriptors. Since the first step takes linear time while the second step is a typical DTW, which takes quadratic time, the total time complexity is quadratic in the length of time series, indicating that shapeDTW has similar computational complexities as DTW. However, compared with DTW and its variants (derivative Dynamic Time Warping (dDTW) [8] and weighted Dynamic Time Warping (wDTW) [9]), it has two clear advantages: (1) shapeDTW obtains lower alignment errors than DTW/dDTW/wDTW on both artificially simulated aligned sequence pairs and real audio signals; (2) the nearest neighbor classifier under the shapeDTW distance measure (NN-shapeDTW) significantly beats NN-DTW on 64 out of 84 UCR time series datasets [4]. NN-shapeDTW outperforms NN-dDTW/NN-wDTW significantly as well. Our shapeDTW time series alignment procedure is shown in Fig. 2.

Extensive empirical experiments have shown that a nearest neighbor classifier with the DTW distance measure (NN-DTW) is the best choice to date for most time series classification problems, since no alternative distance measures outperforms DTW significantly [10–12]. However, in this paper, the proposed temporal alignment algorithm, shapeDTW, if used as a distance measure under the nearest neighbor classifier scheme, significantly beats DTW. To the best of our knowledge, shapeDTW is the first distance measure that outperforms DTW significantly.

Our contributions are several fold: (1) we propose a temporal alignment algorithm, shapeDTW, which achieves quantitatively better alignments than DTW (dDTW, wDTW); (2) Working under the nearest neighbor classifier as a distance measure to classify 84 UCR time series datasets, shapeDTW, under all tested shape descriptors, outperforms DTW significantly; (3) shapeDTW provides a quite generic alignment framework, and users can design new shape descriptors adapted to their domain data characteristics and then feed them into shapeDTW for alignments.

2. Related work

Since shapeDTW is developed for sequence alignment, here we first review work related to sequence alignment. DTW is a typical sequence alignment algorithm, and there are many ways to improve DTW to obtain better alignments. Traditionally, we could enforce global warping path constraints to prevent pathological warpings [1], and several typical such global warping constraints include Sakoe–Chiba band and Itakura Parallelogram. Similarly, we could choose to use different step patterns in different applications: apart from the widely used step pattern - “symmetric1”, there are other popular steps patterns like “symmetric2”, “asymmetric” and “RabinerJuangStepPattern” [13]. However, how to choose an appropriate warping band constraint and a suitable step pattern depends on our prior knowledge on the application domains.

There are several recent works to improve DTW alignment. In [8], to get the intuitively correct “feature to feature” alignment between two sequences, the authors introduced derivative Dynamic Time Warping (dDTW), which computes first-order derivatives of time series sequences, and then aligns two derivative sequences by DTW. Jeong et al. [9] developed weighted DTW (wDTW), which is a penalty-based DTW. wDTW takes the phase difference between two points into account when computing their distances. Batista et al. [14] proposed a complexity-invariant distance measure, which essentially rectifies an existing distance measure (e.g., Euclidean, DTW) by multiplying a complexity correction factor. Although they achieve improved results on some datasets by rectifying the DTW measure, they do not modify the original DTW algorithm. Lajugie et al. [15] proposed to learn a distance metric, and then align temporal sequences by DTW under this new metric. One major drawback is the requirement of ground truth alignments for metric learning, because in reality true alignments are usually unavailable. Candan et al. [16] proposed to utilize time series local structure information to constrain the search of the warping path. They introduce a SIFT-like feature point detector and descriptor to detect and match salient feature points from two sequences first, and then use matched point pairs to regularize the

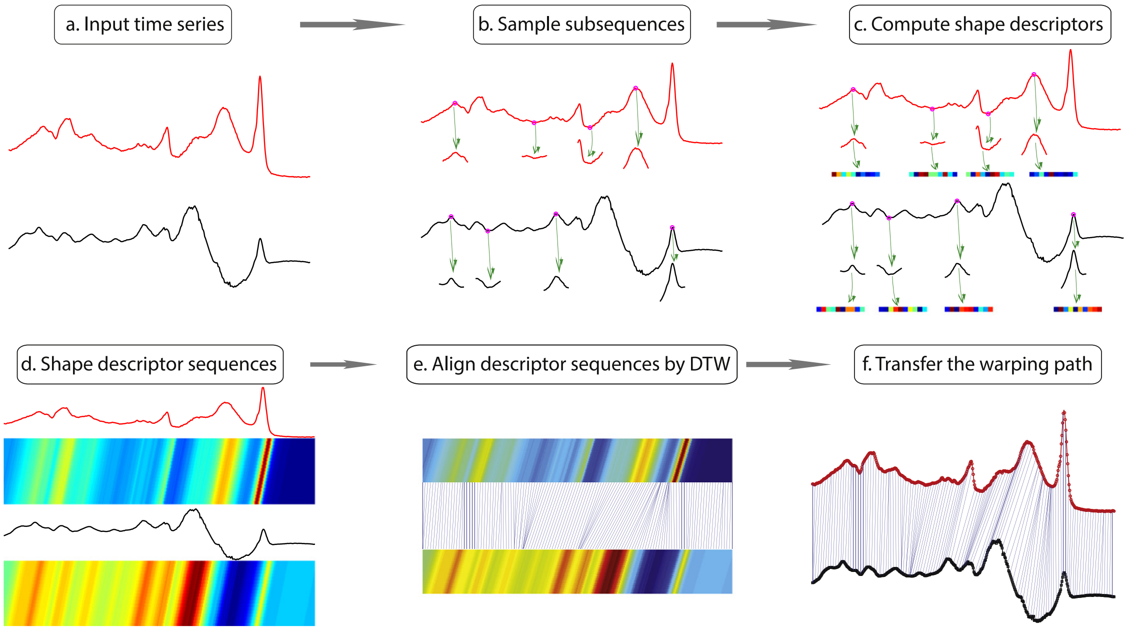

a. Input time series
b. Sample subsequences
c. Compute shape descriptors
d. Shape descriptor sequences
e. Align descriptor sequences by DTW
f. Transfer the warping path

Fig. 2. Pipeline of shapeDTW. shapeDTW consists of two major steps: encode local structures by shape descriptors and align descriptor sequences by DTW. Concretely, we sample a subsequence from each temporal point, and further encode it by some shape descriptor. As a result, the original time series is converted into a descriptor sequence of the same length. Then we align two descriptor sequences by DTW and transfer the found warping path to the original time series.

search scope of the warping path. Their major initiative is to improve the computational efficiency of Dynamic Time Warping by enforcing band constraints on the potential warping paths, such that they do not have to compute the full accumulative distance matrix between the two sequences. Our method is sufficiently different from theirs in following aspects: first, we have no notion of feature points, while feature points are key to their algorithm, since feature points help to regularize downstream DTW; second, our algorithm aims to achieve better alignments, while their algorithm attempts to improve the computational efficiency of the traditional DTW. Petitjean et al. [12] focus on improving the efficiency of the nearest neighbor classifier under the DTW distance measure, but they keep the traditional DTW algorithm unchanged.

Our algorithm, shapeDTW, is different from the above works in that: we measure similarities between two points by computing similarities between their local neighborhoods, while all the above works compute the distance between two points based on their single-point y-values (derivatives).

Since shapeDTW can be applied to classify time series (e.g., NN-shapeDTW), we review representative time series classification algorithms. Lin et al. [17] use the popular Bag-of-Words to represent time series instances, and then classify the representations under the nearest neighbor classifier. Concretely, it discretizes time series into local SAX [18] words, and uses the histogram of SAX words as the time series representation. Rakthanmanon and Keogh [19] developed an algorithm to first extract class-membership discriminative shapelets, and then learn a decision tree classifier based on distances between shapelets and time series instances. In [20], they first represent time series using recurrent plots, and then measure the similarity between recurrence plots using Campana–Keogh (CK-1) distance (PRCD). PRCD distance is used as the distance measure under the one-nearest neighbor classifier to do classification. In [21], a bag-of-feature framework to classify time series is introduced. It uses a supervised codebook to encode time series instances, and then uses random forest classifier to classify the encoded time series. Grabocka and Schmidt-Thieme [22] first encode time series as a bag-of-patterns, and then use polynomial kernel SVM to do the classification. Zhao and Itti [23] proposed to first encode time series by the 2nd order encoding method - Fisher Vectors, and then classify encoded time series by a linear kernel SVM. In their paper, subsequences are sampled from both feature points and flat regions.

shapeDTW is different from above works in that: shapeDTW is developed to align temporal sequences, but can be further applied to classify time series. However, all above works are developed to classify time series, and they are incapable to align temporal sequences at their current stages. Since time series classification is only one application of shapeDTW, we compare NN-shapeDTW against the above time series classification algorithms in the supplementary materials.

The paper is organized as follows: the detailed algorithm for shapeDTW is introduced in Section 3, and in Section 4 we introduce several local shape descriptors. Then we extensively test shapeDTW for both sequence alignments and time series classification in Section 6, and conclusions are drawn in Section 7.

3. shape Dynamic Time Warping

In this section, we introduce a temporal alignment algorithm, shapeDTW. First we introduce DTW briefly.

3.1. Dynamic Time Warping

DTW is an algorithm to search for an optimal alignment between two temporal sequences. It returns a distance measure for gauging similarities between them. Sequences are allowed to have local non-linear distortions in the time dimension, and DTW handles local warpings to some extent. DTW is applicable to both univariate and multivariate time series, and here for simplicity we introduce DTW in the case of univariate time series alignment.

A univariate time series is a sequence of real values, i.e., $\mathcal{T} = (t_1, t_2, \dots, t_L)^T$. Given two sequences $\mathcal{P}$ and $\mathcal{Q}$ of possibly different lengths $\mathcal{L}_\mathcal{P}$ and $\mathcal{L}_\mathcal{Q}$, namely $\mathcal{P} = (p_1, p_2, \dots, p_{\mathcal{L}_\mathcal{P}})^T$ and $\mathcal{Q} = (q_1, q_2, \dots, q_{\mathcal{L}_\mathcal{Q}})^T$, and let $\mathcal{D}(\mathcal{P}, \mathcal{Q}) \in \mathcal{R}^{\mathcal{L}_\mathcal{P} \times \mathcal{L}_\mathcal{Q}}$ be an pairwise distance matrix between sequences $\mathcal{P}$ and $\mathcal{Q}$, where $\mathcal{D}(\mathcal{P}, \mathcal{Q})_{i,j}$ is the distance between p_i and p_j. One widely used pairwise distance measure is the Euclidean distance, i.e., $\mathcal{D}(\mathcal{P}, \mathcal{Q})_{i,j} = |p_i - q_j|$.

