Pattern Recognition 73 (2018) 47–64

Contents lists available at ScienceDirect

Pattern Recognition

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

Comparative study of conventional time series matching techniques for word spotting

Tanmoy Mondal[a,1,*], Nicolas Ragot[a], Jean-yves Ramel[a], Umapada Pal[b]

[a] *Laboratoire d'Informatique, Université François Rabelais, Tours, France*
[b] *Computer Vision and Pattern Recognition Unit, Indian Statistical Institute, Kolkata, India*

ARTICLE INFO

Article history:
Received 7 February 2017
Revised 17 May 2017
Accepted 7 July 2017
Available online 25 July 2017

Keywords:
Word spotting
Degraded historical documents
Hand-written documents
George Washington dataset
Bentham dataset
Japanese handwriting recognition

ABSTRACT

In word spotting literature, many approaches have considered word images as temporal signals that could be matched by classical Dynamic Time Warping algorithm. Consequently, DTW has been widely used as a on the shelf tool. However there exists many other improved versions of DTW, along with other robust sequence matching techniques. Very few of them have been studied extensively in the context of word spotting whereas it has been well explored in other application domains such as speech processing, data mining etc. The motivation of this paper is to investigate such area in order to extract significant and useful information for users of such techniques. More precisely, this paper has presented a comparative study of classical Dynamic Time Warping (DTW) technique and many of its improved modifications, as well as other sequence matching techniques in the context of word spotting, considering both theoretical properties as well as experimental ones. The experimental study is performed on historical documents, both handwritten and printed, at word or line segmentation level and with a limited or extended set of queries. The comparative analysis is showing that classical DTW remains a good choice when there is no segmentation problems for word extraction. Its constrained version (e.g. Itakura Parallelogram) seems better on handwritten data, as well as Hilbert transform also shows promising performances on handwritten and printed datasets. In case of printed data and low level features (pixel's column based), the aggregation of features (e.g. Piecewise-DTW) seems also very important. Finally, when there are important word segmentation errors or when we are considering line segmentation level, Continuous Dynamic Programming (CDP) seems to be the best choice.

1. Introduction

The advancement of high quality document digitization has provided a stirring alternative to preserve and easy, hassle-free access of ancient manuscripts for historians and researchers. To allow searching into these mass of digitized datasets, indexation based on Optical Character Recognition (OCR) or manual (semi-manual) transcriptions is applied traditionally. Nevertheless, the performance of available OCR engines on such historical documents are not up-to the mark because of the writing and font style variability, linguistics and script dependencies and poor document quality caused by high degradation effects. Even when learning is possible, this one becomes a burdensome process due to the need of ground truth. Whereas, the process of manual or semi-automatic transcription of handwritten or printed documents is a tedious and costly job. For these reasons, word-spotting technique appears to be an interesting alternative and research on this topic has been emphasized. This technique can be defined as the *"localization of words of interest in the dataset without actually interpreting the content"* and it allows to index or search inside a document using queries.

For spotting words in handwritten manuscripts and historical printed document images, word images can be thought as 2D signals, that can be matched by sequence matching algorithms like DTW [14,17,32]. In other application domains, DTW's variants have been intensively evaluated to demonstrate their interest [7,34], but they have not been clearly studied and compared in the case of word spotting. In this paper, we propose a detailed comparative study of DTW and it's variants for word spotting. This study extends the one performed in [27] by including more sequence matching algorithms. Some of them have never been tested in word spotting context whereas they have shown promising results

* Corresponding author.
E-mail addresses: tanmoy.mondal@univ-lr.fr, tanmoy.besu@gmail.com (T. Mondal), nicolas.ragot@univ-tours.fr (N. Ragot), jean-yves.ramel@univ-tours.fr (J.-y. Ramel), umapada@isical.ac.in (U. Pal).
[1] The Matlab implementation of this article is available here: https://github.com/tanmayGIT/ICDAR-2015-DTW.

http://dx.doi.org/10.1016/j.patcog.2017.07.011
0031-3203/© 2017 Published by Elsevier Ltd.

Table 1
Extracted features from the word images, considering an image with N columns and M rows.

Feat. Nb.	Feature description
$F1$	Sum of foreground pixel intensities in pixel columns (from gray scale image)
$F2$	Number of background-to-ink transitions in pixel columns
$F3$	Upper profile of sequence (top most position of ink pixels in pixel columns)
$F4$	Lower profile (bottom most position of ink pixels in pixel columns)
$F5$	Distance between upper and lower profile (in number of pixels)
$F6$	Number of foreground pixels in pixel columns
$F7$	Center of gravity (C.G.) of the column obtained from the foreground pixels $1 \leq n \leq N$ $F7(n) = \begin{cases} [\frac{1}{\rho}\sum_{m=1}^{M} m \quad if\ w_b(m,n) = 1]; \mid \rho \neq 0; \rho = No.\ of\ foreground\ pixels\ at\ n^{th}\ column; \\ t \quad \text{Obtained by interpolation;} \mid \rho = 0 \end{cases}$
$F8$	Transition at C.G. obtained from $F7$ $F8(n) = \begin{cases} 1 \quad w_b(F7(n),n) = 0; \text{ and } w_b(F7(n-1),n) = 1 \text{ or} \\ \quad w_b(F7(n),n) = 1; w_b(F7(n-1),n) = 0 \\ t \quad \text{Obtained by interpolation} \end{cases}$

in other domains. Also, more experimental datasets are used (six in total), including both handwritten and printed document images.

The remainder of this paper is organized as follows. The datasets used for experiments as well as the word spotting framework are detailed in Section 2. The baseline of DTW approach and various other dynamic programming (DP) paths, warping constraints are studied in Section 3. The specific techniques to reduce the quadratic time complexity of DTW algorithm are next evaluated in Section 4. Behavior of several other approaches designed for improving the quality of DTW are studied in Section 5. Other dynamic programming based sequence matching approaches, which has shown better performance than classical DTW in several other domains e.g. shape matching, time series signal matching etc. are experimented in Section 7. Finally, a summary of results with discussion and future work is presented in Section 8.

2. Descriptions of datasets and experimental protocols

2.1. Feature extraction

For all experiments and datasets used, the comparison between a query (word image) and a target (word image or text line(piece of) image) is done by transforming text images into a vector sequence using classical features; such as column based features (please see [28]) or Slit Style HOG features [36].

Column-based features : For an image with a width of N pixels, 8 statistical features, $F_1, F_2, \ldots, F_8$ (Table 1) are computed from left to right on each pixel columns. The features $F1 - F6$ have been used earlier in the literature [25,9,33,8]. The feature $F7$ corresponds to the center of gravity of the foreground pixels inside a column. The equation in Table 1, w_b denotes the binarized version of the word image, m represents the row's coordinate and n is the column's coordinate. For columns that has no foreground pixels, $F7$ is calculated by nearest neighbor interpolation with the help of neighboring columns, having foreground pixels. $F8$, which is calculated by using location of the center of gravity (obtained from $F7$) is the number of transitions from foreground to background (1 to 0) or from background to foreground (0 to 1) at these calculated centroids.

Slit style HOG (SSHOG) based features : For calculating SSHOG features (refer to [36]), a fixed sized window is slided over the image in a horizontal direction to extract the HOG features from each slit.

2.2. Datasets

The description of the 6 datasets used and corresponding features used are mentioned below.

Dataset-1 (GW-15) : This dataset is the handwritten manuscript of *George Washington(GW)*[2], consisting of 20 pages of letters, orders, and instructions of *George Washington* from 1755 century. The quality of scanned pages, at 200 dpi, varied from clean to difficult to read by human and the pages originates from large collection, written not only by *George Washington* but also by some of his associates. This experimental dataset is created from the complete set, by considering 10 better quality pages among the 20. Here we consider 15 query images, used in [22,36]. The statistics related to the 15 queries are mentioned in Table 2. This dataset is analyzed by using column-based features.

Dataset-2 (CESR-10) : This is a machine printed historical dataset coming from the resources of the Centre d'Études Supérieures de la Renaissance (CESR)[3], through the BVH (Bibliothèques Virtuelles Humanities[4]) project. The languages in the books are Latin or French. All the pages are scanned with a resolution of 312×312 and saved in grey scale format. Since word segmentation is difficult on these images, pseudo words (i.e. words, piece of words or piece of lines) were extracted with *Horizontal Run Length Smoothing Algorithm* (HRLSA). Moreover, there might be more variations in the spelling of words here than on Dataset-GW-15 since French language from this Renaissance period had some alteration. In this dataset, we selected 10 queries and the pages, where the query (or one of its direct derivative) appears (see Table 2) for testing. Column-based features are used here too for analyzing this dataset.

Dataset-3 (GW-HOG) : This dataset is also created from GW manuscripts by using same 15 queries but instead of segmented words, we consider properly segmented lines and HOG features. Please note that all the data, corresponding to feature values and ground truth (GT) are provided by the author of [36].

Dataset-4 (Japanese-HOG) : It is a historical Japanese handwritten script having 1576 segmented lines and 4 query words [36]. The segmented lines, HOG features and GT are provided by the author of [36].

Dataset-5 (GW-90) : This dataset is also created from GW dataset. It consists of 4860 segmented words, extracted from 20 pages. To choose the queries, we grouped all the same words (from ground truth) inside individual clusters. A total of 1211 clusters are created, representing 1211 unique words (please note that, we ignored "," "-" "." for clustering the words). Then by keeping clusters corresponding to words with more than 3 characters and with at least 10 representatives gives 45 clusters. Then, one query image was randomly chosen from each of the clusters. This whole

[2] http://www.fki.inf.unibe.ch/databases/iam-historical-document-database/washington-database.

[3] http://cesr.univ-tours.fr/.

[4] See Bibliothèques Virtuelles Humanistes Project: http://www.bvh.univtours.fr/presentationen.asp.

Table 2
Statistics of all queries of CESR and GW datasets.

	$\overline{N^o}$	$\overline{\overline{N^o}}$	Query word		$\overline{\overline{N^o}}$	$\overline{N^o}$
CESR	26(0)	21 (779)	victoire	Chrestien	18 (1449)	15(0)
	18(0)	17 (1471)	mortel	cheval	21 (1586)	24(0)
	20(0)	22 (1361)	liberté	Cicero	23 (1786)	21(0)
	27(0)	22 (811)	François	vaisseau	11 (520)	10(0)
	21(0)	22 (1442)	parmy	tantost	13 (804)	20(0)
GW	27(1)	×	Captain	1755	×	37(0)
	17(0)	×	Regiment	Company	×	20(1)
	15(1)	×	October	Cumberland	×	13(1)
	33(0)	×	Orders	December	×	26(0)
	12(0)	×	Recruits	Fort	×	22(0)
	12(0)	×	Sergeant	Instructions	×	21(1)
	14(0)	×	Virginia	Letters	×	22(0)
	15(4)	×	Winchester			

$\overline{\overline{N^o}}$ = No. of Pages (No. of words) ; $\overline{N^o}$ = No. of full (hyphenated) occurrences.

process was repeated for another time. So, a total of $45 \times 2 = 90$ query images were selected and all word images were described by column-based features.

Dataset-6 (Bentham) : This dataset [10] consists of a series of documents from the Bentham collection, prepared in the tranScriptorium project[5]. The Bentham dataset mainly includes manuscripts written by Jeremy Bentham (1748-1832) himself over a period of sixty years, as well as fair copies written by Bentham's secretarial staff. Word level GT, 95 queries and 3234 segmented words are provided with the dataset, which has the resolution of 300 dpi. Again, column-based features are used for the analysis.

2.3. Experimental protocol

Classical word spotting framework is used for experiments which is broadly described in our earlier published work [28]. Only matching techniques are changed to evaluate performances on some specific datasets by differing the features from one dataset to another.

Before performing matching, irrelevant word images (with respect to the query size) are pruned by using simple properties of images for GW-90 and Bentham dataset. Fine tuning of threshold was avoided; instead single, primitive and simple threshold values were used. The following constraints was used for pruning:

$$\frac{1}{2} \geq \frac{query\ aspect\ ratio}{target\ aspect\ ratio} \geq 2 \tag{1}$$

$$\frac{1}{2} \geq \frac{query\ area}{target\ area} \geq 2 \tag{2}$$

$$\frac{1}{2} \geq \frac{\mathscr{K}\text{ of query}}{\mathscr{K}\text{ of target}} \geq 2 \tag{3}$$

Pruning by Eq. (1) does not provide high pruning rate. Due to handwriting variability, words belonging to the same category can often have quite different lengths and areas. For this reason, two other pruning techniques are used. One is based on area of the bounding boxes and the other one is a rough estimation of number of characters in the word. The assumption on bounding box area will not work because words having similar area can have different scale factor. Nevertheless, this constraint has good pruning capabilities without much lowering the recall. The second criterion, based on number of characters, is an intuitive idea for pruning, which is unaffected by the problem of different scaling and multi writer issues. We used a simple technique for estimating number of characters ($\mathscr{K}$), present in a word [5].

By observing and experimenting the pros and cons of each pruning criterion separately, it was observed that combining them together logically provide better robustness and higher recall. Hence all target images, satisfies condition-1 (Eq. (1)) and condition-2 (Eq. (2)) or condition-3 (Eq. (3)) are considered as relevant images. Although this pruning technique is unable to retrieve all relevant words and misses only few of them for every query, but it is a good trade-off between simplicity and high pruning rate.

3. Evaluation of dynamic time warping methods

DTW [31] is a technique for measuring similarity between two different time series by finding their best correspondence. Let's assume, two 2D signal : $X = x_1, x_2, x_3, \ldots, x_p$ and $Y = y_1, y_2, y_3, \ldots., y_q$. To align these two sequences using DTW, we construct an $p \times q$ matrix, where the (ith, jth) element of the matrix contains the distance ($\mathfrak{D}(x_i, y_j)$) between two points x_i and y_j (i.e. $\mathfrak{D}(x_i, y_j) = (x_i - y_j)^2$).[6] The best warping path (W) between these sequences, is a contiguous (described below) set of matrix elements, which defines an optimal mapping between X and Y. The k^{th} element of W is defined as $w_k = (i, j)_k$, where (i, j) is the corresponding indices in the two sequences (i.e. the element x_i of X is mapped with y_j of Y). The optimal warping path has the following property: $W = w_1, w_2, \ldots., w_K;\quad max(p, q) \leq K \leq p + q - 1$. This warping path maintains the following constraints:

i) **Boundary conditions:** $w_1 = (1, 1)$ and $w_K = (p, q)$. The boundary condition restrict the warping path to start and finish in diagonally opposite corner cells of the matrix, thus it forces to match the first elements and last elements of the sequences together.

ii) **Continuity:** The warping path is always continuous, i.e. if $w_k = (m, n)$ and $w_{k-1} = (u, v)$ then $m - u \leq 1$ and $n - v \leq 1$.

[5] For the detail of this dataset, please see ICDAR-2015 "*Keywords spotting*" competition website : http://transcriptorium.eu/~icdar15kws/data.html.

[6] In this paper, we have considered Euclidean distance, but any other distances can also be considered.

This restricts the warping path to always go through adjacent cells (including diagonally adjacent cells) and thus this constraint forces to match all elements of both the sequences.

iii) **Monotonicity** The warping path is always monotonically spaced in time i.e. if $w_k = (m, n)$ and $w_{k-1} = (u, v)$ then $m - u \geq 0$ and $n - v \geq 0$.

Many warping paths satisfies the above mentioned constraints. However the goal here is to find the best one that minimizes the warping cost, obtained by calculating path cost matrix ($\mathfrak{P}_{(i,j)}$) by using dynamic programming techniques (refer to Eqs. (4) and (5)).

$$DTW(X, Y) = \mathfrak{P}_{(p,q)} \tag{4}$$

$$\mathfrak{P}_{(i,j)} = \mathfrak{D}_{(i,j)} + min \begin{cases} \mathfrak{P}_{(i,j-1)} \\ \mathfrak{P}_{(i-1,j-1)} \\ \mathfrak{P}_{(i-1,j)} \end{cases} \Bigg|_{i=2,\ldots,p;\, j=2,\ldots,q}$$
$$\mathfrak{P}_{(1,1)} = \mathfrak{D}_{(1,1)}; \underset{1<i\leq p}{\mathfrak{P}_{(i,1)}} = \mathfrak{P}_{(i-1,1)} + \mathfrak{D}_{(i,1)}; \underset{1<j\leq q}{\mathfrak{P}_{(1,j)}} = \mathfrak{P}_{(1,j-1)} + \mathfrak{D}_{(1,j)} \tag{5}$$

For a detailed discussion on classical DTW and some of it's variants, which have been applied in time series matching problems, please see [2,7].

3.1. DTW with varying step size condition

The warping path formed by classical DP path of DTW, does not have much options (only vertical, horizontal and diagonal), for finding the optimal correspondences. To avoid such situation, in the following section, we recall other step size conditions of different DP-paths that were introduced previously.

3.1.1. Definition

Various DP-paths have been proposed in the literature [34]. These ones are obtained by adding some weights (w_d, w_h, $w_v \in R^3$) on the DP paths to favor vertical (v), horizontal (h) or diagonal (d) directions. The equally weighted case ($w_d, w_h, w_v = 1, 1, 1$) reduces to classical DTW and ($w_d, w_h, w_v = 2, 1, 1$), for example, will influence the warping path to take the horizontal and vertical directions. Then this DP-path is asymmetric. More complex directions can also be considered (see Table 3). Depending on data, these different DP paths could help to improve performance of DTW matching over classical DTW approach, but as mentioned in [34], "the problem of their relative superiority (asymmetric ones) has been left unsolved". This is why we have evaluated several of them in our word spotting context. In Table 3, symmetric DP paths are represented in cyan color, whereas asymmetric ones are in blue.

3.1.2. Comparative evaluation

The performance of these different DP paths, is investigated through the experiments with different datasets. The results on Dataset-GW-15 and Dataset-CESR-10 are shown in Fig. 1. It is visible from the curves that the best performing DP-Paths are: 0-Sym_2 (classical DTW), 0.5-Asym and 3-Sym. In general symmetrical DP-path are better, except for the 0.5 DP-path[7]. Nevertheless, the performance of these three techniques are almost same on both datasets, with small differences only. If we compare the results between Dataset-GW-15 and Dataset-CESR-10, it is visible that accuracy is significantly higher on Dataset-CESR-10. This is mainly due to the handwritten nature of script in Dataset-GW-15.

When these DP-paths are applied on comparatively large datasets, i.e. Dataset-GW-90 and Dataset-Bentham, almost similar behavior is visible i.e. 0-Sym_2 (classical DTW), 0.5-Asym and 3-Sym have outperformed others (see Fig. 2). Again, symmetrical DP-path are better, which implies that adding weights for favoring any directions of a DP path, seems not suitable for word spotting applications. Also, the difference in performance between Dataset-GW-90 and Dataset-Bentham comes from the nature of scripts and the level of noise present in these datasets. Finally, we can also see that the results are clearly dependent on query used since results here are far less than previous ones, where only few queries were used. But one should also consider that among the 90 queries used on such datasets, many could have no meaning for end-users and will not be considered for research.

3.2. Global constraints on warping paths

Classical DTW is computationally expensive and the common way to reduce it is to impose global constraints on the admissible warping path. Two such well known constraints are described below.

3.2.1. Definition

Along with the improvement of computational complexity, global constraints on warping path can also prevents pathological alignments. The most widely used constraints in the literature are Sakoe-Chiba (SC) band and Itakura Parallelogram. Sakoe-Chiba band is a global constraint. It runs along the diagonal and has a fixed (horizontal and vertical) width r. This constraint implies that an element x_i can only be aligned with an element y_j such as $j \in [\frac{p-r}{q-r}.(q-r), \frac{p-r}{q-r}.(q+r)] \cap [1:p]$. Sakoe-Chiba band can be obtained by the following Eq. (6):

$$i - r \leq j \leq i + r; \quad 1 \leq i \leq p; \text{and } 1 \leq j \leq q. \tag{6}$$

The size of the SC-Band can be varied by changing the value of r. The final performance highly depends on the chosen value for r. Unfortunately, tuning this value is data dependent and is difficult to set. For a general constrained region, denoted by R, the constraint warping path p_R can be computed similarly to the unconstrained one by setting $\mathfrak{P}(x_i, y_j) = \infty$ for all $(i, j) \in [\{1: N\} \times \{1: M\}] \backslash R$. So, in the computation of p_R, only the cells that lie inside R are evaluated. Introducing constraints would definitively speedup the process, for example, in the case of Sakoe Chiba band with a fixed width i; $i << p$; $i << q$, only $O(i \times max(p, q))$ computations need to be performed instead of $O(pq)$ as required in classical DTW.

The pseudo code for Itakura parallelogram is presented in Algorithm 1. Since, experimental evaluations have shown that the

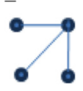
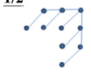
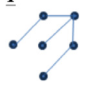
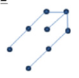
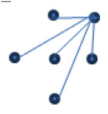
Algorithm 1: Itakura paralleogram.

```
Input: $i(i \in [1, \ldots p])$, $j(j \in [1, \ldots, q])$, $p$(length of query),
       $q$(length of target)
Output: $bool$ (output is 1 if $i$ and $j$ are within Itakura
        window, 0 otherwise)
1 $bool = 0$;
2 if $\{j < 2\} \times \{i \,\&\, i \leq 2 \times j\} \,\&\, \{i \geq (p-1) - (2 \times (q-j))\} \,\&$
  $\{j > (q-1) - (2 \times (p-i))\}$ then
3     $bool = 1$;
4 return $bool$
```

performance of SC-Band and Itakura parallelogram [7] are similar, Itakura parallelogram seems a bit more interesting because

[7] Please notice that only 3-Sym has been tested here whereas 0-Sym_1 and 0-Sym_2 in Table 3 represents symmetric versions of 0 DP-paths. 0_Sym_2 is the classical DTW, which equally favors all three directions but 0_Sym_1 favors vertical and horizontal directions over the central one.

Table 3
Symmetric and Asymmetric DP paths with various slope constraints. The text in cyan represents symmetric equations, while the one in blue color represents asymmetric DP paths.

0 	0-Sym_1: $\min\begin{pmatrix} \mathfrak{P}(i, j-1) + \mathfrak{D}(i, j) \\ \mathfrak{P}(i-1, j-1)+2\times\mathfrak{D}(i, j) \\ \mathfrak{P}(i-1, j) + \mathfrak{D}(i, j) \end{pmatrix}$	0-Sym_2: $\min\begin{pmatrix} \mathfrak{P}(i, j-1) + \mathfrak{D}(i, j) \\ \mathfrak{P}(i-1, j-1)+\mathfrak{D}(i, j) \\ \mathfrak{P}(i-1, j) + \mathfrak{D}(i, j) \end{pmatrix}$
1/2 	0.5-Sym: $\min\begin{pmatrix} \mathfrak{P}(i-1, j-3)+2\times\mathfrak{D}(i, j-2) + \mathfrak{D}(i, j-1) + \mathfrak{D}(i, j) \\ \mathfrak{P}(i-1, j-2)+2\times\mathfrak{D}(i, j-1) + \mathfrak{D}(i, j) \\ \mathfrak{P}(i-1, j-1) + 2\times\mathfrak{D}(i, j) \\ \mathfrak{P}(i-2, j-1) + 2\times\mathfrak{D}(i-1, j) + \mathfrak{D}(i, j) \\ \mathfrak{P}(i-3, j-1) + 2\times\mathfrak{D}(i-2, j) + \mathfrak{D}(i-1, j) + \mathfrak{D}(i, j) \end{pmatrix}$	0.5-Asym: $\min\begin{pmatrix} \mathfrak{P}(i-1, j-3)+(\mathfrak{D}(i, j-2) + \mathfrak{D}(i, j-1) + \mathfrak{D}(i, j))/3 \\ \mathfrak{P}(i-1, j-2)+(\mathfrak{D}(i, j-1) + \mathfrak{D}(i, j))/2 \\ \mathfrak{P}(i-1, j-1) + \mathfrak{D}(i, j) \\ \mathfrak{P}(i-2, j-1) + \mathfrak{D}(i-1, j) + \mathfrak{D}(i, j) \\ \mathfrak{P}(i-3, j-1) + \mathfrak{D}(i-2, j) + \mathfrak{D}(i-1, j) + \mathfrak{D}(i, j) \end{pmatrix}$
1 	1-Sym: $\min\begin{pmatrix} \mathfrak{P}(i-1, j-2)+2\times\mathfrak{D}(i, j-1) + \mathfrak{D}(i, j) \\ \mathfrak{P}(i-1, j-1)+2\times\mathfrak{D}(i, j) \\ \mathfrak{P}(i-2, j-1)+2\times\mathfrak{D}(i-1, j) + \mathfrak{D}(i, j) \end{pmatrix}$	1-Asym: $\min\begin{pmatrix} \mathfrak{P}(i-1, j-2)+(\mathfrak{D}(i, j-1) + \mathfrak{D}(i, j))/2 \\ \mathfrak{P}(i-1, j-1)+\mathfrak{D}(i, j) \\ \mathfrak{P}(i-2, j-1)+\mathfrak{D}(i-1, j) + \mathfrak{D}(i, j) \end{pmatrix}$
2 	2-Sym: $\min\begin{pmatrix} \mathfrak{P}(i-2, j-3)+2\times\mathfrak{D}(i-1, j-2)+2\times\mathfrak{D}(i, j-1) + \mathfrak{D}(i, j) \\ \mathfrak{P}(i-1, j-1)+2\times\mathfrak{D}(i, j) \\ \mathfrak{P}(i-3, j-2)+2\times\mathfrak{D}(i-2, j-1)+2\times\mathfrak{D}(i-1, j) + \mathfrak{D}(i, j) \end{pmatrix}$	2-Asym: $\min\begin{pmatrix} \mathfrak{P}(i-2, j-3)+2\times(\mathfrak{D}(i-1, j-2)+\mathfrak{D}(i, j-1) + \mathfrak{D}(i, j))/3 \\ \mathfrak{P}(i-1, j-1)+\mathfrak{D}(i, j) \\ \mathfrak{P}(i-3, j-2)+\mathfrak{D}(i-2, j-1)+\mathfrak{D}(i-1, j) + \mathfrak{D}(i, j) \end{pmatrix}$
3 	3-Sym: $\min\begin{pmatrix} \mathfrak{P}(i-1, j-1) + \mathfrak{D}(i, j) \\ \mathfrak{P}(i-2, j-1) + 2\times\mathfrak{D}(i, j) \\ \mathfrak{P}(i-1, j-2) + 2\times\mathfrak{D}(i, j) \\ \mathfrak{P}(i-1, j) + \mathfrak{D}(i, j) \\ \mathfrak{P}(i, j-1) + \mathfrak{D}(i, j) \end{pmatrix}$	

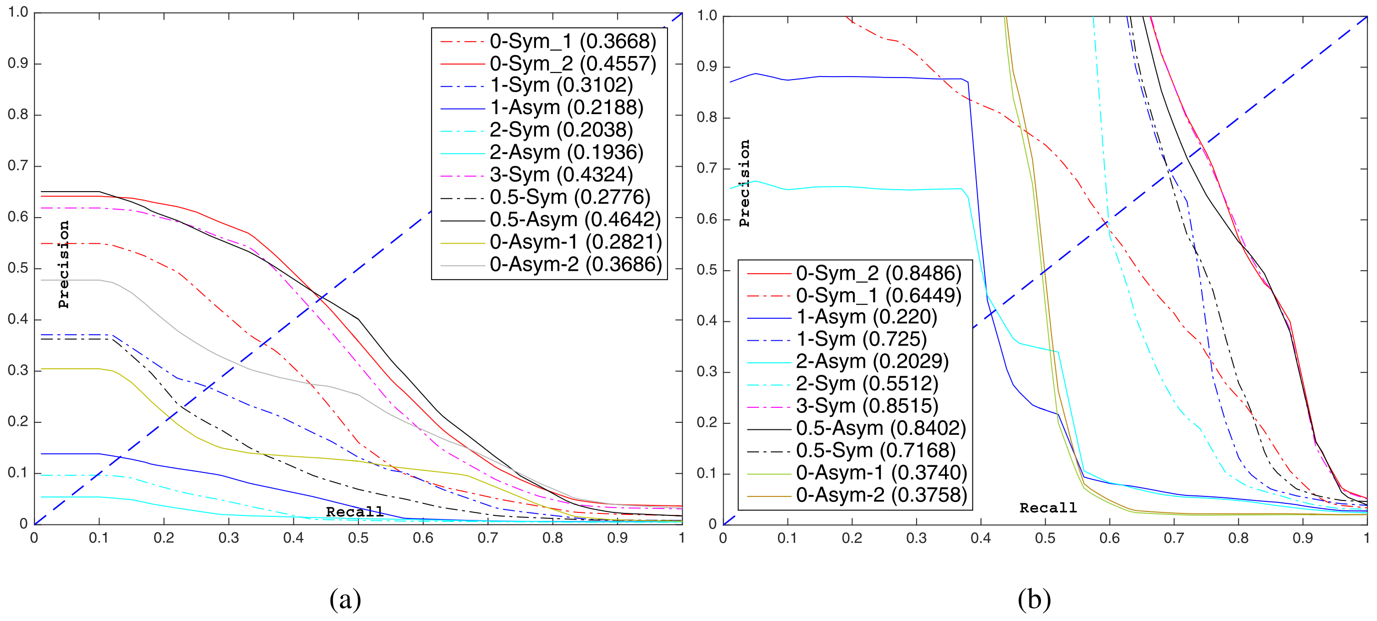


Fig. 1. a) Performance comparison of different DP paths on Dataset-GW-15. b) Performance comparison of different DP paths on Dataset-CESR-10.