The goal of temporal alignment between $\mathcal{P}$ and $\mathcal{Q}$ is to find two sequences of indices α and β of the same length $l\,(l \geq \max(\mathcal{L}_\mathcal{P}, \mathcal{L}_\mathcal{Q}))$, which match index $\alpha(i)$ in the time series $\mathcal{P}$ to index $\beta(i)$ in the time series $\mathcal{Q}$, such that the total cost along the matching path $\sum_{i=1}^{l} \mathcal{D}(\mathcal{P}, \mathcal{Q})_{\alpha(i),\beta(i)}$ is minimized. The alignment path $(\alpha,\ \beta)$ is constrained to satisfy boundary, monotonicity and continuity conditions [24–26]:

$$\begin{cases} \alpha(1) = \beta(1) = 1 \\ \alpha(l) = \mathcal{L}_\mathcal{P},\ \beta(l) = \mathcal{L}_\mathcal{Q} \\ (\alpha(i+1), \beta(i+1)) - (\alpha(i), \beta(i)) \in \{(1,0), (1,1), (0,1)\} \end{cases} \tag{1}$$

Given an alignment path $(\alpha,\ \beta)$, we define two warping matrices $\mathcal{W}^\mathcal{P} \in \{0,1\}^{l \times \mathcal{L}_\mathcal{P}}$ and $\mathcal{W}^\mathcal{Q} \in \{0,1\}^{l \times \mathcal{L}_\mathcal{Q}}$ for $\mathcal{P}$ and $\mathcal{Q}$ respectively, such that $\mathcal{W}^\mathcal{P}(i, \alpha(i)) = 1$, otherwise $\mathcal{W}^\mathcal{P}(i,j) = 0$, and similarly $\mathcal{W}^\mathcal{Q}(i, \beta(i)) = 1$, otherwise $\mathcal{W}^\mathcal{Q}(i,j) = 0$. Then the total cost along the matching path $\sum_{i=1}^{l} \mathcal{D}(\mathcal{P}, \mathcal{Q})_{\alpha(i),\beta(i)}$ is equal to $\parallel \mathcal{W}^\mathcal{P} \cdot \mathcal{P} - \mathcal{W}^\mathcal{Q} \cdot \mathcal{Q} \parallel_1$, thus searching for the optimal temporal matching can be formulated as the following optimization problem:

$$\arg\min_{l,\ \mathcal{W}^\mathcal{P} \in \{0,1\}^{l \times \mathcal{L}_\mathcal{P}},\ \mathcal{W}^\mathcal{Q} \in \{0,1\}^{l \times \mathcal{L}_\mathcal{Q}}} \parallel \mathcal{W}^\mathcal{P} \cdot \mathcal{P} - \mathcal{W}^\mathcal{Q} \cdot \mathcal{Q} \parallel_1 \tag{2}$$

Program 2 can be solved efficiently in $\mathcal{O}(\mathcal{L}_\mathcal{P} \times \mathcal{L}_\mathcal{Q})$ time by a dynamic programming algorithm [27]. Various different moving patterns and temporal window constraints [24] can be enforced, but here we consider DTW without warping window constraints and taking moving patterns as in (1).

Note: in this section, DTW is defined in the case of univariate time series, therefore is formulated under the ℓ_1 norm (ℓ_1 norm is equivalent to ℓ_p norm in the scalar space). For a more generalized definition of DTW under the ℓ_p norm, see [28].

3.2. shape Dynamic Time Warping

DTW finds a global optimal alignment under certain constraints, but it does not necessarily achieve locally sensible matchings. Here we incorporate local shape information around each point into the dynamic programming matching process, resulting in more semantically meaningful alignment results, i.e., points with similar local shapes tend to be matched while those with dissimilar neighborhoods are unlikely to be matched. shapeDTW consists of two steps: (1) represent each temporal point by some shape descriptor; and (2) align two sequences of descriptors by DTW. We first introduce the shapeDTW alignment framework, and in the next section, we introduce several local shape descriptors.

Given a univariate time series $\mathcal{T} = (t_1, t_2, \ldots, t_L)^T, \mathcal{T} \in \mathcal{R}^L$, shapeDTW begins by representing each temporal point t_i by a shape descriptor $d_i \in \mathcal{R}^m$, which encodes structural information of temporal neighborhoods around t_i, in this way, the original real value sequence $\mathcal{T} = (t_1, t_2, \ldots, t_L)^T$ is converted to a sequence of shape descriptors of the same length, i.e., $\mathbf{d} = (d_1, d_2, \ldots, d_L)^T, \mathbf{d} \in \mathcal{R}^{L \times m}$. shapeDTW then aligns the transformed multivariate descriptor sequences $\mathbf{d}$ by DTW, and at last the alignment path between descriptor sequences is transferred to the original univariate time series sequences. We give implementation details of shapeDTW:

Given a univariate time series of length L, e.g.,$\mathcal{T} = (t_1, t_2, \ldots, t_L)^T$, we first extract a subsequence s_i of length l from each temporal point t_i. The subsequence s_i is centered on t_i, with its length l typically much smaller than $L (l \ll L)$. Note we have to pad both ends of $\mathcal{T}$ by $\lfloor \frac{l}{2} \rfloor$ with duplicates of $t_1 (t_L)$ to make subsequences sampled at endpoints well defined. Now we obtain a sequence of subsequences, i.e., $\mathcal{S} = (s_1, s_2, \ldots, s_L)^T$, $s_i \in \mathcal{R}^l$, with s_i corresponding to the temporal point t_i. Next, we design shape descriptors to express subsequences, under the goal that similarly-shaped subsequences have similar descriptors while differently-shaped subsequences have distinct descriptors. The shape descriptor of subsequence s_i naturally encodes local structural information around the temporal point t_i, and is named as shape descriptor of the temporal point t_i as well. Designing a shape descriptor boils down to designing a mapping function $\mathcal{F}(\cdot)$, which maps subsequence $s_i \in \mathcal{R}^l$ to shape descriptor $d_i \in \mathcal{R}^m$, i.e., $d_i = \mathcal{F}(s_i)$, so that similarity between descriptors can be measured simply with the Euclidean distance. Different mapping functions define different shape descriptors, and one straightforward mapping function is the identity function $\mathcal{I}(\cdot)$, in this case, $d_i = \mathcal{I}(s_i) = s_i$, i.e., subsequence itself acts as local shape descriptor. Given a shape descriptor computation function $\mathcal{F}(\cdot)$, we convert the subsequence sequence $\mathcal{S}$ to a descriptor sequence $\mathbf{d} = (d_1, d_2, \ldots, d_L)^T\ d_i \in \mathcal{R}^m$, i.e., $\mathbf{d} = \mathcal{F}(\mathcal{S}) = (\mathcal{F}(s_1), \mathcal{F}(s_2), \ldots, \mathcal{F}(s_L))^T$. At last, we use DTW to align two descriptor sequences and transfer the warping path to the original univariate time series.

Given two univariate time series $\mathcal{P} = (p_1, p_2, \ldots, p_{\mathcal{L}_\mathcal{P}})^T, \mathcal{P} \in \mathcal{R}^{\mathcal{L}_\mathcal{P}}$ and $\mathcal{Q} = (q_1, q_2, \ldots, q_{\mathcal{L}_\mathcal{Q}})^T, \mathcal{Q} \in \mathcal{R}^{\mathcal{L}_\mathcal{Q}}$, let $\mathbf{d}^\mathcal{P} = (d_1^\mathcal{P}, d_2^\mathcal{P}, \ldots, d_{\mathcal{L}_\mathcal{P}}^\mathcal{P})^T$, $d_i^\mathcal{P} \in \mathcal{R}^m$, $\mathbf{d}^\mathcal{P} \in \mathcal{R}^{\mathcal{L}_\mathcal{P} \times m}$ and $\mathbf{d}^\mathcal{Q} = (d_1^\mathcal{Q}, d_2^\mathcal{Q}, \ldots, d_{\mathcal{L}_\mathcal{Q}}^\mathcal{Q})^T$, $d_i^\mathcal{Q} \in \mathcal{R}^m$, $\mathbf{d}^\mathcal{Q} \in \mathcal{R}^{\mathcal{L}_\mathcal{Q} \times m}$ be their shape descriptor sequences, shapeDTW alignment is equivalent to solve the optimization problem:

$$\arg\min_{l,\ \tilde{\mathcal{W}}^\mathcal{P} \in \{0,1\}^{l \times \mathcal{L}_\mathcal{P}}, \tilde{\mathcal{W}}^\mathcal{Q} \in \{0,1\}^{l \times \mathcal{L}_\mathcal{Q}}} \parallel \tilde{\mathcal{W}}^\mathcal{P} \cdot \mathrm{d}^\mathcal{P} - \tilde{\mathcal{W}}^\mathcal{Q} \cdot \mathrm{d}^\mathcal{Q} \parallel_{1,2} \tag{3}$$

Where $\tilde{\mathcal{W}}^\mathcal{P}$ and $\tilde{\mathcal{W}}^\mathcal{Q}$ are warping matrices of $\mathbf{d}^\mathcal{P}$ and $\mathbf{d}^\mathcal{Q}$, and $\parallel \cdot \parallel_{1,\ 2}$ is the ℓ_1/ℓ_2-norm of matrix, i.e., $\parallel \mathcal{M}_{p \times n} \parallel_{1,2} = \sum_{i=1}^{p} \parallel \mathcal{M}_i \parallel_2$, where $\mathcal{M}_i$ is the ith row of matrix $\mathcal{M}$. Program 3 is a multivariate time series alignment problem, and can be effectively solved by dynamic programming in time $\mathcal{O}(\mathcal{L}_\mathcal{P} \times \mathcal{L}_\mathcal{Q})$. The key difference between DTW and shapeDTW is that: DTW measures similarities between p_i and q_j by their Euclidean distance $|p_i - q_j|$, while shapeDTW uses the Euclidean distance between their shape descriptors, i.e., $\parallel d_i^\mathcal{P} - d_j^\mathcal{Q} \parallel_2$, as the similarity measure. shapeDTW essentially handles local non-linear warping, since it is inherently DTW, and, on the other hand, it prefers matching points with similar neighborhood structures to points with similar values. shapeDTW algorithm is described in Algorithm 1.

4. Shape descriptors

shapeDTW provides a generic alignment framework, and users can design shape descriptors adapted to their domain data characteristics and feed them into shapeDTW for alignments. Here we introduce several general shape descriptors, each of which maps a subsequence s_i to a vector representation d_i, i.e., $d_i = \mathcal{F}(s_i)$.

The length l of subsequences defines the size of neighborhoods around temporal points. When $l = 1$, no neighborhood information is taken into account. With increasing l, larger neighborhoods are considered, and in the extreme case when $l = L$ (L is the length of the time series), subsequences sampled from different temporal points become the same, i.e., the whole time series, in which case, shape descriptors of different points resemble each other too much, making temporal points less identifiable by shape descriptors. In practice, l is set to some appropriate value. But in this section, we first let l be any positive integers ($l \geq 1$), which does not affect the definition of shape descriptors. In Section 6, we will experimentally explore the sensitivity of NN-shapeDTW to the choice of l.

4.1. Raw-Subsequence

Raw Subsequence s_i sampled around point t_i can be directly used as the shape descriptor of t_i, i.e., $d_i = \mathcal{I}(s_i) = s_i$, where $\mathcal{I}(\cdot)$ is the identity function. Although simple, it inherently captures the local subsequence shape and helps to disambiguate points with similar values but different local shapes.