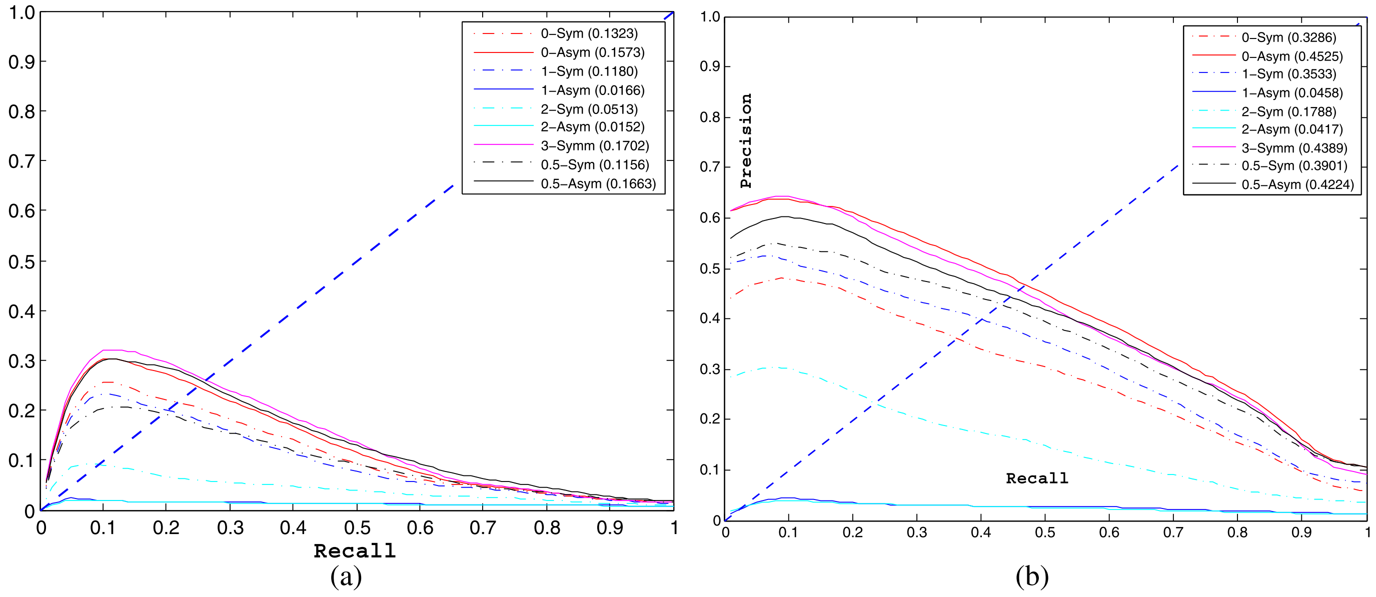


Fig. 2. a) Performance comparison of different DP paths on Dataset-GW-90. b) Performance comparison of different DP paths on Dataset-Bentham.

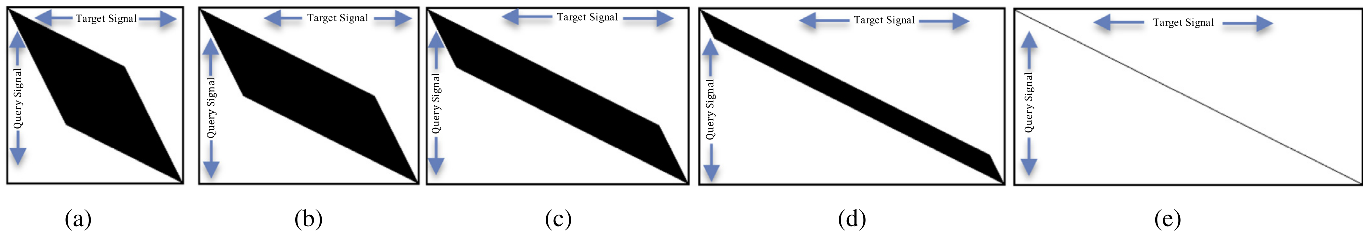


Fig. 3. The change in the shape of Itakura parallelogram with the change in length of target signal.

there are no tunable parameters. The corresponding DTW, constrained by Itakura parallelogram is denoted as ($\underline{\underline{\text{DTW}}}$) in the following sections. Before going into the discussion of another sequence matching technique, we would like to discuss here about one important bottleneck of *Itakura parallelogram*. This bottleneck has a big impact on word spotting results. By analyzing the mentioned pseudo code of Itakura parallelogram, it is visible that when $q \geq 2 \times p$, *Itakura parallelogram* could not be formed. We consider a toy example to demonstrate the problem. As shown in Fig. 3, when the length of target signal is gradually increased, the shape of Itakura parallelogram gets changed and it gradually becomes thinner. The most appropriate shape of Itakura parallelogram can be visible in the left most figure (see Fig. 3a), where $p = 200$ and $q = 200$, i.e. when the two sequences have the same size. But when $p = 200, q = 250$ (see Fig. 3b), $p = 200, q = 300$ (see Fig. 3c), $p = 200, q = 350$ (see Fig. 3c) and $p = 200, q = 399$ (see Fig. 3d) respectively, the change in shape of Itakura Parallelogram can be visible. When $p = 200$ and $q = 400$, (see Fig. 3d) i.e. when the target's length is double of query's length, we could not formulate Itakura parallelogram. This dependency of Itakura parallelogram formation on the length of query and target signal is an obvious factor but it has not been well mentioned in the literature [31]. Moreover, no concrete solution is provided in the literature to overcome this problem, except re-sampling. In the experimental section, we propose a simple technique to handle this problem for word spotting.

3.2.2. Experimental protocol and results

To evaluate the effects of these constraint based DP paths, we tested them on Dataset-GW-15 and Dataset-CESR-10. Fixing the proper radius of SC-Band is a cumbersome task. It takes a lot of manual efforts to find the best performing radius of SC-Band and needs an experimental dataset with ground truth information. We have experimentally set the optimized radius as 23% (30%) of the length of target sequence for Dataset-GW-15 (Dataset-CESR-10)[8]. The Precision-Recall (PR) plot on Dataset-GW-15 and Dataset-CESR-10 shows that both constraints have performed equally well even if Itakura Band (without any parameter tuning) has slightly outperformed SC-Band. But the most interesting observation here is that for the case of Dataset-GW-15, we see that constraining the DP path significantly improves the result over classical DTW (0-Asymmetric). The mAP[9] value of SC-Band and Itakura Parallelogram are 0.5876 and 0.6017 for Dataset-GW-15 whereas mAP of

[8] Please note that, we did the optimization with full dataset. Consequently, the results are optimal for these datasets and we could not expect to achieve the similar performance in real cases.

[9] The average precision (*AveP*) for retrieving one query is defined by:

$$AveP = \frac{\sum_{i=1}^{\kappa}(P_i \times rel_i)}{\text{number of relevant words}} \tag{7}$$

κ = number of considered ranks for query$q$

$$mAP = \frac{\sum_{q=1}^{Q} AveP(q)}{Q} \tag{8}$$

Q = total number of queries

Precision P_i is defined as the number of retrieved relevant word instances divided by index i, corresponding to number of words, we would like to retrieve. Recall R_i is defined as the number of relevant word instances, retrieved at index i, divided by the total number of existing relevant words in the dataset. The rel_i is an indicator function equal to 1, if the retrieved word at rank i is relevant, zero otherwise. The mean average precision (mAP) for a set of queries is obtained by calculating the mean of the average precision scores for each query.

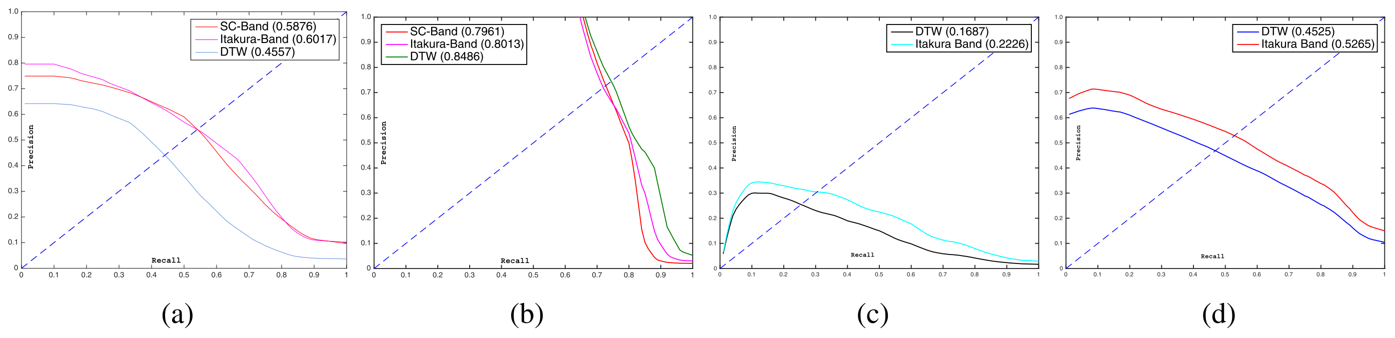


Fig. 4. a) Performance comparison of DTW constraints on Dataset-GW-15. b) Performance comparison of DTW constraints on Dataset-CESR-10. a) Performance comparison of DTW constraints on Dataset-GW-90. b) Performance comparison of DTW constraints on Dataset-Bentham.

classical DTW was 0.4576, which correspond to an increase of 15% in accuracy. But for Dataset-CESR-10, this trend cannot be observed. Indeed here, classical DTW has outperformed constrained DTW. The most probable reason of these kinds of behavior is that for Dataset-GW-15, constraining the warping path is helpful as it limits pathological matching, which could occur because of the nature of handwriting. On Dataset-CESR-10, we must remember that there exist segmentation problems and also some linguistic variations. Consequently, some flexibility is needed in the matching to compensate these variations and segmentation problems.

Due to the complexity for choosing optimal r, and because previous results shows that SC-Band and Itakura parallelogram has almost similar performance, we did not use SC-Band for our experiments with Dataset-GW-90 and Dataset-Bentham. The performance of Itakura Parallelogram on Dataset-GW-90 and Dataset-Bentham, can be seen from Fig. 4a and Fig. 4b respectively. The same trend of improvement (i.e. Itakura Parallelogram has outperformed classical DTW on both datasets) can be visible from these two datasets also, which are also handwritten texts as Dataset-GW-15.

4. Speeding up DTW

Beside the global constraints already mentioned in Section 3.2, some other techniques to reduce the time and space complexity of DTW (which is $O(mn)$), can be broadly classified into following two categories.

4.1. Data abstraction based techniques

An effective strategy to quickly calculate DTW distance is to perform the alignment on coarsened versions of the sequences X and Y, thus reducing the lengths of the two considered sequences. Such a strategy is also known as dimensionality reduction or data abstraction. In the following subsections, we present several of such data abstraction based techniques.

4.1.1. Piecewise DTW (PDTW)

Most time series data can be efficiently approximated by piecewise aggregates, so that DTW can operate on higher level of data [7]. The main idea of PDTW is to reduce the size of original signal by capturing the significant representations of the full signal through sub-sampling technique. Let $\mathfrak{R}$ be the reduced dimension of transformed time series ($1 \leq \mathfrak{R} \leq p$). Although not necessary but for convenience of explanation let's assume $\mathfrak{R}$ is a factor of p. So a time series X of length p is represented by a vector of $\hat{X} = \hat{x}_1, \hat{x}_2, \ldots, \hat{x}_{\mathfrak{R}}$. The i^{th} element of $\hat{X}$ is calculated by the equation: $\hat{x}_i = \frac{\mathfrak{R}}{p} \sum_{j=\frac{p}{\mathfrak{R}}(i-1)+1}^{\frac{p}{\mathfrak{R}}i} x_j$. Simply stated, to reduce the data from p dimensions to $\mathfrak{R}$ dimensions, the data is divided into $\mathfrak{R}$ equal sized *"frames"*. The mean value of the data falling within a frame is calculated and a vector obtained from these values become the data reduced representation (also called as sub-sampling). Two special cases worth noting are when $\mathfrak{R} = p$, the transformed representation is identical to the original representation. When $\mathfrak{R} = 1$, the transformed representation is simply the mean of the original sequence. This transformation produces a piecewise constant approximation of the original sequence, this approach is named as *Piecewise Aggregate Approximation (PAA)*. The compression rate $\mathfrak{c} = \frac{p}{\mathfrak{R}}$ is denoted by the ratio of the length of original sequence to the length of its PAA represented sequence. In choosing a value for $\mathfrak{c}$, there is a classic trade-off between memory savings and fidelity. The *"best"* compression rate depends on the structure of data itself and the task (i.e. clustering/classification/retrieval etc). This approach of dimensionality reduction is applied on both the query (X) and target (Y) signals then DTW is applied on the reduced signals i.e. on $\hat{X}$ and $\hat{Y}$.