Algorithm 1 shape Dynamic Time Warping.

```
Inputs: univariate time series $\mathcal{P} \in \mathcal{R}^{\mathcal{L}_\mathcal{P}}$and $\mathcal{Q} \in \mathcal{R}^{\mathcal{L}_\mathcal{Q}}$; subsequence length $l$; shape descriptor function $\mathcal{F}$

shapeDTW:

1. Sample subsequences: $\mathcal{S}^\mathcal{P} \leftarrow \mathcal{P}$, $\mathcal{S}^\mathcal{Q} \leftarrow \mathcal{Q}$;

2. Encode subsequences by shape descriptors:

$\mathtt{d}^\mathcal{P} \leftarrow \mathcal{F}(\mathcal{S}^\mathcal{P})$, $\mathtt{d}^\mathcal{Q} \leftarrow \mathcal{F}(\mathcal{S}^\mathcal{Q})$;

3. Align descriptor sequences $\mathtt{d}^\mathcal{P}$ and $\mathtt{d}^\mathcal{Q}$by DTW.

Outputs:

warping matrices: $\tilde{W}_*^\mathcal{P}$ and $\tilde{W}_*^\mathcal{Q}$;

shapeDTW distance: $\| \tilde{W}_*^\mathcal{P} \cdot \mathtt{d}^\mathcal{P} - \tilde{W}_*^\mathcal{Q} \cdot \mathtt{d}^\mathcal{Q} \|_{1,2}$
```

4.2. PAA

Piecewise aggregate approximation (PAA) is introduced in [29,30] to approximate time series. Here we use it to approximate subsequences. Given a l-dimensional subsequence s_i, it is divided into m ($m \le l$) equal-lengthed intervals, the mean value of temporal points falling within each interval is calculated and a vector of these mean values gives the approximation of s_i and is used as the shape descriptor d_i of s_i, i.e., $\mathcal{F}(\cdot) = PAA$, $d_i = PAA(s_i)$.

4.3. DWT

Discrete Wavelet Transform (DWT) is another widely used technique to approximate time series instances. Again, here we use DWT to approximate subsequences. Concretely, we use a Haar wavelet basis to decompose each subsequence s_i into 3 levels. The detail wavelet coefficients of all three levels and the approximation coefficients of the third level are concatenated to form the approximation, which is used the shape descriptor d_i of s_i, i.e., $\mathcal{F}(\cdot) = DWT$, $d_i = DWT(s_i)$.

4.4. Slope

All the above three shape descriptors encode local shape information inherently. However, they are not invariant to y-shift, to be concrete, given two subsequences p, q of exactly the same shape, but p is a y-shifted relative to q, e.g., $p = q + \Delta \cdot \mathbf{1}$, where Δ is the magnitude of y-shift, then their shape descriptors under *Raw-Subsequence*, *PAA* and *DWT* differ approximately by Δ as well, i.e., $d(p) \approx d(q) + \Delta \cdot \mathbf{1}$. Although magnitudes do help time series classification, it is also desirable that similarly-shaped subsequences have similar descriptors. Here we further exploit three shape descriptors in experiments, *Slope*, *Derivative* and *HOG1D*, which are invariant to y-shift.

Slope is extracted as a feature and used in time series classification in [21,31]. Here we use it to represent subsequences. Given a l-dimensional subsequence s_i, it is divided into m ($m \le l$) equal-lengthed intervals. Within each interval, we employ the total least square (TLS) line fitting approach [32] to fit a line according to points falling within that interval. By concatenating the slopes of the fitted lines from all intervals, we obtain a m-dimensional vector representation, which is the slope representation of s_i, i.e., $\mathcal{F}(\cdot) = Slope$, $d_i = Slope(s_i)$.

4.5. Derivative

Similar to *Slope*, *Derivative* is y-shift invariant as well for shape representation. Given a subsequence s, its first-order derivative sequence is s', where s' is the first order derivative according to time t. To keep consistent with derivatives used in derivative Dynamic Time Warping [8] (dDTW), we follow their formula to compute numeric derivatives.

4.6. HOG1D

HOG1D is introduced in [33] to represent 1D time series sequences. It inherits key concepts from the histogram of oriented gradients (HOG) descriptor [34], and uses concatenated gradient histograms to represent shapes of temporal sequences. Similarly to *Slope* and *Derivative* descriptors, *HOG1D* is invariant to y-shift as well.

In experiments, we divide a subsequence into 2 non-overlapping intervals, compute gradient histograms (under 8 bins) in each interval and concatenate two histograms as the HOG1D descriptor (a 16D vector) of that subsequence. We refer interested readers to Zhao and Itti [33] for computation details of HOG1D. We have to emphasize that: Zhao and Itti [33] introduce a global scaling factor σ and tune it using all training sequences; but here, we fix σ to be 0.1 in all our experiments, therefore, HOG1D computation on one subsequence takes only linear time $\mathcal{O}(l)$, where l is the length of that subsequence. See our published code for details.

4.7. Compound shape descriptors

Shape descriptors, like *HOG1D*, *Slope* and *Derivative*, are invariant to y-shift. However, in the application of matching two subsequences, y-magnitudes may sometimes be important cues as well, e.g., DTW relies on point-wise magnitudes for alignments. Shape descriptors, like *Raw-Subsequence*, *PAA* and *DWT*, encode magnitude information, thus they complement y-shift invariant descriptors. By fusing pure-shape capturing and magnitude-aware descriptors, the compound descriptor has the potential to become more discriminative of subsequences. In the experiments, we generate compound descriptors by concatenating two complementary descriptors, i.e., $d = (d_A, \gamma d_B)$, where γ is a weighting factor to balance two simple descriptors, and d is the generated compound descriptor.

5. Alignment quality evaluation

Here we adopt the "*mean absolute deviation*" measure used in the audio literature [35] to quantify the proximity between two alignment paths. "*Mean absolute deviation*" is defined as the mean distance between two alignment paths, which is positively proportional to the area between two paths. Intuitively, two spatially proximate paths have small between-areas, therefore low "*Mean absolute deviation*". Formally, given a reference sequence $\mathcal{P}$, a target sequence $\mathcal{Q}$ and two alignment paths α, β between them, the *Mean absolute deviation* between α and β is calculate as: $\delta(\alpha, \beta) = \mathcal{A}(\alpha, \beta)/\mathcal{L}_\mathcal{P}$, where $\mathcal{A}(\alpha, \beta)$ is the area between α and β and $\mathcal{L}_\mathcal{P}$ is the length of the reference sequence $\mathcal{P}$. Fig. 3 shows two alignment paths α, β, blue and red curves, between $\mathcal{P}$ and $\mathcal{Q}$. $\mathcal{A}(\alpha, \beta)$ is the area of the slashed region, and in practice, it is computed by counting the number of cells falling within it. Here a cell (i, j) refers to the position (i, j) in the pairwise distance matrix $\mathcal{D}(\mathcal{P}, \mathcal{Q}) \in \mathcal{R}^{\mathcal{L}_\mathcal{P} \times \mathcal{L}_\mathcal{Q}}$ between $\mathcal{P}$ and $\mathcal{Q}$.

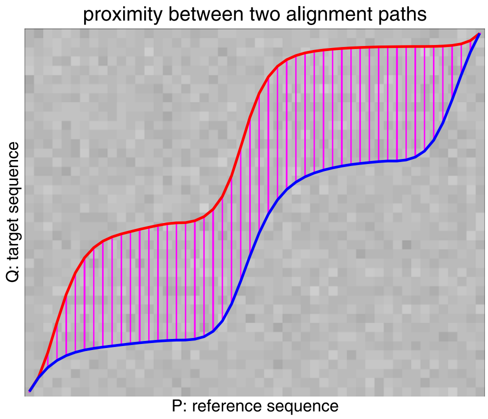


Fig. 3. "*Mean absolute deviation*", which measures the proximity between alignment paths. The red and blue curves are two alignment paths between sequences $\mathcal{P}$ and $\mathcal{Q}$, and "*Mean absolute deviation*" between these two paths is measured as the area of the slashed region divided by the length of the reference sequence. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

6. Experimental validation

We test shapeDTW for sequence alignment and time series classification extensively on 84 UCR time series datasets [4] and the Bach10 dataset [36]. For sequence alignment, we compare shapeDTW against DTW and its other variants both qualitatively and quantitatively: specifically, we first visually compare alignment results returned by shapeDTW and DTW (and its variants), and then quantify their alignment path qualities on both synthetic and real data. Concretely, we simulate aligned pairs by artificially scaling and stretching original time series sequences, align those pairs by shapeDTW and DTW (and its variants), and then evaluate the alignment paths against the ground-truth alignments. We further evaluate the alignment performances of shapeDTW and DTW (and its variants) on audio signals, which have the ground-truth point-to-point alignments. For time series classification, since it is widely recognized that the nearest neighbor classifier with the distance measure DTW (NN-DTW) is very effective and is hard to beat [10,37], we use the nearest neighbor classifier as well to test the effectiveness of shapeDTW (NN-shapeDTW), and compare NN-shapeDTW against NN-DTW. We further compare NN-shapeDTW against six other state-of-the-art classification algorithms in the supplementary materials.

6.1. Sequence alignment

We evaluate sequence alignments qualitatively in Section 6.1.2 and quantitatively in Sections 6.1.3 and 6.1.4. We compare shapeDTW against DTW, derivative Dynamic Time Warping (dDTW) [8] and weighted Dynamic Time Warping (wDTW) [9]. dDTW first computes derivative sequences, and then aligns them by DTW. wDTW uses a weighted ℓ_2 distance, instead of the regular ℓ_2 distance, to compute distances between points, and the weight accounts for the phase differences between points. wDTW is essentially a DTW algorithm. Here, both dDTW and wDTW are variants of the original DTW. Before the evaluation, we briefly introduce some popular step patterns in DTW.

6.1.1. Step pattern in DTW

Step pattern in DTW defines the allowed transitions between matched pairs, and the corresponding weights. In both Programs 2 (DTW) and 3 (shapeDTW), we use the default step pattern, whose recursion formula is $D(i,j) = d(i,j) + \min\{D(i-1,j-1), D(i,j-1), D(i-1,j)\}$. In the following alignment experiments, we try other well-known step patterns as well, and we follow the naming convention in [13] to name these step-patterns. Five popular step-patterns, "symmetric1", "symmetric2", "symmetric5", "asymmetric" and "rabinerJuang", are listed in Fig. 4. Step-pattern (a), "symmetric1", is the one used by shapeDTW in all the following alignment and classification experiments, and we will not explicitly mention that in following texts.

6.1.2. Qualitative alignment assessment

We plot alignment results by shapeDTW and DTW/dDTW, and evaluate them visually. shapeDTW under 5 shape descriptors, *Raw-Subsequence*, *PAA*, *DWT*, *Derivative* and *HOG1D*, obtains similar alignment results, here we choose *Derivative* as a representative to report results, with the subsequence length set to be 30. Here, shapeDTW, DTW and dDTW all use step pattern (a) in Fig. 4.