4.1.2. Fast-DTW

Another data abstraction [35] based technique known as Fast-DTW works by initially finding the optimal path through a coarse representation of data and then it refines it to the original data. This is achieved by using following mentioned three stages.

Coarsening: Generate reduced representation of the time series that represents the same curve as accurately as possible with less data points.

Projection: From reduced representation of the data, the minimum distance warping path at the lower resolution is calculated and used as an initial guess for high resolution's minimum distance warp path.

Refinement: Refine the warping path, projected from lower resolution into the warping path of higher resolution through local adjustment of the warping path.

Fast-DTW runs in $O(n)$ time with sufficient accuracy, and thus this approach speeds up the classical DTW method significantly without much loss of accuracy.

4.1.3. Sparse DTW

To compute DTW optimal distance between two time series, a new space-efficient approach (Sparse DTW) is proposed in [1]. Contrary to other techniques, which typically sacrifice optimality to attain space efficiency, this one maintain optimality, while improving speed. This technique dynamically exploits the existence of similarity and/or correlation between the time series. The amount of similarity between two time series is proportional to the amount of space required for computing DTW. The constraint band of Sparse DTW, evolves dynamically and are much smaller than the constraint band of conventional approaches (i.e. SC-Band and Itakura Parallelogram). The warping matrix is represented by using sparse matrices, which leads to better average space complexity. Sparse DTW always yields the optimal warping path because it never have to set apriori constraints, independently of the data (contrary to SC-Band and Itakura Parallelogram). Thanks to this ability, it can be easily used in conjunction with lower bound approaches [16].

4.1.4. Accurate and Fast DTW (AF_ DTW)

Classical DTW defines cumulative distance in $\mathfrak{P}_(i, j)$ by adding minimum distance of three adjacent elements in forward direction. Whereas, *Accurate and Fast DTW (AF_DTW)* is computed by using backward strategy [23]. AF_DTW starts from (p, q) to $(1, 1)$ and elements of $\mathfrak{P}$ is calculated by taking maximum of three right adjacent elements subtracted by the distance $\mathfrak{D}(i, j)$. After forming the path cost matrix by Eq. 9, the warping path $(P' = \{p_1, p_2, ..., p_K\})$ is calculated in the same way as DTW.

$$\underset{i=p,p-1,....,1;\,j=q,q-1,......,1}{\mathfrak{P}_{(i,j)}} = max\begin{cases}\mathfrak{P}_{(i,j+1)}\\ \mathfrak{P}_{(i+1,j+1)}\\ \mathfrak{P}_{(i+1,j)}\end{cases} - \mathfrak{D}_{(i,j)} \tag{9}$$
$$\mathfrak{P}_{p+1,q+1} = 0;\ \mathfrak{P}_{p,q+1} = \mathfrak{P}_{p+1,q} = -\infty$$

The constructed dynamic programming based warping path also follows three constraints : boundary condition, continuity and monotonicity. This warping path is used to calculate the best warping path:

$$AF\_DTW(X,Y) = \max_{P}\sum_{k=1}^{K}\mathfrak{D}(p_K) \tag{10}$$

AF_DTW has the same accuracy and absolute distance value as of classical DTW. In addition, the time and space complexity of AF_DTW is reduced by adopting a strategy, inspired by Itakura and Sakoe Chiba band. The values in $\mathfrak{P}$ depends on the initial value at $\mathfrak{P}_{(p+1,q+1)}$. If $\mathfrak{P}_{(p+1,q+1)}$ is initially set to zero then all the elements in $\mathfrak{P}$ would be negative but if $\mathfrak{P}_{(p+1,q+1)}$ is set to a positive value then some of the elements in $\mathfrak{P}$ would be positive and these positive elements would be adjacent. It is shown by some toy examples in [23] that the best warping path would always appear in the scope of these positive cells. Let's consider, $\mathfrak{P}_{(p+1,q+1)} = \theta$ as the initial value then the best warping path would always exists in the reduced scope with regard to the special value of θ (refer to [23]). So, if the algorithm is forced to find the best warping path in the reduced scope, (i.e. only through the positive cells) the time and space complexity could be reduced significantly. The main problem is to choose a good value of θ so that the best warping path could always be surrounded by the positive cells. It is shown in [23] that a good approximation of θ is $\theta = \sum_{k=1}^{n}(x_p - y_p)^2; n = p = q$.

4.2. Experimental protocol and results

We experimented the above mentioned approaches, on Dataset-GW-15 and Dataset-CESR-10[10]. Due to the inherent architecture of Fast-DTW, it will always speedup with some loss of accuracy. From Fig. 5, it can be seen that on both datasets, Fast-DTW has slightly lower accuracy than the classical DTW but with almost half computational time than DTW. For PDTW, the compression rate $(\mathfrak{c})$ is set to the *average character width* for Dataset-CESR-10 and (2 × *average stroke width*) for handwritten dataset[11]. Depending on datasets and features, PDTW has variable accuracy (refer to Fig. 5). It is worse than DTW on Dataset-GW-15 whereas it is better on Dataset-CESR-10. It might be because, in printed images of Dataset-CESR-10, with a resolution of 300 dpi, there might be more redundancies in the signals when it is based on low level features (pixels's column based) and this could disturb the matching process. Then, the data compression effect of PDTW could be useful on such dataset. On the contrary, on handwritten data (see also Fig. 7a and 7b) and especially on GW dataset with a lower resolution of 200 dpi, PDTW seems to reduce the information, making it less or equally efficient. But in all cases, PDTW is always faster than classical DTW (refer to Table 6).

4.3. Indexing based techniques

For large time series classification or clustering based problems, these kind of techniques introduce lower bounding functions for reducing the number of times, DTW must be run during the complete process. Iterative Deepening DTW is one such example [6]. It is based on a model, in which the distribution of distance approximation errors can be described and can be calculated by the approximation of DTW at increasingly finer levels of representations. The algorithm begins by approximating the query and target signals (i.e. X & Y) by PAA approximation technique at an initial dimensionality reduction level of d_1 and given a user confidence (or user defined acceptable tolerance value for false dismissal), the algorithm determines whether Y (target signal) could be potential object for possible expansion or not (by calculating the approximate distance between reduced dimensional query and target signals). If so, the algorithm computes the distance between X and Y with more precise approximation, using a lower dimensionality reduction rate d_{low} $(D_{PDTW}(d_{low}))$. This process goes on until d_{min} or stopped because no further expansion is needed/possible. At that point, the approximation becomes the true warping distance D_{DTW} between X and Y. This technique is suitable for dataset, containing large amount of time series signals of big lengths (approximatively in thousands). In our case, signal length (features extracted from word images) are not very large (mostly in hundreds). So, this algorithm would not be much useful and that's why we have not tested it.

5. Improving the quality of DTW

Here, we discuss the techniques proposed in the literature for improving the performance of DTW.

5.1. Derivative dynamic time warping (DDTW)

The typical condition of classical DTW may lead to unexpected singularities (the alignments between a point of a series with multiple points of the other series) and unintuitive alignments. To overcome these weaknesses of DTW, DDTW transforms the original points into higher level features, which contain the structural information of the signal. This technique considers the first derivative of the signals instead of original signals $(X = x_i; 1 \le i \le p$; $Y = y_j; 1 \le j \le q)$ [7], which intrinsically helps to handle the presence of noise in the signal. The first derivative of the signal is calculated as follows:

$$\bar{x}_m = \frac{(x_i - x_{i-1}) + ((x_{i+1} - x_{i-1})/2)}{2}; 2 \le i \le p-1; 1 \le m \le p-1$$
$$\bar{y}_n = \frac{(x_j - x_{j-1}) + ((x_{j+1} - x_{j-1})/2)}{2}; 2 \le j \le q-1; 1 \le n \le q-1 \tag{11}$$

5.2. Piecewise Derivative Dynamic Time Warping (PDDTW)

The *Piecewise Derivative Dynamic Time Warping* (PDDTW) [38], uses at the same time the derivative of the signal in order to reduce singularities and data abstraction for capturing the benefits of both PDTW and DDTW together. In order to align two sequences X and Y, a reduced dimensional series $\hat{X}$ and $\hat{Y}$ are obtained first.

[10] Notice that the results of Sparse DTW and AF_DTW are not presented here because they have not shown significant increase in computation time and space as claimed.

[11] Average character width is calculated by obtaining individual characters or glyphs by connected component labeling technique, whereas average stroke width for handwritten characters is calculated by the technique mentioned in [4].

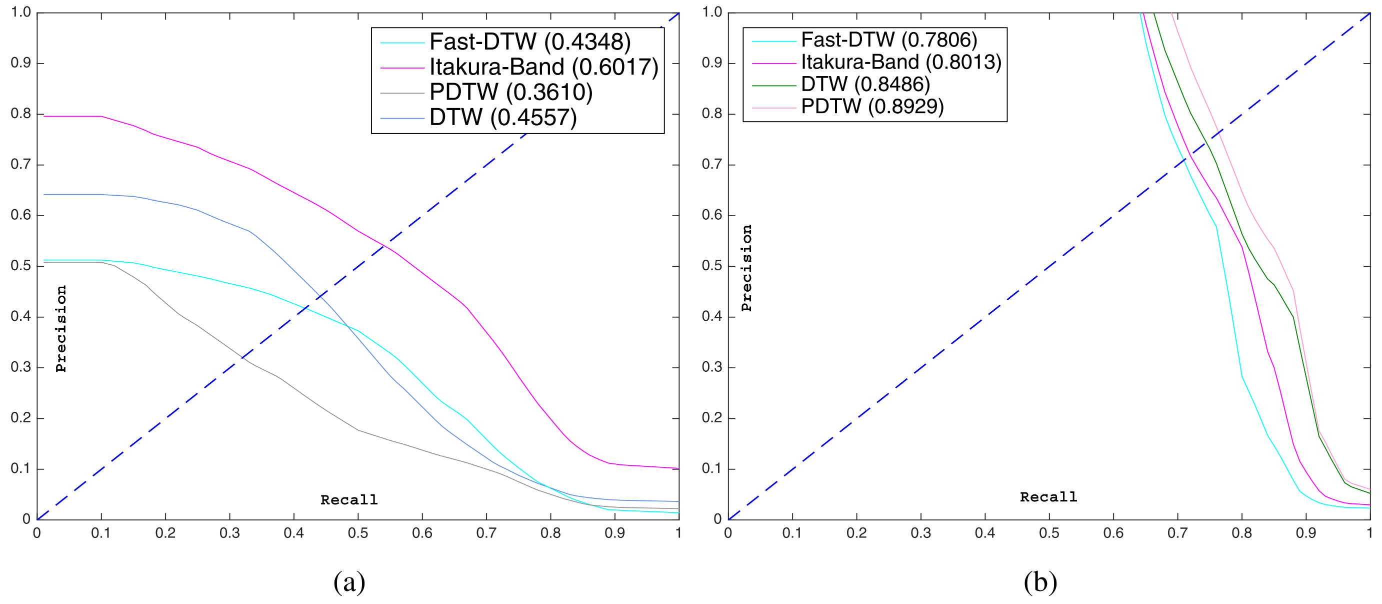


Fig. 5. a) Performance comparison of Fast-DTW and PDTW in comparison with classical DTW on Dataset-GW-15. b) Performance comparison of Fast-DTW and PDTW in comparison with classical DTW on Dataset-CESR-10.

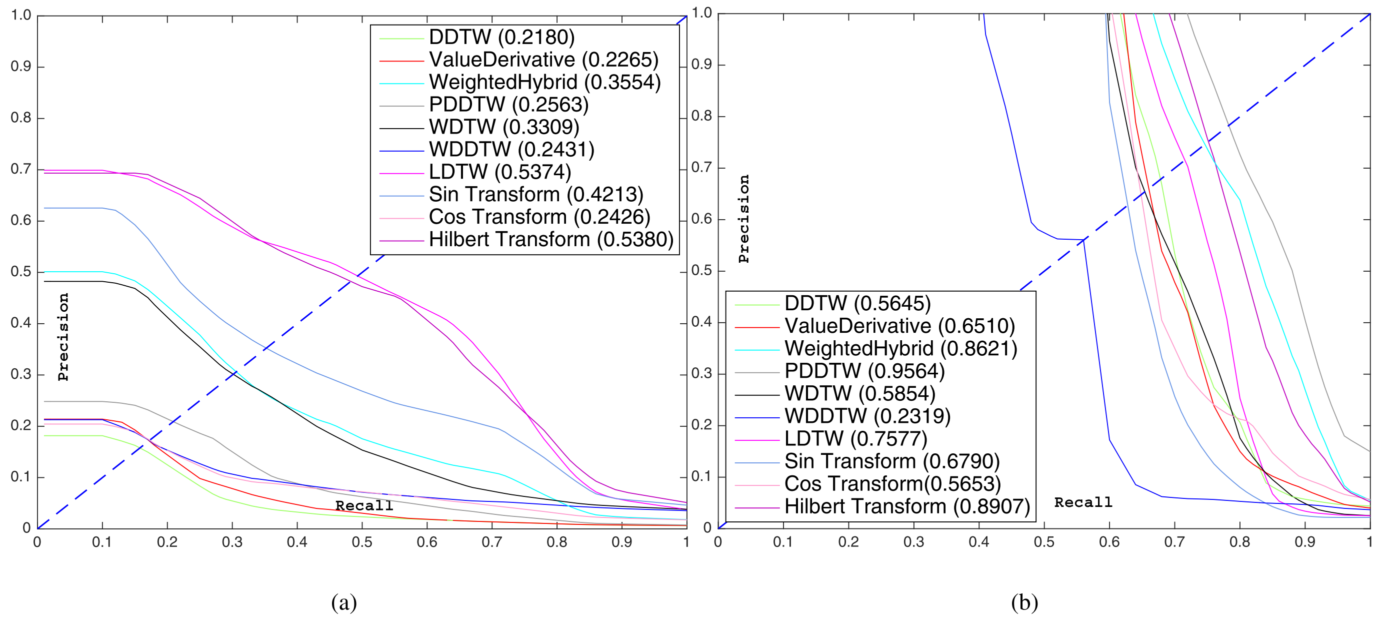


Fig. 6. a) Performance comparison of different algorithms for improving the quality of DTW on Dataset-GW-15. b) Performance comparison of different algorithms for improving the quality of DTW on Dataset-CESR-10.