Time series with rich local features: time series with rich local features, such as those in the "OSUleaf" dataset (bottom row in Fig. 5), have many bumps and valleys; DTW becomes quite brittle to align such sequences, since it matches two points based on their single-point y-magnitudes. Because single magnitude value does not incorporate local neighborhood information, it is hard for DTW to discriminate a peak point $\mathtt{p}$ from a valley point $\mathtt{v}$ with the same magnitude, although $\mathtt{p}$ and $\mathtt{v}$ have dramatically different local shapes. dDTW bears similar weakness as DTW, since it matches points bases on their derivative differences and does not take local neighborhood into consideration either. On the contrary, shapeDTW distinguishes peaks from valleys easily by their highly different local shape descriptors. Since shapeDTW takes both nonlinear warping and local shapes into account, it gives more perceptually interpretable and semantically sensible alignments than DTW (dDTW). Some typical alignment results of time series from feature rich datasets "OSUleaf" and "Fish" are shown in Fig. 5.

6.1.3. Simulated sequence-pair alignment

We simulate aligned sequence pairs by scaling and stretching original time series. Then we run shapeDTW and DTW (and its variants) to align the simulated pairs, and compare their alignment paths against the ground-truth. In this section, shapeDTW is run under the fixed settings: (1) fix the subsequence length to be 30, (2) use *Derivative* as the shape descriptor and (3) use "symmetric1" as the step-pattern.

Aligned-pairs simulation algorithm: given a time series $\mathcal{T}$ of length L, we simulate a new time series by locally scaling and stretching $\mathcal{T}$. The simulation consists of two sequential steps: (1) scaling: scale $\mathcal{T}$ point-wisely, resulting in a new time series $\hat{\mathcal{T}} = \mathcal{T} \otimes S$, where S is a positive scale vector with the same length as $\mathcal{T}$, and $\otimes$ is a point-wise multiplication operator; (2) stretching: randomly choose α percent of temporal points from $\hat{\mathcal{T}}$, stretch each point by a random length τ and result in a new time series $\mathcal{T}'$. $\mathcal{T}'$ and $\mathcal{T}$ are a simulated alignment pair, with the ground-truth alignment known from the simulation process. The simulation algorithm is described in Algorithm 2.

One caveat we have to pay attention to is that: scaling an input time series by a random scale vector can make the resulting time series perceptually quite different from the original one, such that simulated alignment pairs make little sense. Therefore, in practice, a scale vector S should be smooth, i.e., adjacent elements in S cannot be random, instead, they should be similar in magnitude, making adjacent temporal points from the original time series be scaled by a similar amount. In experiments, we first use a random process, which is similar to Brownian motion, to initialize scale vectors, and then recursively smooth it. The scale vector generation algorithm is shown in Algorithm 2. As seen, adjacent scales are initialized to be differed by at most 1 (i.e., $s(t+1) = s(t) + sin(\pi \times randn)$), such that the first order derivatives are

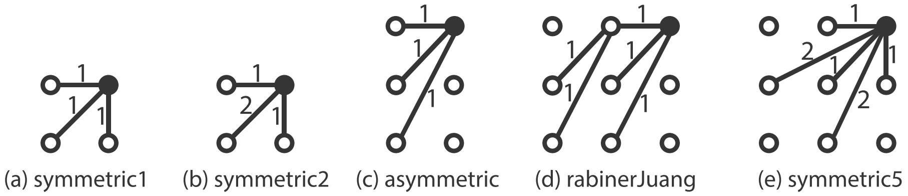


Fig. 4. Five step patterns. Numbers on transitions indicate the multiplicative weight for the local distance $d(i, j)$. Step-pattern (a) "symmetric1" is the default step pattern for DTW and (b) gives more penalties to the diagonal directions, such that the warping favors stair-stepping paths. Step patterns (a) and (b) obtain a continuous warping path, while step patterns (c), (d) and (e) may result in skipping elements, i.e., some temporal points from one sequence are not matched to any points from the other sequence, and vice versa.

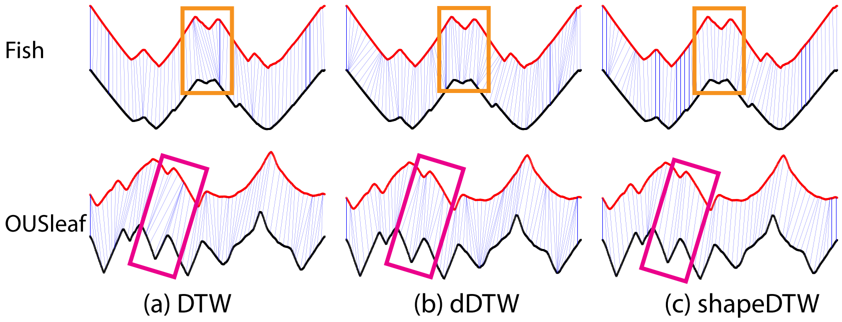


Fig. 5. Alignments between time series with rich local features. Time series at the top and bottom row are from "Fish"(train-165, test-1) and "OSUleaf"(test-114, test-134) datasets respectively. In each pair of time series, temporal points with similar local structures are boxed out by rectangles. Perceptually, shapeDTW aligns these corresponding points better than both DTW and dDTW.

Algorithm 2 Simulate alignment pairs.

```
Simulate an alignment pair:
Inputs: a time series instance $\mathcal{T}$; scale vectorrange $[a\,b]$, smoothing iterations $\Gamma$; stretching percentage$\alpha$, stretching amount $\tau$
1. simulate a scale vector $S$;
2. scale $\mathcal{T}$ point-wisely, $\hat{\mathcal{T}} \leftarrow \mathcal{T} \otimes S$;
3. stretching $\alpha$ percent of points from $\hat{\mathcal{T}}$ by a random amount $\tau$, resulting in a simulated time series $\mathcal{T}'$.
Outputs: $\mathcal{T}'$

Simulate a scale vector:
Inputs: length $L$, iteration $\Gamma$, range $[a\,b]$
1. Initialize:
    $\begin{cases} s(1) = randn \\ s(t+1) = s(t) + sin(\pi \times randn), t \in \{1, 2, \ldots, L-1\} \end{cases}$
2. smoothing:
while iteration < $\Gamma$
    a. set the cumulative sum up to $t$ as the scale at $t$:
        $\begin{cases} s(1) \leftarrow s(1) \\ s(t+1) \leftarrow s(t+1) + s(t), t \in \{1, 2, \ldots, L-1\} \end{cases}$
    b. squash scale at $t$ into the range $[-1, 1]$:
        $s(t) \leftarrow \sin(s(t)),\ t \in \{1, 2, \ldots, L\}$
end
3. squash elements in the scale vector $S$ into range $[a\,b]$by linear scaling.
Outputs: a scale vector $S = \{s(1), s(2), \ldots, s(L)\}$
```

bounded and initialized scale vectors do not change abruptly. Initialized scale vectors usually have local bumps, and we further recursively utilize cumulative summation and sine-squashing, as described in the algorithm, to smooth the scale vectors. Finally, the smoothed scale vectors are linearly squashed into a positive range $[a\,b]$.

After non-uniformly scaling an input time series by a scale vector, we obtain a scale-transformed new sequence, and then we randomly pick α percent of points from the new sequence and stretch each of them by some random amount τ. Stretching at point p by some amount τ is to duplicate p by τ times.

Aligned-pairs simulation: using training data from each UCR dataset as the original time series, we simulate their alignment pairs by running Algorithm 2. Since there are 27,136 training time series instances from 84 UCR datasets, we simulate 27,136 aligned-pairs in total. We fix most simulation parameters as follows: $[a\,b] = [0.5\ 1]$, $\Gamma = 5$, $\tau = \{1, 2, 3\}$, and the stretching percentage α is the only flexible parameter we will vary, e.g., when $\alpha = 15\%$, each original input time series is on average stretched by 30% (in length). Typical scale vectors and simulated alignment pairs are shown in Fig. 6. The scale vectors are smooth and the simulated time series are both scaled and stretched, compared with the original ones.

Alignment comparison between shapeDTW and DTWs: we align simulated time series pairs by running shapeDTW and DTWs (including DTW/dDTW/wDTW), and compare alignment paths against the ground-truth in terms of "*Mean Absolute Deviation*" scores. DTW and dDTW are parameter-free, but wDTW has one tuning parameter g (see Eq. (3) in their paper), which controls the curvature of the logistic weight function. However in the case of aligning two sequences, g is impossible to be tuned and should be pre-defined by experiences. Here we fix g to be 0.1, which is the approximate mean value of the optimal g in the original paper. For the purpose of comparing the alignment qualities of different algorithms, we use the default step pattern, (a) in Fig. 4, for both shapeDTW and DTW/dDTW/wDTW, but we further evaluate effects of different step-patterns in the following experiments.

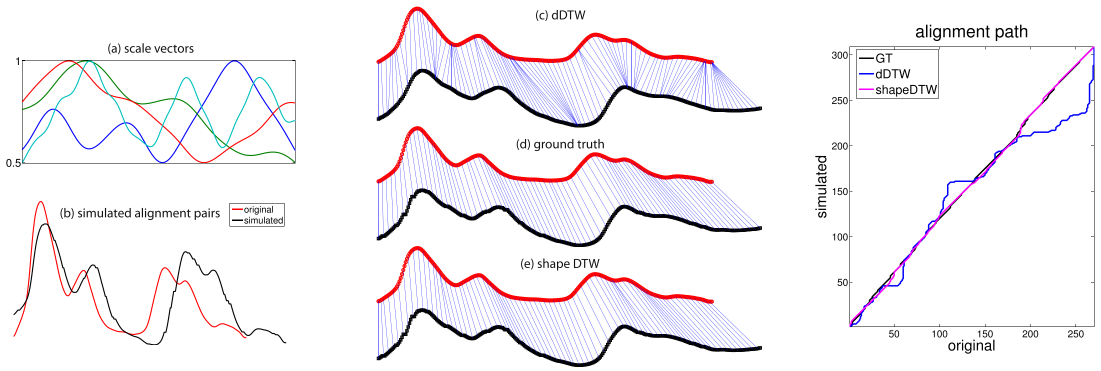


Fig. 6. Alignments between simulated time series pairs. (a) simulated scale vectors: they are smooth and squashed to the range [0.5 1.0]; (b) a simulated alignment pair: generated by artificially scale and stretch the original time series; (c) dDTW alignment: run dDTW to align the simulated pair; (d) ground truth alignment; (e) shapeDTW alignment. The plot on the right shows alignment paths of dDTW, shapeDTW and the ground-truth, visually the alignment path of shapeDTW is closer to the ground-truth, and quantitatively, shapeDTW has 1.1 alignment errors in terms of “*Mean Absolute Deviation*” score, compared with 4.7 of dDTW.

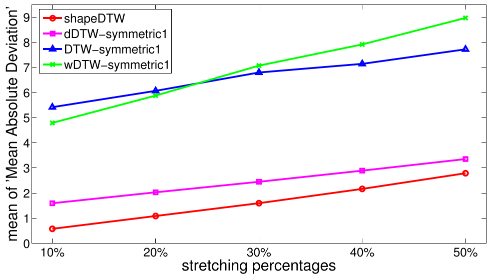


Fig. 7. Alignment quality comparison between shapeDTW and DTW/dDTW/wDTW, under the step pattern “symmetric1”. As seen, as the stretching amount increases, the alignment qualities of both shapeDTW and DTW/dDTW/wDTW drop. However, shapeDTW consistently achieves lower alignment errors under different stretching amounts, compared with DTW, dDTW and wDTW.

We simulate alignment pairs by stretching raw time series by different amounts, 10%, 20%, 30%, 40% and 50%, and report the alignment qualities of shapeDTW and DTW/dDTW/wDTW under each stretching amount in terms of the mean of “*Mean Absolute Deviation*” scores over 27,136 simulated pairs. The results are shown in Fig. 7, which shows shapeDTW achieves lower alignment errors than DTW / dDTW / wDTW over different stretching amounts consistently. shapeDTW almost halves the alignment errors achieved by dDTW, although dDTW already outperforms its two competitors, DTW and wDTW, by a large margin.

Effects of different step patterns: choosing a proper step pattern is a traditionally way to improve sequence alignments, and it usually needs domain knowledge to make the right choice. Here, instead of choosing an optimal step pattern, we run DTW/dDTW/wDTW under all 5 step patterns in Fig. 4 and compare their alignment performances against shapeDTW. Similar as the above experiments, we simulate aligned-pairs under different amounts of stretches, report alignment errors under different step patterns in terms of the mean of “*Mean Absolute Deviation*” scores over 27,136 simulated pairs, and plot the results in Fig. 8. As seen, different step patterns obtain different alignment qualities, and in our case, step patterns, “symmetric1” and “asymmetric”, have similar alignment performances and they reach lower alignment errors than the other 3 step patterns. However, shapeDTW still wins DTW/dDTW/wDTW (under “symmetric1” and “asymmetric” step-patterns) by some margin.

From the above simulation experiments, we observe dDTW (under the step patterns “symmetric1” and “asymmetric”) has the closest performance as shapeDTW. Here we simulate aligned-pairs with on average 30% stretches, run dDTW (under “symmetric1” step pattern) and shapeDTW alignments. shapeDTW has lower “*Mean Absolute Deviation*” scores on 56 datasets, and the mean of “*Mean Absolute Deviation*” on 84 datasets of shapeDTW and dDTW are 1.68/2.75 respectively, indicating shapeDTW achieves much lower alignment errors. This shows a clear superiority of shapeDTW to dDTW for sequence alignment.

The key difference between shapeDTW and DTW/dDTW/wDTW is that whether neighborhood is taken into account when measuring similarities between two points. We demonstrate that taking local neighborhood information into account (shapeDTW) does benefit the alignment.

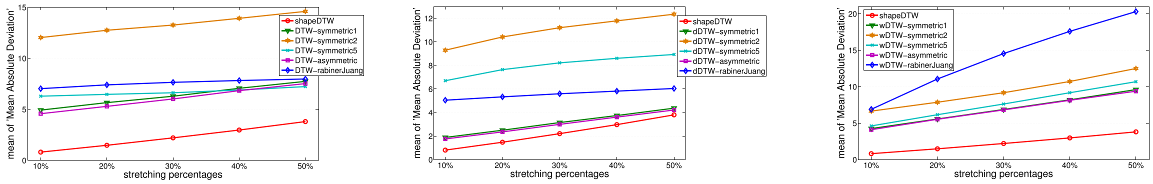
6.1.4. *MIDI-to-audio alignment*

We showed the superiority of shapeDTW to align synthesized alignment pairs, and in this section, we further empirically demonstrate its effectiveness to align audio signals, which have ground-truth alignments.

The Bach10 dataset [36] consists of audio recordings of 10 pieces of Bach’s Chorales, as well as their MIDI scores and the ground-truth alignment between the audio and the MIDI score. MIDI scores are symbolic representations of audio files, and by aligning symbolic MIDI scores with audio recordings, we can do musical information retrieval from MIDI input-data [38]. Many previous work used DTW to align MIDI to audio sequences [26,36,38], and they typically converted MIDI data into audios as a first step, and the problem boils down to audio-to-audio alignment, which is then solved by DTW. We follow this convention to convert MIDI to audio first, but run shapeDTW instead for alignments.

Each piece of music is approximately 30 seconds long, and in experiments, we segment both the audio and the converted audio from MIDI data into frames of 46ms length with a hopsize of 23ms, extract features from each 46ms frame window, and in this way, the audio is represented as a multivariate time series with the length equal to the number of frames and dimension equal to the feature dimensions. There are many potential choices of frame features, but how to select and combine features in an optimal way to improve the alignment is beyond the scope of this paper, we refer the interested readers to Kirchhoff and co-workers [26,35]. Without loss of generality, we use Mel-frequency cepstral coefficients (MFCCs) as features, due to its common usage and good performance in speech recognition and musical information retrieval [39]. In our experiments, we use the first 5 MFCCs coefficients.

After MIDI-to-audio conversion and MFCCs feature extraction, MIDI files and audio recordings are represented as 5-dimensional multivariate time series, with approximately length $L \approx 1300$. A typical audio signal, MIDI-converted audio signal, and their 5D MFCCs features are shown in Fig. 9. We align 5D MFCCs sequences



(a) shapeDTW vs. DTW (different step patterns)

(b) shapeDTW vs. dDTW (different step patterns)

(c) shapeDTW vs. wDTW (different step patterns)

Fig. 8. Align sequences under different step patterns. We align sequence-pairs by DTW/dDTW/wDTW under 5 different step patterns (Fig. 4), "symmetric1", "symmetric2", "symmetric5", "asymmetric" and "rabinerJuang", and compare their alignment errors against those obtained by shapeDTW. As seen, different step patterns usually reach different alignment results, which shows the importance of choosing an appropriate step pattern adapted to the application domain. In our case, "asymmetric" step pattern achieves slightly lower errors than "symmetric1" step pattern (under DTW, wDTW and dDTW), however, shapeDTW consistently wins DTW/dDTW/wDTW under the best step pattern - "asymmetric".

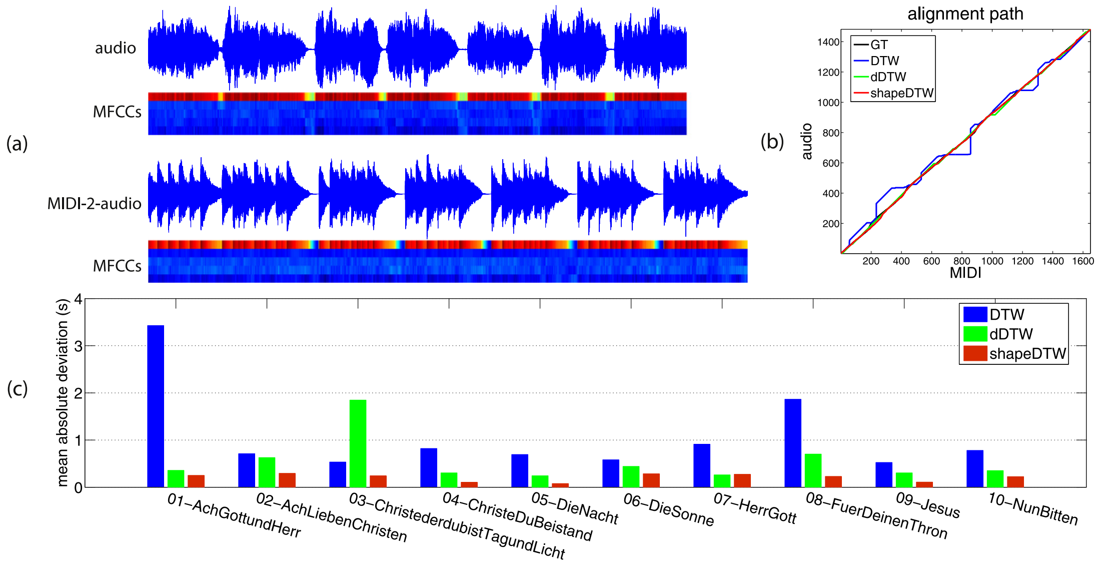


Fig. 9. Align audio and midi-2-audio sequences. (a) top: the audio waveform of the Chorale '05-DieNacht' and its 5D MFCCs features; bottom: the converted audio waveform from the MIDI score of the Chorale '05-DieNacht' and its corresponding 5D MFCCs features; (b) alignment paths: align two MFCCs sequences by DTW, dDTW and shapeDTW, and the plot shows their alignment paths, together with the ground-truth alignment. As seen, the alignment paths of dDTW and shapeDTW are closer to the ground-truth than that of DTW. (c) "*Mean Absolute Deviation*" from the ground truth alignment: on 9 (10) out of 10 chorales, shapeDTW achieves smaller alignment errors than dDTW (DTW), showing that shapeDTW outperforms DTW/dDTW to align real sequence pairs as well.

by shapeDTW: although shapeDTW is designed for univariate time series alignments, it naturally extends to multivariate cases: first extract a subsequence from each temporal point, then encode subsequences by shape descriptors, and in this way, the raw multivariate time series is converted to a descriptor sequence. In the multivariate time series case, each extracted subsequence is multi-dimensional, having the same dimension as the raw time series, and to compute the shape descriptor of a multi-dimensional subsequence, we compute shape descriptors of each dimension independently, concatenate all shape descriptors, and use it as the shape representation of that subsequence.

We compare alignments by shapeDTW against DTW/dDTW, and all of them use the "symmetric1" step pattern. The length of subsequences in shapeDTW is fixed to be 20 (we tried 5,10, 30 as well and achieved quite similar results), and *Derivative* is used as the shape descriptor. The alignment qualities in terms of "*Mean Absolute Deviation*" on 10 Chorales are plotted in Fig. 9. To be consistent with the convention in the audio community, we actually report the mean-delayed-second between the alignment paths and the ground-truth. The mean-delayed-second is computed as: dividing "*Mean Absolute Deviation*" by the sampling rate of the audio signal. shapeDTW outperforms dDTW/DTW on 9/10 MIDI-to-audio alignments. This shows taking local neighborhood information into account does benefit the alignment.

6.2. Time series classification

We compare NN-shapeDTW with NN-DTW on 84 UCR time series datasets for classification. Since these datasets have standard partitions of training and test data, we experiment with these given partitions and report classification accuracies on the test data.

In the above section, we explore the influence of different steps patterns, but here both DTW and shapeDTW use the widely adopted step pattern "symmetric1" (Fig. 4 (a)) under no temporal window constraints to align sequences.