The length of these reduced dimensional signals are $\frac{p}{\phi}$ and $\frac{q}{\phi}$, where the term ϕ denotes the sampling frequency for *piecewise aggregate approximations*. After that the derivative of these sampled signals ($\bar{\bar{X}}$ and $\bar{\bar{Y}}$) are calculated in the same manner as DDTW. Then, the distance matrix ($\mathfrak{D}(\bar{\bar{X}}, \bar{\bar{Y}})$) between two signals $\bar{\bar{X}}(=\bar{\bar{x}}_s; 1 \le s \le (\frac{p}{\phi}-1))$ and $\bar{\bar{Y}}(=\bar{\bar{y}}_t; 1 \le t \le (\frac{q}{\phi}-1))$ is calculated in the same manner as DDTW and the remaining same steps are followed to calculate the distance.

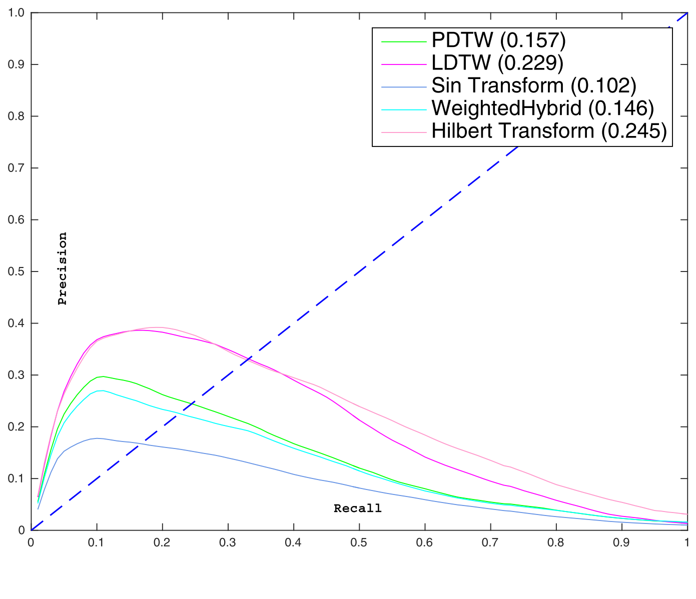
5.3. Non isometric transforms based DTW

In [12], the authors proposed to use mathematical functions other than the derivative, to transform the signals. The idea is to choose some non-isometric transforms to calculate distances, which can bring some extra information. In [12], authors have proposed to use three popular non isometric transforms : Cosine transform, Sine transform and Hilbert transform, which seems to be useful in the domain of sequence matching. For a series $X = \{x_i : i = 1, 2, ..., p\}$, it's transform $\bar{X} = \{\bar{x}_k : k = 1, 2, ..p\}$ is obtained by:

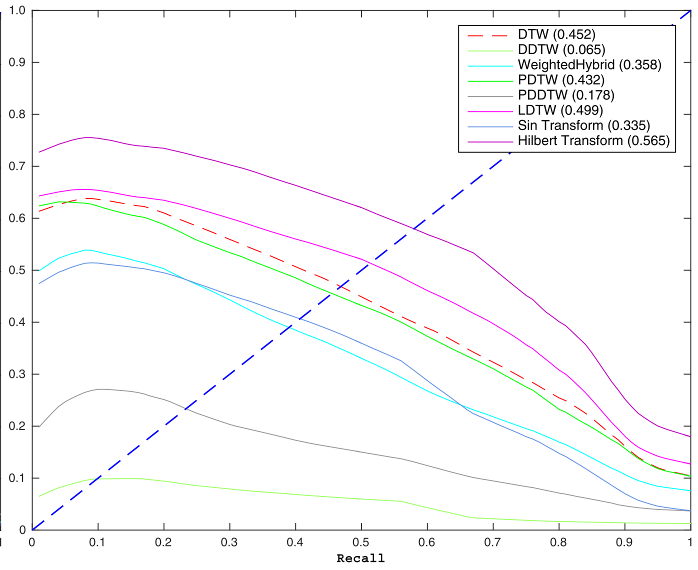
Cosine transform	Sine transform	Hilbert transform
$\bar{x}_k = \sum_{i=1}^{p} x_i \cos[\frac{\pi}{p}(i-\frac{1}{2})(k-1)]$	$\bar{x}_k = \sum_{i=1}^{p} x_i \sin[\frac{\pi}{p}(i-\frac{1}{2})(k-1)]$	$\bar{x}_k = \sum_{\substack{i=1 \\ i \neq k}}^{p} \frac{x_i}{k-1}$

5.4. Value Derivative DTW

Standard DTW uses Euclidean metric for calculating the distance between the elements of target and query sequences. This distance is good to compare single points but not appropriate for comparing the vector sequences. One more intelligent way [18] to calculate the distance is by giving consideration to adjacent val-



(a)



(b)

Fig. 7. a) Performance comparison of different algorithms for improving the quality of DTW on Dataset-GW-90. b) Performance comparison of different algorithms for improving the quality of DTW on Dataset-Bentham.

ues of time series, which is sensitive on the local changes among time series elements. The derivative of query and target signals are calculated by $\bar{x}_i = x_i - x_{i-1}$ and $\bar{y}_i = y_i - y_{i-1}$. On the basis of DDTW, this algorithm [18] propose to calculate the distance matrix ($\mathfrak{D}(x_i, y_j) = \sqrt{(x_i - y_j)^2.(\bar{x}_i - \bar{y}_j)^2}$) between these signals in a little bit different manner (notice the different process of calculation of the $\mathfrak{D}$, compared to classical DTW). The first part in the above equation gives the information about offset between points to be compared. The second part adds the "intelligence" to the entire measure. After calculating this distance matrix, the classical DTW based DP-path formation technique is applied to obtain the distance.

5.5. Weighted hybrid derivative dynamic time warping (WHDDTW)

The information from raw signal contains useful information and smoothing the raw signal helps to stabilize the process. Moreover, the derivative provides better knowledge. For example, the first derivative gives information on speed and the second derivative gives information on accelerations and decelerations. To handle the noise sensitiveness of DDTW, an improvement is proposed in [3]. The distance matrix ($\mathfrak{D}^{WHDDTW}_{i,j}$) is computed in the following manner, where $\mathfrak{D}_{i,j}$ represents distance between ith and jth elements of query and target signals. Then, $\bar{\mathfrak{D}}_{i,j}$ and $\bar{\bar{\mathfrak{D}}}_{i,j}$ represents the distance between 1st and 2nd order derivative of query and target signal; $\mathfrak{D}^{WHDDTW}_{i,j} = w_0.\mathfrak{D}_{i,j} + w_1\bar{\mathfrak{D}}_{i,j} + w_2.\bar{\bar{\mathfrak{D}}}_{i,j}$. The path cost matrix and final distance between query and target signal is calculated as classical DTW.

5.6. Weighted dynamic time warping (WDTW)

The standard DTW calculates the distance of all points with equal penalization of each point regardless of the phase difference between a reference point and a testing point (difference between indices). The WDTW algorithm [15] penalizes the points according to this phase difference between target and query elements to prevent minimum distance distortion by outliers. The key idea is that, if the phase difference is low, smaller weight is imposed (i.e. less penalty is imposed) because neighboring points are important, otherwise larger weight is used. While creating the $p \times q$ path cost matrix, the distance between the two points x_i and y_j is calculated by $\mathfrak{D}_w(x_i, y_j) = \|W_{|i-j|}(x_i - y_j)\|_p$, where $W_{|i-j|}$ is a positive weight value between the two points x_i and y_j. The optimal distance between two sequences is defined as the minimum path over all possible paths as follows: $WDTW_p(X, Y) = \sqrt[p]{\mathfrak{P}(i, j)}$ where, $\mathfrak{P}(i, j) = |W_{|i-j|}(x_j - y_j)^p| + min\{\mathfrak{P}(i-1, j-1), \mathfrak{P}(i-1, j), \mathfrak{P}(i, j-1)\}$. A technique called as *"Modified logistic weight function(MLWF)"* is used to systematically assign weights as a function of phase difference between two points. The parameter g controls the amount of penalization considering phase difference. The weight value $W_{(t)} = \frac{W_{max}}{1+e^{(-g(t-m_c))}}$; where $t = 1, ..., p$; p is the length of a sequence and m_c is the midpoint of a sequence. W_{max} is the desired upper bound for the weight parameter and g is an empirical constant that controls the level of penalization for the points with larger phase difference (from zero to infinity).

5.7. Weighted derivative dynamic time warping (WDDTW)

The idea of weighted dynamic time warping can be extended to variants of DTW, for example the idea of derivative dynamic time warping can be extended to it's weighted version, denoted as : $WDDTW_p(X_i, Y_j) = WDTW_p(\bar{X}_i, \bar{Y}_j)$ where $\bar{X}$ and $\bar{Y}$ are the 1st order derivative of query and target signals respectively.

5.8. Local dynamic time warping (LDTW)

DTW algorithm is modified here to perform pseudo-local alignment using some specific DP-paths [24] at different location of path cost matrix ($\mathfrak{P}$) for handling stretching and compression of individual points in time series data. The DTW equation are changed in the following way. The general equation is :

$$\underset{1\le i\le p-1;1\le j\le q-1}{\mathfrak{P}(i, j)} = \mathfrak{D}(i, j) + min$$

$$\times \begin{cases} \mathfrak{P}(i-1,j-1) \\ \mathfrak{P}(i-2,j-1)+\mathfrak{D}(i-1,j)+\frac{1}{3} \\ \mathfrak{P}(i-1,j-2)+\mathfrak{D}(i,j-1)+\frac{1}{3} \\ \mathfrak{P}(i-3,j-1)+\mathfrak{D}(i-2,j)+\mathfrak{D}(i-1,j)+\frac{2}{3} \\ \mathfrak{P}(i-1,j-3)+\mathfrak{D}(i,j-2)+\mathfrak{D}(i,j-1)+\frac{2}{3} \end{cases} \tag{12}$$

For the last column the equation is:

$$\mathfrak{P}(i,j)_{1\le i\le p-1;\, j=q} = min$$

$$\times \begin{cases} \mathfrak{P}(i-1,j-1) \\ \mathfrak{P}(i-2,j-1)+\mathfrak{D}(i-1,j) \\ \mathfrak{P}(i-1,j-2)+\mathfrak{D}(i,j-1)+\frac{1}{3} \\ \mathfrak{P}(i-3,j-1)+\mathfrak{D}(i-2,j) \\ \mathfrak{P}(i-1,j-3)+\mathfrak{D}(i,j-2)+\mathfrak{D}(i,j-1)+\frac{2}{3} \end{cases} \tag{13}$$

For the last row the equation is:

$$\mathfrak{P}(i,j)_{i=p;\, 1\le j\le q-1} = min$$

$$\times \begin{cases} \mathfrak{P}(i-1,j-1) \\ \mathfrak{P}(i-2,j-1)+\mathfrak{D}(i-1,j)+\frac{1}{3} \\ \mathfrak{P}(i-1,j-2)+\mathfrak{D}(i,j-1) \\ \mathfrak{P}(i-1,j-3)+\mathfrak{D}(i,j-2) \\ \mathfrak{P}(i-3,j-1)+\mathfrak{D}(i-2,j)+\mathfrak{D}(i-1,j)+\frac{2}{3} \end{cases} \tag{14}$$

and finally for the last row and last column, it is

$$\mathfrak{P}(i,j)_{i=p;\, j=q} = min \begin{cases} \mathfrak{P}(i-1,j-1), \\ \mathfrak{P}(i-2,j-1)+\mathfrak{D}(i-1,j) \\ \mathfrak{P}(i-1,j-2)+\mathfrak{D}(i,j-1) \\ \mathfrak{P}(i-3,j-1)+\mathfrak{D}(i-2,j) \\ \mathfrak{P}(i-1,j-3)+\mathfrak{D}(i,j-2) \end{cases} \tag{15}$$

To find the warping path, we must look through the last row and the last column and find the cell that has the smallest value. That cell is the end of a local alignment, and the warping path can be found by tracking back from that cell until we reach the first row or column. Consequently, this algorithm also has the specific property to do partial matching by discarding elements of query/target at the begin or end of the sequences.

5.9. Continuous DTW (CDTW)

Here, a sample point in one of the signal is allowed to be matched with an implicit point lying between two sample points of the other signal [29]. This method compares planar curves which enables it to perform matching at sub-sampling resolution, which can be useful when we know that sources of signals does not have a good acquisition of frequency. Due to the intrinsic properties of this algorithm, it is difficult to properly adapt it to word image matching problem. Moreover, if resolution of images are good and sampling for feature extraction is high enough (e.g. our datasets) then, there is no need of such process. Finally, it might be possible to over-sample the signal to obtain similar behavior without the need of new algorithm [31]. Consequently, we have not implemented and applied this approach for our evaluation purpose.

5.10. Experimental protocol and results

We performed experiments with 4 datasets for evaluation of above mentioned algorithms. The g parameter of WDTW and WDDTW was set by using the full dataset and by trying to optimize the value of the parameter[12]. For WHDDTW, as mentioned in [3], we used $w_0 = 1$, $w_1 = 2$ and $w_2 = 2$, which gives satisfactory results here. The experiments conducted on Dataset-GW-15 and Dataset-CESR-10 (refer to Fig. 6) are showing some interesting behavior. On Dataset-GW-15, *Hilbert transform* has outperformed all other techniques, whereas *Local DTW* is second best. Also, *Hilbert transform* and *Local DTW* are performing better than classical DTW but their accuracy remains inferior to the one of constrained DTW (SC-Band or Itakura, see Fig. 4). Sine transform performed quite well compared to other algorithms but could not outperform classical DTW. Finally, derivative and weighting based approaches are also unable to improve results of classical DTW.