NN-DTW: each test time series is compared against the training set, and the label of the training time series with the minimal DTW distance to that test time series determines the predicted label. All training and testing time series are z-normalized in advance.

shapeDTW: we test all 5 shape descriptors. We z-normalize time series in advance, sample subsequences from the time series, and compute 3 magnitude-aware shape descriptors, *Raw-Subsequence*, *PAA* and *DWT*, and 2 y-shift invariant shape descriptors, *Slope* and *HOG1D*. Parameter setting for 5 shape descriptors: (1) The length of subsequences to be sampled around temporal points is fixed to 30, as a result *Raw-Subsequence* descriptor is a 30D vector; (2) *PAA* and *Slope* uses 5 equal-lengthed intervals, therefore they have the dimensionality 5; (3) As mentioned,

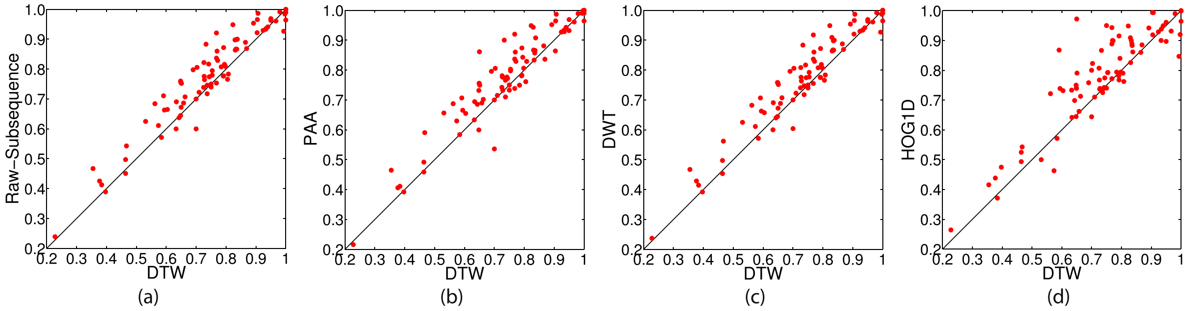


Fig. 10. Classification accuracy comparisons between NN-DTW and NN-shapeDTW on 84 UCR time series datasets. shapeDTW under 4 shape descriptors, *Raw-Subsequence*, *PAA*, *DWT* and *HOG1D*, outperforms DTW on 64/63/64/61 datasets respectively, and Wilcoxon signed rank test shows shapeDTW under all descriptors performs significantly better than DTW. *Raw-Subsequence* (*PAA* and *DWT* as well) outperforms DTW on more datasets than *HOG1D* does, but *HOG1D* achieves large accuracy improvements on more datasets, concretely, *HOG1D* boosts accuracies by more than 10% on 18 datasets, compared with on 12 datasets by *Raw-Subsequence*.

HOG1D uses 8 bins and 2 non-overlapping intervals, and the scale factor σ is fixed to be 0.1. At last HOG1D is a 16D vector representation.

NN-shapeDTW: first transform each training/testing time series to a shape descriptor sequence, and in this way, original univariate time series are converted into multivariate descriptor time series. Then apply NN-DTW on the multivariate time series to predict labels.

NN-shapeDTW vs. NN-DTW: we compare NN-shapeDTW, under 4 shape descriptors *Raw-Subsequence*, *PAA*, *DWT* and *HOG1D*, with NN-DTW, and plot their classification accuracies on 84 datasets in Fig. 10. shapeDTW outperforms (including ties) DTW on 64/63/64/61 (*Raw-Subsequence*/*PAA*/*DWT*/*HOG1D*) datasets, and by running the Wilcoxon signed rank test between performances of NN-shapeDTW and NN-DTW, we obtain p-values $5.5 \cdot 10^{-8}/5.1 \cdot 10^{-7}/4.8 \cdot 10^{-8}/1.7 \cdot 10^{-6}$, showing that shapeDTW under all 4 descriptors performs significantly better than DTW. Compared with DTW, shapeDTW has a preceding shape descriptor extraction process, and approximately takes time $\mathcal{O}(l \cdot L)$, where l and L is the length of subsequence and time series respectively. The second step of shapeDTW is a typical DTW, and runs in $\mathcal{O}(l \cdot L^2)$. The total time complexity of shapeDTW is $\mathcal{O}(l \cdot L^2)$, while running DTW to align 1D time series takes time $\mathcal{O}(L^2)$. By trading off a slight amount of time and space, shapeDTW brings large accuracy gains.

Since *PAA* and *DWT* are approximations of *Raw-Subsequence*, and they have similar performances as *Raw-Subsequence* under the nearest classifier, we choose *Raw-Subsequence* as a representative for following analysis. *Raw-Subsequence* loses on 20 datasets, on 18 of which it has minor losses ($<4\%$), and on the other 2 datasets, “Computers” and “Synthetic-control”, it loses by 10% and 6.6%. Time series instances from these 2 datasets either have high-frequency spikes or have many abrupt direction changes, making them resemble noisy signals. Possibly, comparing the similarity of two points using their noisy neighborhoods is not as good as using their single coordinate values (DTW), since temporal neighborhood may accumulate and magnify noise.

HOG1D loses on 23 datasets, on 18 of which it has minor losses ($<5\%$), and on the other 5 datasets, “CBF”, “Computers”, “ItalyPowerDemand”, “Synthetic-control” and “Wine”, it loses by 7.7%, 5.6%, 5.3%, 14% and 11%. By visually inspecting, time series from “Computers”, “CBF” and “Synthetic-control” are spiky and bumpy, making them highly non-smooth. This makes the first-order-derivative based descriptor *HOG1D* inappropriate to represent local structures. Time series instances from ’ItalyPowerDemand’ have length 24, while we sample subsequences of length 30 from each point, this makes *HOG1D* descriptors from different local points almost the same, such that *HOG1D* becomes not discriminative of local structures. This makes shapeDTW inferior to DTW. Although *HOG1D* loses on more datasets than *Raw-Subsequence*, *HOG1D* boosts accuracies by more than 10% on 18 datasets, compared with on 12 datasets by *Raw-Subsequence*. On datasets “OSUleaf” and “BirdChicken”, the accuracy gain is as high as 27% and 20%. By checking these two datasets closely, we find different classes have membership-discriminative local patterns (a.k.a shapelets [5]), however, these patterns differ only slightly among classes. *Raw-Subsequence* shape descriptor can not capture these minor differences well, while *HOG1D* is more sensitive to shape variations since it calculates derivatives.

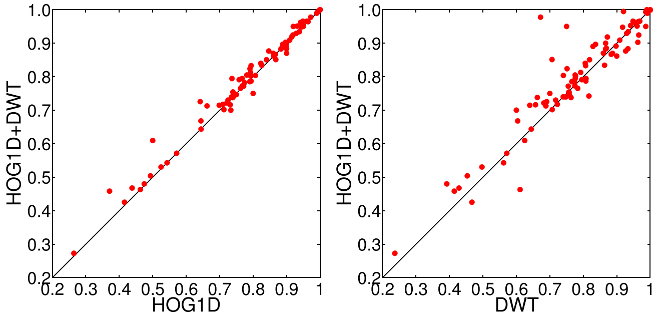


Fig. 11. Performance comparisons between the fused descriptor *HOG1D+DWT* and individual ones *HOG1D*/*DWT*. *HOG1D+DWT* outperforms *HOG1D*/*DWT* on 66/51 (out of 84) datasets, and statistical hypothesis tests show the improvements are significant.

Both *Raw-Subsequence* and *HOG1D* bring significant accuracy gains, however, they boost accuracies to different extents on the same dataset. This indicates the importance of designing domain-specific shape descriptors. Nevertheless, we show that even by using simple and dataset-independent shape descriptors, we still obtain significant improvements over DTW. Classification error rates of DTW, *Raw-Subsequence* and *HOG1D* on 84 datasets are documented in Table 1.

Superiority of Compound shape descriptors: as mentioned in Section 4, a compound shape descriptor obtained by fusing two complementary descriptors may inherit benefits from both descriptors, and becomes even more discriminative of subsequences. As an example, we concatenate a y-shift invariance descriptor *HOG1D* and a magnitude-aware descriptor *DWT* using equal weights, resulting in a compound descriptor $HOG1D + DWT = (HOG1D, DWT)$. Then we evaluate classification performances of 3 descriptors under the nearest neighbor classifier, and plot the comparisons in Fig. 11. *HOG1D+DWT* outperforms (including ties) *HOG1D* / *DWT* on 66/51 (out of 84) datasets, and by running the Wilcoxon signed rank hypothesis test between performances of *HOG1D+DWT* and *HOG1D* (*DWT*), we get p-values $5.5 \cdot 10^{-5}/0.0034$,

Table 1
Error rates of NN-DTW and NN-shapeDTW (under descriptors *Raw-Subsequence* and *HOG1D*) on 84 UCR datasets. Underscored rates are those that shapeDTW has improved the accuracies by over 10%.

Classification error rates on 84 UCR datasets							
datasets	DTW	Raw-Subsequence	HOG1D	datasets	DTW	Raw-Subsequence	HOG1D
50words	0.310	**0.202**	**0.242**	MedicalImages	0.263	**0.254**	0.264
Adiac	0.396	**0.335**	**0.269**	MiddlePhalanxOutlineAgeGroup	**0.250**	0.260	0.260
ArrowHead	0.297	**0.194**	**0.177**	MiddlePhalanxOutlineCorrect	0.352	**0.240**	**0.250**
Beef	0.367	0.400	**0.267**	MiddlePhalanxTW	**0.416**	0.429	0.429
BeetleFly	0.300	**0.300**	**0.200**	MoteStrain	0.165	**0.101**	**0.110**
BirdChicken	0.250	**0.250**	**0.050**	NonInvasiveFatalECG-Thorax1	**0.209**	0.223	0.219
Car	0.267	**0.117**	**0.133**	NonInvasiveFatalECG-Thorax2	0.135	**0.110**	0.140
CBF	**0.003**	0.016	0.080	OliveOil	0.167	**0.133**	**0.100**
ChlorineConcentration	**0.352**	0.355	0.355	OSULeaf	0.409	**0.289**	**0.132**
CinC-ECG-torso	0.349	**0.248**	**0.209**	PhalangesOutlinesCorrect	0.272	**0.235**	**0.261**
Coffee	**0.000**	0.036	0.036	Phoneme	0.772	**0.761**	**0.736**
Computers	**0.300**	0.400	0.356	Plane	0.000	**0.000**	**0.000**
Cricket-X	0.246	**0.221**	**0.208**	ProximalPhalanxOutlineAgeGroup	**0.195**	0.234	0.210
Cricket-Y	0.256	**0.226**	**0.226**	ProximalPhalanxOutlineCorrect	0.216	**0.192**	**0.206**
Cricket-Z	0.246	**0.205**	**0.208**	ProximalPhalanxTW	**0.263**	0.282	0.275
DiatomSizeReduction	**0.033**	0.039	0.069	RefrigerationDevices	0.536	0.549	**0.507**
DistalPhalanxOutlineAgeGroup	**0.208**	0.223	0.233	ScreenType	0.603	0.611	**0.525**
DistalPhalanxOutlineCorrect	0.232	0.247	**0.228**	ShapeletSim	0.350	**0.328**	**0.028**
DistalPhalanxTW	0.290	**0.277**	**0.290**	ShapesAll	0.232	**0.163**	**0.112**
Earthquakes	0.258	**0.183**	**0.258**	SmallKitchenAppliances	0.357	0.363	**0.301**
ECG200	0.230	**0.140**	**0.100**	SonyAIBORobotSurface	0.275	**0.261**	**0.193**
ECG5000	0.076	**0.070**	**0.071**	SonyAIBORobotSurfaceII	0.169	**0.136**	0.174
ECGFiveDays	0.232	**0.079**	**0.057**	Strawberry	0.060	**0.059**	**0.051**
FaceAll	**0.192**	0.217	0.238	SwedishLeaf	0.208	**0.128**	**0.085**
FaceFour	0.170	**0.102**	**0.091**	Symbols	0.050	**0.031**	**0.039**
FacesUCR	0.095	**0.034**	**0.081**	synthetic-control	**0.007**	0.073	0.153
FISH	0.177	**0.051**	**0.051**	ToeSegmentation1	0.228	**0.171**	**0.101**
FordA	0.438	**0.316**	**0.279**	ToeSegmentation2	0.162	**0.100**	**0.138**
FordB	0.406	**0.337**	**0.261**	Trace	0.000	0.010	**0.000**
Gun-Point	0.093	**0.013**	**0.007**	TwoLeadECG	0.096	**0.078**	**0.006**
Ham	0.533	**0.457**	**0.457**	Two-Patterns	0.000	**0.000**	0.001
HandOutlines	0.202	**0.191**	0.206	UWaveGestureLibraryAll	0.108	**0.046**	**0.058**
Haptics	0.623	**0.575**	**0.562**	uWaveGestureLibrary-X	0.273	**0.224**	**0.263**
Herring	0.469	**0.375**	0.500	uWaveGestureLibrary-Y	0.366	**0.309**	**0.358**
InlineSkate	0.616	**0.587**	0.629	uWaveGestureLibrary-Z	0.342	**0.314**	**0.338**
InsectWingbeatSound	0.645	**0.533**	**0.584**	wafer	0.020	**0.008**	**0.010**
ItalyPowerDemand	0.050	**0.037**	0.103	Wine	0.426	**0.389**	0.537
LargeKitchenAppliances	0.205	**0.184**	**0.160**	WordsSynonyms	0.351	**0.245**	**0.260**
Lighting2	0.131	0.131	**0.115**	WordSynonyms	0.351	**0.245**	**0.260**
Lighting7	0.274	**0.178**	**0.233**	Worms	0.536	**0.503**	**0.475**
MALLAT	0.066	**0.064**	**0.062**	WormsTwoClass	0.337	**0.293**	**0.287**
Meat	0.067	**0.067**	0.100	yoga	0.164	**0.133**	**0.117**

showing the compound descriptor outperforms individual descriptors significantly. We can generate compound descriptors by weighted concatenation, with weights tuned by cross-validation on training data, but this is beyond the scope of this paper.

Texas Sharpshooter plot: although NN-shapeDTW performs better than NN-DTW, knowing this is not useful unless we can tell in advance on which problems it will be more accurate, as stated in [14]. Here we use the Texas sharpshooter plot [14] to show when NN-shapeDTW has superior performance on the test set as predicted from performance on the training set, compared with NN-DTW. We run leave-one-out cross validation on training data to measure the accuracies of NN-shapeDTW and NN-DTW, and we calculate the expected gain: accuracy(NN-shapeDTW)/accuracy(NN-DTW). We then measure the actual accuracy gain using the test data. The Texas Sharpshooter plots between *Raw-Subsequence/HOG1D* and DTW on 84 datasets are shown in Fig. 12. 87%/86% points (*Raw-Subsequence/HOG1D*) fall in the TP and TN regions, which means we can confidently predict that our algorithm will be superior/inferior to NNDTW. There are respectively 7/7 points falling inside the FP region for descriptors *Raw-Subsequence/HOG1D*, but they just represent minor losses, i.e., actual accuracy gains lie within [0.9 1.0].

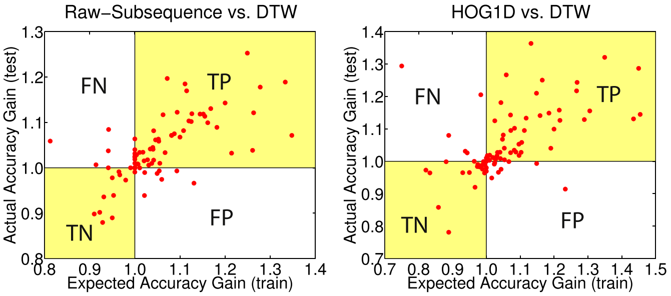


Fig. 12. Texas sharpshoot plot between *Raw-Subsequence/HOG1D* and DTW on 84 datasets. TP: true positive (our algorithm was expected from the training data to outperform NNDTW, and it actually did on the test data). TN: true negatives, FP: false positives, FN: false negatives. There are 87%/86% points (*Raw-Subsequence/HOG1D* vs. DTW) falling in the TP and TN regions, which indicates we can confidently predict that our algorithm will be superior/inferior to NNDTW.

6.3. Sensitivity to the size of neighborhood

In the above experiments, we showed that shapeDTW outperforms DTW both qualitatively and quantitatively. But we are still left with one free-parameter: the size of neighborhood, i.e., the

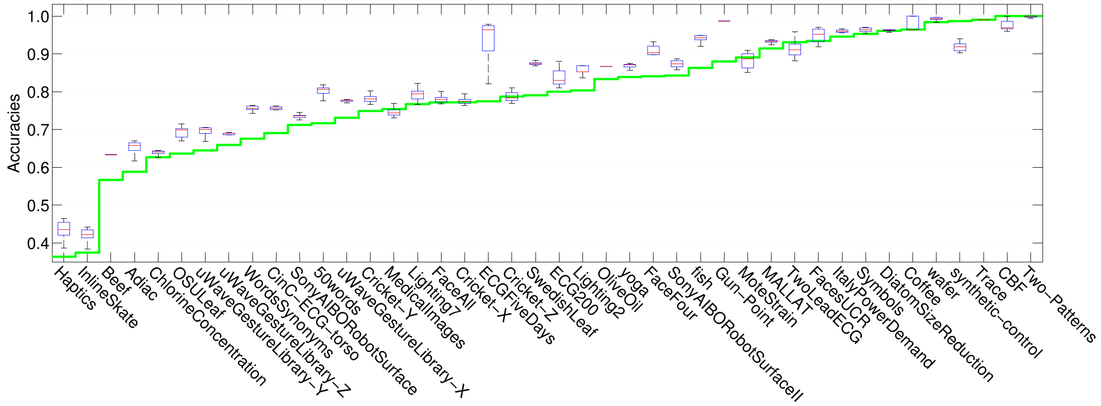


Fig. 13. Performances of shapeDTW are insensitive to the neighborhood size. The green stairstep curve shows dataset-wise test accuracies of NN-DTW, and the box plot shows performances of NN-shapeDTW under the shape descriptor *Raw-Subsequence*. On each dataset, we plot a blue box with two tails: the lower and upper edge of each blue box represent 25th and 75th percentiles of 20 test accuracies (obtained under different neighborhood sizes, i.e., 5, 10, 15, ... , 100) on that dataset, with the red line inside the box marking the median accuracy and two tails indicating the best and worst test accuracies. On 36 out of 42 datasets, the median accuracies of NN-shapeDTW are larger than accuracies obtained by NN-DTW, and on 33 datasets, even the worst performances by NN-shapeDTW are better than NN-DTW. All these statistics show shapeDTW works well under wide ranges of neighborhood sizes. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

length of the subsequence to be sampled from each point. Let t_i be some temporal point on the time series $\mathcal{T} \in \mathcal{R}^L$, and s_i be the subsequence sampled at t_i. When $|s_i| = 1$, shapeDTW (under the *Raw-Subsequence* shape descriptor) degenerates to DTW; when $|s_i| = L$, subsequences sampled at different points become almost identical, make points un-identifiable by their shape descriptors. This shows the importance to set an appropriate subsequence length. However, without dataset-specific domain knowledge, it is hard to determine the length intelligently. Here instead, we explore the sensitivity of the classification accuracies to different subsequence lengths. We conduct experiments on 42 old UCR datasets.

We use *Raw-Subsequence* as the shape descriptor, and NN-shapeDTW as the classifier. We let the length of subsequences to vary from 5 to 100, with stride 5, i.e., we repeat classification experiments on each dataset for 20 times, and each time set the length of subsequences to be $5 \times i$, where i is the index of experiments ($1 \le i \le 20$, $i \in \mathcal{Z}$). The test accuracies under 20 experiments are shown by a box plot (Fig. 13). On 33 out of 42 datasets, even the worst performances of NN-shapeDTW are better than DTW, indicating shapeDTW performs well under wide ranges of neighborhood sizes.

6.4. Comparison with the state-of-the-art time series classification algorithms

As shown in [10–12,37,40], 1NN classifier with the ordinary DTW distance as the similarity measure (1NN-DTW) is very hard to beat. As further shown in [4,40], 1NN-DTW with a warping window set through cross validation works even better than 1NN-DTW without warping window constraints. In this section, we set 1NN-DTW (with the optimal warping window set through cross-validation) as the baseline, and evaluate other time series classification algorithms. For consistency and reproducibility, we directly use 1NN-DTW (with the optimal warping window tuned) classification accuracies published on the UCR time series website.

For our algorithm, 1NN-shapeDTW, we run under the following settings: (1) set the subsequence length to 30; (2) use *Raw-Subsequence* as the shape descriptor; (3) run DTW without the warping window constraints. 1NN-shapeDTW wins/draws the baseline on 62 datasets (out of 84), and the signed rank Wilcoxon test returns p-values 0.0012, showing significant accuracy improvement over 1NN-DTW (with the optimal warping window tuned).

We further compare our algorithm to 19 state-of-the-art algorithms surveyed in [40] and one additional algorithm in [23]. In [40], every algorithm is compared with the baseline by computing two scores: the percentage of the datasets it wins over 1NN-DTW and the mean of the classification accuracy differences (see Table 4 in [40]). Our algorithm beats 1NN-DTW on 73.81% datasets, with the mean accuracy difference 2.55%. The algorithm in [23], BOW-local, wins 1NN-DTW on 71.43% datasets, with the mean difference 3.72%. From Table 4 in Bagnall et al. [40], we thus outperform 14 of the 20 algorithms (19 algorithms in [40] and one algorithm in [23]). We don't outperform all 20, but we believe we still have merits: our algorithm is a simple NN classifier with no parameter to tune, while other algorithms improve accuracy using complicated data preprocessing (e.g., Fourier transform, symbolization, feature crafting) and complicated classifiers (e.g., random forest, neural network, ensemble method). To be specific, the top 4 methods that perform better than ours, COTE [41], EE [42], BOSS [43] and ST [44], are all ensemble classifiers: e.g., COTE involves pooling 35 classifiers into a single ensemble. Our method is a single classifier, and on some datasets it does not perform as well as these 4 ensemble classifiers; however, compared with ensembles, our method is more interpretable and is more time efficient. For the other two winning methods [23,45], both of them consists of two sequential steps: time series representation and SVM classification. First of all, both of them use the advanced classifier - SVM, which involves hyperparameter tuning, while ours uses the 1NN classifier, with no hyperparameter to tune. Second, both algorithms have to preprocess the time series and represent it by a feature vector, in which process they have to manually design and craft features. However, our algorithm 1NN-shapeDTW under the *Raw-Subsequence* descriptor significantly outperforms the baseline but with no feature-crafting.