In the case of printed words (Dataset-CESR-10), derivative-based approaches are not working well if applied on the raw signal (performance is less than DTW). But when combined with PDTW, the accuracy is the best one obtained on this dataset, 5% better than PDTW that was previously the best approach. Again, we can explain this by the fact that if you are working with a signal containing some redundancies (because of high resolution as well as printed nature), combined with local noise, then using derivation is not working properly. After having signal compression with piecewise aggregation, derivation becomes efficient, contrary to what was observed on handwritten words. On this printed dataset, *Hilbert transform* and *Weighted Hybrid DTW* are also providing some improvements in comparison with classical DTW but weighted version are difficult to parameterize. Finally LDTW is not working properly which is surprising considering segmentation problems of this dataset.[13]

After observing interesting behavior of different aforementioned algorithms on Dataset-GW-15 and Dataset-CESR-10, we choose some of the best performing techniques to study them comparatively on bigger datasets (see Fig. 7). We experimented first, many of them on Dataset-Bentham and then applied only the best ones on Dataset-GW-90, which is a bigger dataset but nearly of same nature (both are handwritten). It is visible from the results on both datasets that a similar behavior is observed as the one on Dataset-GW-15. More precisely, Hilbert transform is outperforming other techniques whereas LDTW is second best. We can also notice that Hilbert transform is even outperforming Itakura Band on both datasets, which was previously the best algorithms on these datasets (it was not observed on the smaller dataset i.e. Dataset-GW-15). LDTW is also better than Itakura but only on Dataset-GW-90. As before, weighted and derivative-based approaches are not accurate in comparison with DTW.

6. Finding subsequence with DTW

All of the above mentioned algorithms was designed for matching all elements of the sequences. But none of these above mentioned techniques can handle subsequence matching, which is specially needed in word spotting especially for Dataset-GW-HOG and Dataset-Japanese-HOG. In this section, we speak about simple modifications of classical DTW for subsequence matching.

6.1. DTW with correspondence window (DTW-CW)

When a query sequence has to be matched with a subsequence of a target sequence, the matching can be performed by using a

[12] Like SC-Band parameter, weighting factors (g) for WDTW (WDDTW) are heuristically set to 0.31 (0.26) for Dataset-GW-15 and to 0.03 (0.01) for Dataset-CESR-10, using same data for testing. Even with such favoring conditions (not applicable in real cases), the approaches are not showing promising results to generate further interest on such method.

[13] One explanation could be that the specific DP-path is not adapted to the printed nature of data combined with high resolution images and low level features.

sliding correspondence window, having same length as query, and an overlapping of (ζ) [19]. The position of optimal subsequence within the target sequence is obtained by calculating DTW distance inside each window position over a query sequence. Consequently, DTW-CW is computationally expensive and choosing the value of ζ could also be troublesome for some applications.

6.2. Subsequence DTW (SSDTW)

This algorithm can find a continuous subsequence that would optimally match the shorter query sequence [2]. Let $X = x_1, x_2, ...x_p$ and $Y = y_1, y_2, ...y_q$; $q \gg p$ be the query and target sequences. The goal is to find the continuous indices from a^* to b^* of Y so that $Y(a^* : b^*) = (y_{a^*}, y_{a^*+1}, ..., y_{b^*})$ with $1 \leq a^* \leq b^* \leq q$ that minimizes the DTW distance to X. In other words: $(a^*, b^*) = \underset{(a^*,b^*):1\leq a^*\leq b^*\leq q}{\text{argmin}} \; DTW(X, Y(a^* : b^*))$. This technique works by relaxing the boundary conditions of classical DTW. The formation of warping path (backtracking) can start from any column at the last row of path cost matrix ($\mathfrak{P}$) and can end at any column at the first row. $\mathfrak{P}$ is initialized in the following manner. $\mathfrak{P}_{(1,1)} = \mathfrak{D}_{(1,1)}$; $\underset{1<i\leq p}{\mathfrak{P}_{(i,0)}} = \mathfrak{P}_{(i-1,0)} + \mathfrak{D}_{(i,0)}$; $\underset{1<j\leq q}{\mathfrak{P}_{(0,j)}} = \mathfrak{P}_{(0,j-1)} + \mathfrak{D}_{(0,j)}$. The optimal distance ($\mathfrak{V}$) is stored in the cell $\mathfrak{C}_{start} = [\underset{1\leq i\leq q}{\text{argmin}}\{\mathfrak{P}_{(p,\text{i})}\}]$. The backtracking for warping path calculation also starts from this particular cell, which ends at any cell in 1st row; i.e. at $\mathfrak{C}_{end} = [\underset{1\leq i\leq q}{\text{argmin}}\{\mathfrak{P}_{(1,\text{i})}\}]$. SSDTW has the same computational complexity as DTW.

6.3. Meshesha Jawahar DTW (MJ-DTW)

When target signal differs from reference signal in terms of prefixes and suffixes, the warping path can deviate (from diagonal) in horizontal or in vertical directions at the beginning or at end. Such situation can significantly increases the matching cost and can disturb the matching for the corresponding part of the target and reference signals. To avoid such behavior, a DTW based partial sequence matching technique, dedicated to word spotting, is proposed in [26], for handling the variations at the beginning and at the end of the sequences. To reduce unwanted extra cost, MJ-DTW first analyzes whether the dissimilarity between words is concentrated at the end, at the beginning or both. Then, the extra cost at the two extremes are removed to reduce the total matching score and to obtain the optimal dissimilarity value.

6.4. Experimental protocol and results

To evaluate the performance of three above mentioned algorithms, we performed multiple experiments with different datasets. Even if these methods (except MJ-DTW) are not really defined to operate at word segmentation level, we evaluated them on Dataset-GW-15, Dataset-CESR-10, Dataset-GW-90 and Dataset-Bentham (see Fig. 8a, Fig. 8b, Fig. 8c and Fig. 8d), to see whether they can be used in variable level of segmentation. No parameter tuning are required for SSDTW and MJ-DTW algorithms. In the case of DTW-CW, ζ is taken as ($2 \times$ average stroke width) for Dataset-CESR-10 and average character width for handwritten datasets (i.e. Dataset-GW-15, Dataset-GW-90 and Dataset-Bentham)[14].

From the results, it can be concluded that these approaches are not really efficient on segmented words. Even MJ-DTW seems not so useful, except when many queries are used (probably because targets may change more often when considering many different queries), and especially on Dataset-GW-15. On Dataset CESR-10, results are a bit different: the considered approaches are not working poorly and DTW-CW is even better than DTW. This is clearly understandable since the word segmentation process on this dataset was not so accurate and then there exist some pieces of lines for which SSDTW and DTW-CW could be of interest. Nevertheless, it's high computational cost due to the exhaustive search for obtaining the best match is a bottleneck for DTW-CW (see Section 6.1) and deciding the value of ζ is also a cumbersome process. On the contrary, due to the inherent architecture of SSDTW, it is computationally inexpensive.

To evaluate the performance on datasets with segmented lines (Dataset-GW-HOG and Dataset-Japanese-HOG), we have only experimented SSDTW and DTW-CW. Indeed, MJ-DTW is irrelevant here since this technique is specifically designed for segmented words with pre and/or post conjugations. Fixing the skipping parameter ζ for DTW-CW is a tedious task as before, especially on Dataset-Japanese-HOG (because of the complex script structure of Japanese language). So, we decided to find this threshold experimentally using a set of 3 queries. We performed the experiments by increasing the value of ζ with a gap of 5, i.e. $\zeta = 1; 5; 10; 15; 20; 25; 30$ are considered. Increasing the value of ζ is stopped when the accuracy start to decrease or becomes stable. From Fig. 9a and Fig. 9c[15], it can be visible that the accuracy started to decrease significantly from $\zeta = 10$. Of course, optimal results are obtained with $\zeta = 1$ but it is highly time consuming. Consequently, we considered the value of $\zeta = 5$ as the best trade-off between accuracy and speed ($\zeta = 10$ remain also a good choice). The accuracy obtained by DTW-CW in such way is comparable with the accuracy of SSDTW. However SSDTW is computationally inexpensive and thus is preferable.

By analyzing the algorithms on small datasets, we observed that the inter performance difference is not very high and the accuracies are close to each other.

7. Other relevant sequence matching techniques

There are others relevant sequence matching techniques, which were proposed to overcome some of the architectural drawbacks of DTW by removing some constraints (especially boundary and continuity conditions), which helps these techniques to skip outliers from query and/or target sequences. At the same time, the many-to-one and one-to-many matching property of DTW is missing in these techniques.

7.1. Longest common subsequence (LCSS)

The longest common subsequence dissimilarity measure is an algorithm based on *edit distance* or *Levenshtein distance*. The basic idea is to match corresponding elements, keeping the order and allowing some elements to be unmatched or left out (e.g. outliers). The LCSS [37] measure has two parameters, δ and ϵ. The constant δ, which is usually set to a percentage of the sequence length, is a constrained window size for matching a given point from one sequence to a point in another sequence. It controls how far in time, we can go in order to match a given point from one trajectory to a point in another trajectory. The constant $0 < \epsilon < 1$ is the matching threshold: two points from two sequences can be matched, if their distance is less than ϵ. The performance of LCSS highly depends

[14] Because the dataset is handwritten, it is difficult to have a proper estimation of average character width. So, we decided to consider ($2 \times$ average stroke width) as a rough approximation of character width.

[15] It is not feasible to divide these 4 queries into learning and testing set for obtaining the optimal value of ζ. Moreover, for comparing with other techniques, it is necessary to have all the 4 queries. So, it can be said that the results are bit biased and is favoring DTW-CW on this dataset.

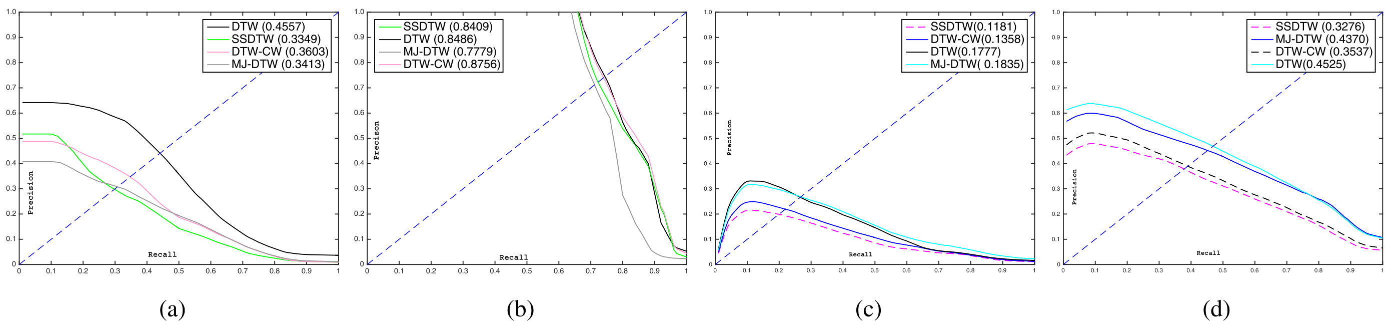


Fig. 8. a) Performance comparison of different algorithms in this section on Dataset-GW-15. b) Performance comparison of different algorithms in this section on Dataset-CESR-10. c) Performance comparison of different algorithms in this section on Dataset-GW-90. d) Performance comparison of different algorithms in this section on Dataset-Bentham.

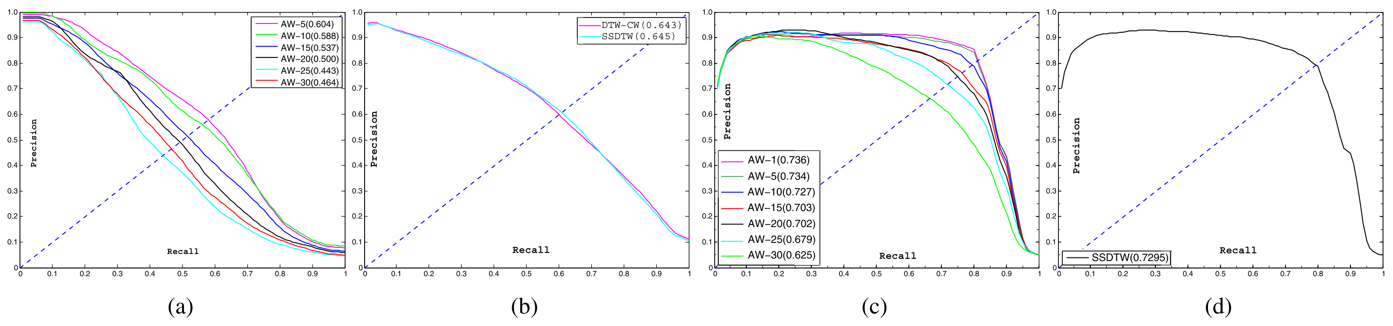


Fig. 9. a) Performance comparison of DTW-CW technique for different values of sliding width (AW) for Dataset-GW-HOG on the learning set of 3 query images. b) Precision-Recall plot for DTW-CW and SSDTW for Dataset-GW-HOG on the complete query set. c) Performance comparison of DTW-CW technique for different values of sliding width (AW) for Dataset-Japanese-HOG on the full query set. d) Precision-Recall plot for SSDTW for Dataset-Japanese-HOG on the full query set.

on the correct setting of this threshold, which may be a difficult problem for some applications.

By considering the ratio between length of the calculated longest common subsequence and that of whole sequence, the dissimilarity between query and target sequence is calculated. Since the inherent goal of finding longest common subsequence is to find optimal substructure between two compared sequences, the problem of LCSS is often solved with dynamic programming. Given two sequences $X = x_1, x_2, ...x_p$ and $Y = y_1, y_2, \ldots, y_q$, the length of their longest common subsequence, denoted as $\mathscr{L}(X, Y)$ is calculated by:

$$\mathscr{L}_{\substack{i,j\\1\le i\le p;1\le j\le q}} = \begin{cases}0 & if \quad i=0 \quad and \quad j=0\\ \mathscr{L}_{i-1,j-1}+1 & if \quad |x_i-y_j| < \epsilon \ and \ |i-j| \le \delta \\ max(\mathscr{L}_{i,j-1}, \mathscr{L}_{i-1,j}) & Otherwise\end{cases} \tag{16}$$

$\mathscr{L}(X, Y)$ takes values from $\frac{p+q-2\mathscr{L}_{p,q}}{p+q}$ to 1 and is expected to perform better than DTW in presence of noise.