6.5. More comparison between shapeDTW and DTW variants

In Section 6.2, we showed 1NN-shapeDTW outperforms 1NN-DTW significantly. Here we further compare our algorithm against 1NN-dDTW and 1NN-wDTW for time series classification. We again run 1NN-shapeDTW by setting the subsequence length to 30 and using *Raw-Subsequence* as the shape descriptor for the subsequence. 1NN-shapeDTW wins/draws 1NN-dDTW/1NN-wDTW on 52/56 (out of 84) datasets, the mean accuracy (and its standard deviation) for 1NN-shapeDTW/1NN-dDTW/1NN-wDTW is 0.7771(0.16)/0.7447(0.17)/0.7402(0.17), and the Wilcoxon signed rank test between 1NN-shapeDTW and 1NN-dDTW/1NN-wDTW returns p-values 0.03/0.007, showing our algorithm outperforms both 1NN-dDTW and 1NN-wDTW significantly.

There are two consecutive steps to compute sequence-to-sequence distance: find the alignment and then compute the cumulative point-to-point distance along the alignment path. To compare 1NN-shapeDTW with 1NN-DTW/1NN-wDTW/1NN-dDTW in a more controlled way, we only use shapeDTW and DTW variants to search the alignment path, and then we use the same method to calculate the cumulative point-to-point distance along the alignment path. To be concrete, we compute the point-to-point distance along the alignment path as in shapeDTW, i.e. compute point-to-point distance by the Euclidean distance between their *Raw-Subsequence* descriptors. The mean accuracy (and its standard deviation) for 1NN-shapeDTW, 1NN-DTW, 1NN-dDTW and 1NN-wDTW are 0.7771(0.16), 0.7540(0.17), 0.7681(0.17) and 0.7584(0.16); 1NN-shapeDTW wins/draws 1NN-DTW/1NN-dDTW/1NN-wDTW on 59/48/59 (out of 84) datasets, and the signed rank test between 1NN-shapeDTW and 1NN-DTW/1NN-dDTW/1NN-wDTW returns p-values 0.002/0.694/0.007. All these shows: (1) 1NN-shapeDTW is only different from 1NN-DTW/1NN-dDTW/1NN-wDTW in the first alignment step, but still outperforms 1NN-DTW/1NN-wDTW significantly; although 1NN-shapeDTW does not win 1NN-dDTW significantly, it does boost the mean accuracy by ∼0.9%; all these indirectly demonstrate the superior alignment quality of shapeDTW; (2) just by replacing the point-to-point distance calculation, we improve the performance of 1NN-DTW/1NN-dDTW/1NN-wDTW by ∼2%, showing that point-to-point similarity is better measured by their surrounding shapelets.

7. Conclusion

We have proposed an new temporal sequence alignment algorithm, shapeDTW, which achieves quantitatively better alignments than DTW and its variants. shapeDTW is a quite generic framework as well, and uses can design their own local subsequence descriptor and fit it into shapeDTW. We experimentally showed that shapeDTW under the nearest neighbor classifier obtains significantly improved classification accuracies than NN-DTW. Therefore, NN-shapeDTW sets a new accuracy baseline for further comparison.

Acknowledgments

This work was supported by the National Science Foundation (grant number CCF-1317433), the Office of Naval Research (N00014-13-1-0563) and the Army Research Office (W911NF-11-1-0046 and W911NF-12-1-0433). The authors affirm that the views expressed herein are solely their own, and do not represent the views of the United States government or any agency thereof.

References

[1] L. Rabiner, B. Juang, Fundamentals of Speech Recognition, 14, PTR Prentice Hall Englewood Cliffs, 1993.
[2] E. Hsu, K. Pulli, J. Popović, Style translation for human motion, ACM Trans. Graph. 24 (3) (2005) 1082–1089.
[3] K. Kulkarni, G. Evangelidis, J. Cech, R. Horaud, Continuous action recognition based on sequence alignment, IJCV (2014) 1–25.
[4] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, G. Batista, The ucr time series classification archive, 2015, www.cs.ucr.edu/~eamonn/time_series_data/.
[5] L. Ye, E. Keogh, Time series shapelets: a new primitive for data mining, in: SIGKDD, ACM, 2009, pp. 947–956.
[6] S. Agarwal, D. Roth, Learning a sparse representation for object detection, in: ECCV, Springer, 2002, pp. 113–127.
[7] D. Lowe, Distinctive image features from scale-invariant keypoints, IJCV 60 (2) (2004) 91–110.
[8] E. Keogh, M. Pazzani, Derivative dynamic time warping., in: SDM, 1, SIAM, 2001, pp. 5–7.
[9] Y. Jeong, M. Jeong, O. Omitaomu, Weighted dynamic time warping for time series classification, Pattern Recognit. 44 (9) (2011) 2231–2240.
[10] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, E. Keogh, Experimental comparison of representation methods and distance measures for time series data, DMKD 26 (2) (2013) 275–309.
[11] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, E. Keogh, Searching and mining trillions of time series subsequences under dynamic time warping, in: SIGKDD, ACM, 2012, pp. 262–270.
[12] F. Petitjean, G. Forestier, G. Webb, A. Nicholson, Y. Chen, E. Keogh, Dynamic time warping averaging of time series allows faster and more accurate classification, ICDM, 2014.
[13] T. Giorgino, Computing and visualizing dynamic time warping alignments in r: the dtw package, J. Stat. Softw. 31 (7) (2009) 1–24.
[14] G. Batista, X. Wang, E. Keogh, A complexity-invariant distance measure for time series., in: SDM, 11, 2011, pp. 699–710.
[15] R. Lajugie, D. Garreau, F. Bach, S. Arlot, Metric learning for temporal sequence alignment, in: NIPS, 2014, pp. 1817–1825.
[16] S. Candan, R. Rossini, X. Wang, M. Sapino, Sdtw: computing dtw distances using locally relevant constraints based on salient feature alignments, Proc. VLDB Endowment 5 (11) (2012) 1519–1530.
[17] J. Lin, R. Khade, Y. Li, Rotation-invariant similarity in time series using bag-of–patterns representation, J. Intell. Inf. Syst. 39 (2) (2012) 287–315.
[18] J. Lin, E. Keogh, S. Lonardi, B. Chiu, A symbolic representation of time series, with implications for streaming algorithms, in: SIGMOD Workshop, ACM, 2003, pp. 2–11.
[19] T. Rakthanmanon, E. Keogh, Fast shapelets: a scalable algorithm for discovering time series shapelets, SDM, 2013.
[20] D. Silva, V. De Souza, G. Batista, et al., Time series classification using compression distance of recurrence plots, in: ICDM, IEEE, 2013, pp. 687–696.
[21] M. Baydogan, G. Runger, E. Tuv, A bag-of-features framework to classify time series, PAMI 35 (11) (2013) 2796–2802.
[22] J. Grabocka, L. Schmidt-Thieme, Invariant time-series factorization, DMKD 28 (5–6) (2014) 1455–1479.
[23] J. Zhao, L. Itti, Classifying time series using local descriptors with hybrid sampling, TKDE 28 (3) (2016) 623–637.
[24] H. Sakoe, S. Chiba, Dynamic programming algorithm optimization for spoken word recognition, IEEE Trans. Acoust. Speech Signal Process. 26 (1) (1978) 43–49.
[25] E. Keogh, C. Ratanamahatana, Exact indexing of dynamic time warping, Knowl. Inf. Syst. 7 (3) (2005) 358–386.
[26] D. Garreau, R. Lajugie, S. Arlot, F. Bach, Metric learning for temporal sequence alignment, in: NIPS, 2014, pp. 1817–1825.
[27] D. Ellis, Dynamic time warp (dtw) in matlab, 2003, www.ee.columbia.edu/~dpwe/resources/matlab/dtw/.
[28] D. Lemire, Faster retrieval with a two-pass dynamic-time-warping lower bound, Pattern Recognit. 42 (9) (2009) 2169–2180.
[29] E. Keogh, K. Chakrabarti, M. Pazzani, S. Mehrotra, Dimensionality reduction for fast similarity search in large time series databases, Knowl. Inf. Syst. 3 (3) (2001) 263–286.
[30] B. Yi, C. Faloutsos, Fast time sequence indexing for arbitrary lp norms, VLDB, 2000.
[31] H. Deng, G. Runger, E. Tuv, M. Vladimir, A time series forest for classification and feature extraction, Inf. Sci. 239 (2013) 142–153.
[32] D. Forsyth, J. Ponce, Computer vision: a modern approach(2003).
[33] J. Zhao, L. Itti, Decomposing time series with application to temporal segmentation, in: Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on, IEEE, 2016, pp. 1–9.
[34] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: CVPR, 1, IEEE, 2005, pp. 886–893.
[35] H. Kirchhoff, A. Lerch, Evaluation of features for audio-to-audio alignment, J. New Music Res. 40 (1) (2011) 27–41.
[36] Z. Duan, B. Pardo, Soundprism: an online system for score-informed source separation of music audio, IEEE J. Sel. Top. Signal Process. 5 (6) (2011) 1205–1215.
[37] A. Bagnall, J. Lines, An experimental evaluation of nearest neighbour time series classification, 2014 arXiv:1406.4757.
[38] N. Hu, R. Dannenberg, G. Tzanetakis, Polyphonic audio matching and alignment for music retrieval, Comput. Sci. Department (2003) 521.
[39] B. Logan, et al., Mel frequency cepstral coefficients for music modeling., ISMIR, 2000.
[40] A. Bagnall, A. Bostrom, J. Large, J. Lines, The great time series classification bake off: an experimental evaluation of recently proposed algorithms. extended version, 2016 arXiv:1602.01711.
[41] A. Bagnall, J. Lines, J. Hills, A. Bostrom, Time-series classification with cote: the collective of transformation-based ensembles, TKDE 27 (9) (2015) 2522–2535.
[42] J. Lines, A. Bagnall, Time series classification with ensembles of elastic distance measures, DAMI 29 (3) (2015) 565–592.
[43] P. Schäfer, The boss is concerned with time series classification in the presence of noise, DAMI 29 (6) (2015) 1505–1530.
[44] A. Bostrom, A. Bagnall, Binary shapelet transform for multiclass time series classification, in: International Conference on Big Data Analytics and Knowledge Discovery, Springer, 2015, pp. 257–269.
[45] R.J. Kate, Using dynamic time warping distances as features for improved time series classification, DAMI 30 (2) (2016) 283–312.

Jiaping Zhao received his bachelor and master degree from Wuhan University in 2008 and 2010 respectively. Currently he is a Ph.D student at iLab, University of Southern California, working under the supervision of Dr. Laurent Itti. His research interests include computer vision, data mining, visual attention and probabilistic graphical model.

Laurent Itti received the MS degree in image processing from École nationale supérieure des télécommunications in Paris in 1994 and the PhD degree in computation and neural systems from California Institute of Technology in 2000. Now he is a professor of computer science, psychology, and neurosciences at University of Southern California. His research interests include computational neuroscience, neural networks, visual attention, brain modeling. He is a member of the IEEE.