7.2. Derivative based longest common subsequence

We can define two variants of LCSS as follows:

i) **1D-LCSS:** In this approach, the derivative ($\bar{X}$ and $\bar{Y}$) of the signals X and Y are considered for computing LCSS, which should make it more robust to noise [11].

ii) **DD-LCSS:** This is an extension of 1D-LCSS, which considers second derivative of time series signal ($\bar{\bar{X}}, \bar{\bar{Y}}$), where, $\bar{\bar{X}}$ and $\bar{\bar{Y}}$ represents the second order derivatives of X and Y [11].

7.3. Minimal variance matching (MVM)

This algorithm [20] is designed to handle partial sequence matching of two sequences : $X = x_1, x_2, ..., x_p$ and $Y = y_1, y_2, ..., y_q$; $p \le q$, which combines the strengths of both DTW and LCSS. Due to this outliers skipping property of MVM, this approach is able to find a subsequence Y' of Y of length p such that X best matches with $Y' \in Y$. The distance matrix $\mathfrak{D}_{(i,j)}$ can be used as a directed acyclic graph (DAG) in which a parent node $\mathfrak{D}_{(k,l)}$ can be linked to a node $\mathfrak{D}_{(i,j)}$ at the next row and its left up to a given extent, which allows to skip some elements. The cost function for linking two elements is defined by:

$$\mathcal{H}(\mathfrak{D}_{k,l}, \mathfrak{D}_{i,j}) = \begin{cases}\mathfrak{D}_{i,j} & \text{if } i = k+1 \text{ and } l+1 \le j \le \mathcal{L}\\ \infty & \text{otherwise}\end{cases} \tag{17}$$

$$\mathcal{L} = (l+1) + elasticity - |j-i|; elasticity = |q-p| \tag{18}$$

Considering a path leading to $\mathfrak{D}_{i,j}$, its cost $\mathfrak{P}(i, j)$ formally defined by:

$$\mathfrak{P}(i,j) = \left(\begin{array}{l} (\mathfrak{P}(i,j))^2 \quad if \quad i = 1; 1 \le j \le q \\ \min \left(\begin{array}{l} \mathfrak{P}(i,j) \\ \{\mathfrak{P}(i-1,k) + \mathcal{H}(\mathfrak{D}_{i-1,k}, \mathfrak{D}_{i,j})\} \end{array} \right) if \quad \mathfrak{L} \\ \infty \quad Otherwise \end{array} \right)$$

$$\mathfrak{L} = 2 \le i \le m; i \le k \le i + (q-p); \\ k+1 \le j \le k+1+(q-p) \tag{19}$$

The optimal structure condition guarantees that the returned matrix $\mathfrak{P}$ contains the cost of the shortest path leading to every node. The path we are looking for, satisfies two conditions: *i)* starting in the first row, between columns 1 and $(q-p)+1$; i.e. at $\mathfrak{D}_{1,j}$ for $j = 1....(q-p)+1$; *ii)* ending at some node in the last row $\mathfrak{D}_{p,j}$ for $j = p....q$. To find it, we just need to check the minimum value of $\mathfrak{P}$ in the last row and at specific columns. After that, we need to back track for obtaining the warping path.

7.4. Optimal sequence bijection (OSB)

This algorithm is an extension of MVM [21] algorithm. It is particularly suitable for partial and elastic matching as it can skip outliers present in query as well as in target. In this way, OSB can also match a query longer than the target, which is not possible with MVM. The goal of OSB is to find subsequences X' of X; $(X' \in X)$ and Y' of Y; $(Y' \in Y)$ such that X' best matches Y'. However, the freedom of skipping elements should also be restricted to prevent unnecessary correspondences. To solve this purpose, a penalty of skipping is introduced, which is called *"jumpCost"*, denoted as $\mathfrak{C}$. The optimal correspondence can be found by generating a DAG, using the distance matrix $\mathfrak{D}$. The nodes of the DAG are all index pairs $(i, j) \in \{1...p\} \times \{1...q\}$ and the edge cost $\mathcal{W}$ between the node (i, j) and (k, l) is defined by:

$$\mathcal{W}\{(i,j)(k,l)\} = \begin{cases} \sqrt{(k-i-1)^2 + (l-j-1)^2}.\mathfrak{C} + \mathfrak{D}(x_k, y_l) & if \quad i<k \bigwedge j<l \\ \infty & \text{otherwise} \end{cases} \tag{20}$$

Thus, the cost of an edge is defined by the Euclidean distance between vertices (i, j) and (k, l) in the matrix $\{1...p\} \times \{1...q\}$ times the jump cost, plus the dissimilarity measure between elements x_k and y_l.

7.5. Continuous dynamic programming (CDP)

CDP[16] [30] is able to perform subsequence matching (finding full query in longer target sequence) and to locate multiple occurrences of the query in the target series.

The path cost matrix ($\mathfrak{P}$) of CDP is obtained in the following way:

$$\mathfrak{P}(j,i)\Big|_{i=1} = 3 \times \mathfrak{D}(j,1) \tag{21a}$$

$$\mathfrak{P}(j,i)\Big|_{i=2} = \min\begin{pmatrix} \mathfrak{P}(j-2,1) + 2\times\mathfrak{D}(j-1,2) + \mathfrak{D}(j,2) \\ \mathfrak{P}(j-1,1) + 3\times\mathfrak{D}(j,2) \\ \mathfrak{P}(j,1) + 3\times\mathfrak{D}(j,2) \end{pmatrix} \tag{21b}$$

$$\mathfrak{P}(j,i)\Big|_{3\le i\le p} = \min\begin{pmatrix} \mathfrak{P}(j-2,i-1) + 2\times\mathfrak{D}(j-1,i) + \mathfrak{D}(j,i) \\ \mathfrak{P}(j-1,i-1) + 3\times\mathfrak{D}(j,i) \\ \mathfrak{P}(j-1,i-2) + \mathfrak{D}(j,i-1) + \mathfrak{D}(j,i) \end{pmatrix} \tag{21c}$$

The output is obtained by $A(j) = \frac{1}{3.p}\mathfrak{P}(j,p)$; where $\mathfrak{P}(j,p)$ is given by:

$$\mathfrak{P}(j,p) = \min_{(1\le i\le p,\ j\ge\beta(i),\ \beta(i+1)\ge\beta(i))} \sum_{i=1}^{i=p} \mathfrak{D}(x(i), y(j-\beta(i)))$$

The above defined formula has an initial condition: $\mathfrak{P}(-1,j) = \mathfrak{P}(0,j) = \infty$. As it can be visible from Equation 21 that more resistance is present for the diagonal link of the DP path compared to other two links.

7.6. Experimental protocol and results

The performance of these sequence matching techniques on segmented words from Dataset-GW-15 and Dataset-CESR-10 are shown in Table 4[17] and Fig. 10. For these experiments, LCSS (1D-LCSS and DD-LCSS) parameters are set in the following way: δ is set to 100% because LCSS is very resistant to the changes of δ [11]. To compute ϵ automatically, we randomly choose two query sequences. For each, we randomly choose two corresponding targets that should match with query. Then the distance matrices between the chosen queries and targets are calculated separately and merged together (let's say it $\mathfrak{D}^{merged}$). Now, ϵ is calculated by: $\epsilon = mean(\mathfrak{D}^{merged}) + 2 \times std(\mathfrak{D}^{merged})$. For 1D-LCSS and DD-LCSS (see Section 7.2), ϵ is calculated by using $\mathfrak{D}^{merged}$, obtained from 1st and 2nd derivative of the signals respectively. It is visible from these results that LCSS family has low accuracy compared to other matching algorithms[18]. CDP outperforms the other techniques here on both the datasets, whereas OSB is also performing comparatively well[19]. But it's high computational complexity (due to it's inherent architecture) is a bottleneck for exploring it on bigger dataset such as Dataset-GW-90 and Dataset-Bentham. In all cases, these approaches remains less accurate than other best performing approaches mentioned in aforementioned sections. It is observed that Itakura-Band and even classical DTW remains better on Dataset-GW-15. Whereas, PDDTW remains better than CDP on Dataset-CESR-10 even if CDP has performed better than DTW, which (DTW) has shown overall good performance on all the datasets.

Table 4
Performance comparison of other relevant techniques on GW-15 dataset

Method name	Accuracy (mAP)
LCSS	0.0354
1DLCSS	0.0489
DDLCSS	0.0489
OSB	0.2785
MVM	0.2026
CDP	0.3619

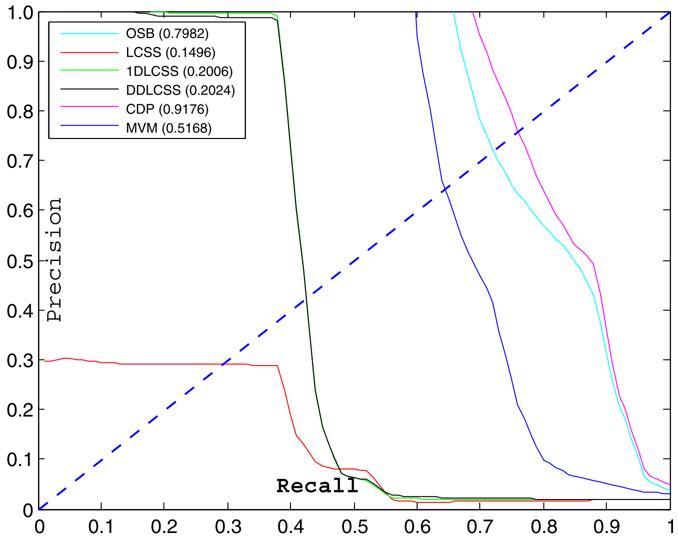


Fig. 10. Performance comparison of other relevant techniques on Dataset-CESR-10.

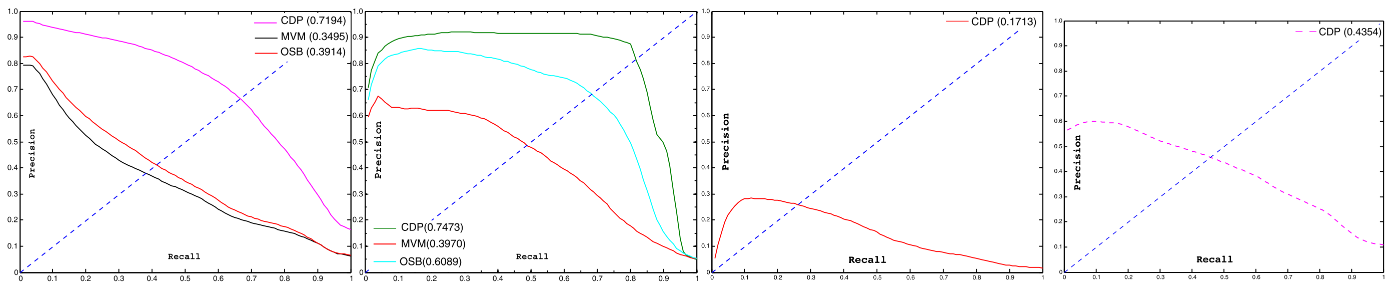
For experiments on segmented lines (Dataset-GW-HOG and Dataset-Japanese-HOG), we only tested CDP, MVM and OSB techniques. The P-R plots are shown in Fig. 11. Thanks to its special DP-path and associated weights, CDP has outperformed MVM and OSB on both the datasets. Moreover, CDP also achieves better accuracy than aforementioned SSDTW. One can notice the drop in performance of MVM and OSB here whereas they were competitive on previous benchmarks. This can be explained by the fact that, in the case of slit style HOG features, wrong jumps performed

[16] The implementation of CDP, used in our experiments, is taken from: http://www.diva-portal.org/smash/get/diva2:347724/FULLTEXT01.pdf. Page no. 86

[17] Please note that, the P-R plot of Dataset-GW-15 is not shown because the accuracy of LCSS family (LCSS, 1DLCS, 2DLCSS) are very low compared to others. Due to this reason, when all the curves are put together in one graph, the curve of LCSS family is difficult to visualize. So, we prefer just to provide the statistical results.

[18] Of course, it might be possible to optimize the parameters of such approaches, especially using learning data but then, the comparison with other approaches would become unfair.

[19] please see Section 3 in [21] to understand the *"jumpCost"* calculation process, we used the same approach in our case.



(a) (b) (c) (d)

Fig. 11. a) Performance comparison of other relevant sequence matching techniques on Dataset-GW-HOG. b) Performance comparison of other relevant sequence matching techniques on Dataset-Japanese-HOG. c) Performance comparison of different DP paths on Dataset-GW-90. d) Performance comparison of different DP paths on Dataset-Bentham.

Table 5
Comparative word spotting accuracy of all the above mentioned sequence matching techniques on six datasets.

Technique	Dataset-GW-15	Dataset-CESR-10	Dataset-GW-HOG	Dataset-Japanese-HOG	Dataset-GW-90	Dataset-Bentham
0-Sym_1	0.3668	0.6449	×	×	0.1323	0.3286
0-Sym_2 (classical DTW)	0.4576	0.8503	×	×	0.1573	0.4525
1-Sym	0.3074	0.6922	×	×	0.1180	0.3533
1-Asym	0.2188	0.220	×	×	0.0166	0.0458
2-Sym	0.2038	0.5512	×	×	0.0513	0.1788
2-Asym	0.1936	0.2029	×	×	0.0152	0.0417
3-Sym	0.4324	0.8515	×	×	0.1702	0.4389
0.5-Sym	0.2776	0.7168	×	×	0.1156	0.3901
0.5-Asym	0.4642	0.8402	×	×	0.1663	0.4224
SC-Band	0.5876	0.7961	×	×	×	×
Itakura parallelogram	0.6017	0.8013	×	×	0.2226	0.5265
DDTW	0.2180	0.5645	×	×	×	0.065
Value Derivative DTW	0.2265	0.6510	×	×	×	×
Weighted Hybrid DTW	0.3554	0.8621	×	×	0.146	0.358
PDTW	0.3466	0.8929	×	×	0.157	0.432
PDDTW	0.2563	0.9564	×	×	×	0.178
WDTW	0.3309	0.5854	×	×	×	×
WDDTW	0.2431	0.2319	×	×	×	×
LDTW	0.5374	0.7577	×	×	0.229	0.499
Sin transform	0.4213	0.6790	×	×	0.102	0.335
Cos transform	0.2426	0.5653	×	×	×	×
Hilbert transform	0.5380	0.8907	×	×	0.245	0.566
SSDTW	0.3349	0.8409	0.645	0.7295	0.1181	0.3276
DTW-CW	0.3603	0.8756	0.643	0.734	0.1358	0.3537
MJ-DTW	0.3413	0.7779	×	×	0.1835	0.4370
LCSS	0.0354	0.1496	×	×	×	×
1DLCSS	0.0489	0.2006	×	×	×	×
DDLCSS	0.0489	0.2024	×	×	×	
OSB	0.2785	0.7982	0.3914	0.6089	×	×
MVM	0.2026	0.5168	0.3495	0.3970	×	×
CDP	0.3619	0.9176	0.7194	0.7473	0.1713	0.4354
Fast-DTW	0.4348	0.7806	×	×	×	×

by MVM or OSB can completely disturb the matching process (because of the size of slits) whereas these jumps could be useful to limit noise impact in column based features used with Dataset-GW-15 and Dataset-CESR-10.

In the case of Dataset-GW-90 and Dataset-Bentham, due to the high computational complexity of OSB and also because CDP was performing better in almost all the datasets, we have only tested CDP. The corresponding P-R curves and mAP values are shown in Fig. 11c and Fig. 11d respectively. The accuracy on both datasets is close to classical DTW, which makes it of little interest on such kind of datasets.

8. Overall comparative analysis of algorithms and conclusions

In this paper, different dynamic programming matching techniques were explored for word spotting purpose. Indeed, there exists a wide variety of variations of the popular DTW, only classical-DTW has been used most of the time without any justification. Our comparison was based on experimental protocols, involving hand-

Table 6
Time required for finding one keyword in GW.

Algorithm Name	Time taken (Sec.)		Algorithm Name
Classical DTW	602.85	397.24	SSDTW
SC-Band	170.08	574.33	DDTW
Itakura Band	240.00	155.28	PDTW
0.5-Symmetric	672.98	172.44	PDDTW
0.5-Asymmetric	575.95	1755.90	DTW-CW
0-Symmetric	608.55	810.50	WDTW
0-Asymmetric	567.62	825.43	WDDTW
1-Symmetric	618.24	649.19	MVM
1-Asymmetric	602.42	8474.15	OSB
2-Symmetric	611.033	404.23	MJ-DTW
2-Asymmetric	597.47	245.30	CDP
3-Symmetric	519.43	208.54	LCSS
LDTW	519.43	2.62	Fast-DTW
		5.23	DTW

written datasets (George Washington, Bentham and a Japanese dataset) and a historical printed document. Two levels of segmentation were considered: word level, with perfect segmentation as well as basic segmentation results (which includes wrong segmentation, i.e. pieces of words or of lines); line level (with good segmentation). Different size of query sets were also used, including small ones, composed only of relevant queries, as well as larger ones which includes many possible queries, except very small and unique words. To describe the word images, column-based features were mainly used.

The mAP values of all algorithms are summarized in Table 5. Most of the highlighted algorithms have shown credible performances on multiple datasets. The conclusions drawn from these results are that classical DTW remains a good option in general for segmented words. Its constrained version with Itakura parallelogram (easier to use than SC-band) on handwritten data seems to prevent pathological matching and provides very good results. It is also faster. But when there are some segmentation problems or local variations in the signal (because of linguistic variations for example) such kind of constraints can deteriorate the performances (e.g. Dataset-CESR-10). Various DP-path and weighting (e.g. 0-Sym_1, 0.5-Sym etc.) do not present significant improvements in accuracy over classical DTW. Only LDTW seems to improve classical DTW but it remains less accurate than Itakura which has a better complexity.

Considering signal transforms, the performance of Hilbert transform on all the 4 datasets is quite impressive. It is better than Itakura on handwritten datasets with many queries and one of the best methods on printed data. On the printed Dataset-CESR-10, only PDTW and PDDTW are better (in-fact the best). This is probably because the column-based features used here, extracted on 300 dpi images and combined with printed nature of data, might introduce redundancies distinguishable only by noise effect. Then using aggregation of data and derivation could be improving the overall signal quality.

Finally, on line segmented datasets, CDP seems to be the best option, considering substring matching. It is also working well on improperly segmented words of Dataset-CESR-10, even if less accurate than PDDTW. The Table 6 is presenting complementary information about time needed to do the matching for one query with all words in Dataset-GW-15. The experiments were performed, by using Intel i7 processor and 4 GB RAM. MatLab-7.14 is used for implementing algorithms except Fast-DTW, in Java. For proper comparison, DTW was also implemented in Java (in blue). The large difference in time factor between the two implementations is clearly visible. As expected, we can see that SC and Itakura Bands, as well as PDTW, are speeding up DTW, whereas, OSB and DTW-CW are much slower, but CDP is faster.

Otherwise, most algorithms have the same time complexity as classical DTW. Please note that any of the dataset out of six is not divided as training & testing set except for the case of 1D-LCSS and DD-LCSS (details of training & testing set is mentioned in Section 7.6).

In future works, several points could be investigated in detail. The very important one is the relationship between the feature level extraction, the resolution and the nature of the data. Finding a relationship between these elements may avoid confusions of choice between DTW and data-abstraction based methods (or to select directly the appropriate method). Another study of interest could be how to combine different approaches. For example, parametric combinations have been studied in the domain of time series matching [12,13]. We tried to apply these ones in our case by combining pairs of algorithms (e.g. DTW with LDTW). The results achieved are not showing a significant increase in accuracy. Moreover, the parametric combination is not easy to obtain and would need training data. Consequently, this kind of combination is probably not the one to investigate at first. On the contrary, combining or doing hybridization between algorithms seems more promising. For example, we tested LDTW constrained with Itakura parallelogram (named as $\underline{\underline{\text{LDTW}}}$, see Algorithm 2). This combination seems

Algorithm 2: $\underline{\underline{\text{LDTW}}}$

```
Input: $p, q, \mathfrak{D}$
Output: $\mathscr{D}$
1 for $i \leftarrow 1$ to $p$ do
2     for $j \leftarrow 1$ to $q$ do
3         $bool = Itakura(i, j, p, q)$;
4         if $bool$ then
5             $\mathfrak{Q}(i, j) \leftarrow LDTW(i, j, \mathfrak{D})$;
6 $\mathscr{D} = \mathfrak{D}(p, q)/|w_k|$                    ▷ The final distance
7 return;
```

interesting since LDTW is a DP based approach, which applies different DP paths at different location of path cost matrix. Without constraints, LDTW specific equation might be counter-productive on some parts of the matching. What we observed is that this combination is giving nearly same results on Dataset-Bentham, whereas it increases the accuracy of 4% on Dataset-GW-90 (from 0.229% for LDTW and 0.226% for Itakura to 0.276% for LDTW constrained by Itakura).

Acknowledgment

This work is partly supported by Indo-French Center for Promotion of Advanced Research (IFCPAR/CEFIPRA). Authors want to thank Myong K. Jeong and Longin Latecki for providing the code of WDTW and OSB, respectively.

References

[1] G. Al-Naymat, S. Chawla, J. Taheri, SparseDTW: A novel approach to speed up dynamic time warping, Conferences in Research and Practice in Information Technology Series 101 (2007) (2009) 117–127.
[2] T. Albrecht, Dynamic Time Warping (DTW) (2009) 69–85.
[3] L. Benedikt, V. Kajic, D. Cosker, P.L. Rosin, D. Marshall, Facial Dynamics in Biometric Identification, in: Proceedings of the British Machine Vision Conference, 2008.
[4] H. Chen, Robust Text Detection In Natural Images With Edge-Enhanced Maximally Stable Extremal Regions, in: 18th IEEE International Conference on Image Processing (ICIP), 2011, pp. 2609–2612.
[5] A. Choudhary, R. Rishi, S. Ahlawat, A New Character Segmentation Approach for Off-Line Cursive Handwritten Words, Procedia Comput. Sci. 17 (2013) 88–95.
[6] S. Chu, E. Keogh, D. Hart, Iterative Deepening Dynamic Time Warping for Time Series, in: Proc 2nd SIAM International Conference on Data Mining, 2002, pp. 195–212.

[7] M.J.P. Eamonn J. Keogh, Scaling up dynamic time warping for datamining applications, KDD (2000) 285–289.
[8] A. Fischer, A. Keller, V. Frinken, H. Bunke, Lexicon-free handwritten word spotting using character HMMs, Pattern Recognit. Lett. 33 (7) (2012) 934–942.
[9] V. Frinken, A. Fischer, R. Manmatha, H. Bunke, A novel word spotting method based on recurrent neural networks., IEEE TPAMI 34 (2) (2012) 211–224.
[10] B. Gatos, G. Louloudis, T. Causer, K. Grint, V. Romero, J.A. Sánchez, A.H. Toselli, E. Vidal, Ground-Truth production in the tranScriptorium project, in: 11th IAPR International Workshop on Document Analysis Systems (DAS), 2014, pp. 237–241.
[11] T. Górecki, Using derivatives in a longest common subsequence dissimilarity measure for time series classification, Pattern Recognit. Lett. 45 (2014) 99–105.
[12] T. Górecki, M. Luczak, Non-isometric transforms in time series classification using DTW, Knowl. Based Syst. 61 (2014) 98–108.
[13] T. Górecki, M. Łuczak, Multivariate time series classification with parametric derivative dynamic time warping, Expert Syst. Appl. 42 (5) (2015) 2305–2312.
[14] C.V. Jawahar, M. Meshesha, A. Balasubramanian, Searching in document images., ICVGIP (2004) 622–627.
[15] Y.S. Jeong, M.K. Jeong, O.A. Omitaomu, Weighted dynamic time warping for time series classification, in: Pattern Recognition, 44, Elsevier, 2011, pp. 2231–2240.
[16] E. Keogh, C.A. Ratanamahatana, Exact indexing of dynamic time warping, Knowl. Inf. Syst. 7 (3) (2005) 358–386.
[17] K. Khurshid, C. Faure, N. Vincent, Word spotting in historical printed documents using shape and sequence comparisons, Pattern Recognition 45 (7) (2012) 2598–2609.
[18] M. Kulbacki, M. Kulbacki, J. Segen, J. Segen, A. Bak, A. Bak, Unsupervised Learning Motion Models Using Dynamic Time Warping, Systems Research (July 2015) (2002) 1–10.
[19] L.J. Latecki, S. Koknar-tezel, Q. Wang, V. Megalooikonomou, Sequence Matching Capable of Excluding Outliers, in: Int. Conf. on Knowledge Discovery and Data Mining (KDD), 2007a.
[20] L.J. Latecki, V. Megalooikonomou, Q. Wang, D. Yu, An elastic partial shape matching technique, Pattern Recognition 40 (11) (2007b) 3069–3080.
[21] L.J. Latecki, Q. Wang, S. Koknar-Tezel, V. Megalooikonomou, Optimal subsequence bijection, ICDM (2007c) 565–570.
[22] Y. Leydier, F. Lebourgeois, H. Emptoz, Text search for medieval manuscript images, Pattern Recognition 40 (12) (2007) 3552–3567.
[23] H. Li, L. Yang, Accurate and Fast Dynamic Time Warping, Advanced Data Mining and Applications (2013) 133–144.
[24] J. Listgarten, R.M. Neal, S.T. Roweis, A. Emili, Multiple alignment of continuous time series, Adv. Neural Inf. Process. Syst. 17 (17) (2005) 817–824.
[25] H. Manmatha, R. Chengfeng, E. Riseman, Word spotting: a new approach to indexing handwriting, in: CVPR, IEEE Comput. Soc. Press, 1996, pp. 631–637.
[26] M. Meshesha, C.V. Jawahar, Matching word images for content-based retrieval from printed document images, International Journal of Document Analysis and Recognition (IJDAR) 11 (1) (2008) 29–38.
[27] T. Mondal, N. Ragot, J.-Y. Ramel, U. Pal, Performance Evaluation of DTW and its 1065 Variants for Word Spotting in Degraded Documents, in: Proceedings of the 13th International Conference on Document Analysis and Recognition (ICDAR '15). IEEE Computer Society Washington, DC, USA, 2015, 1141-1145.
[28] T. Mondal, N. Ragot, J.Y. Ramel, U. Pal, Flexible Sequence Matching technique: An effective learning-free approach for word spotting, Pattern Recognition 60 (2016) 596–612.
[29] M. Munich, P. Perona, Continuous dynamic time warping for translation-invariant curve alignment with applications to signature verification, ICCV 1 (1999) 108–115.
[30] R. Oka, Spotting method for classification of real world data, The Computer Journal 41 (8) (1998) 1–6.
[31] C. Ratanamahatana, E. Keogh, Everything you know about dynamic time warping is wrong, Third Workshop on Mining Temporal and Sequential Data (2004) 22–25.
[32] T.M. Rath, R. Manmatha, Word spotting for historical documents, International Journal of Document Analysis and Recognition (IJDAR) 9 (2-4) (2006) 139–152.
[33] J.a. Rodríguez-Serrano, F. Perronnin, Handwritten word-spotting using hidden Markov models and universal vocabularies, Pattern Recognit. 42 (9) (2009) 2106–2116.
[34] H. Sakoe, S. Chiba, Dynamic programming algorithm optimization for spoken word recognition, IEEE Transactions on Acoustics, Speech, and Signal Processing 26 (1) (1978) 43–49.
[35] S. Salvador, P. Chan, FastDTW : Toward Accurate Dynamic Time Warping in Linear Time and Space, Intell. Data Anal. 11 (5) (2007) 561–580.
[36] K. Terasawa, Y. Tanaka, Slit Style HOG Feature for Document Image Word Spotting, in: Proceedings of the 2009 10th International Conference on Document Analysis and Recognition (ICDAR '09), IEEE Computer Society, Washington, DC, USA, 2009, pp. 116–120.
[37] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, E.J. Keogh, Indexing multi-dimensional time-series with support for multiple distance measures, in: KDD, ACM Press, New York, USA, 2003, pp. 216–225.
[38] A. Zifan, S. Saberi, M.H. Moradi, F. Towhidkhah, Automated ECG Segmentation Using Piecewise Derivative Dynamic Time Warping, International Journal of Medical, Health, Pharmaceutical and Biomedical Engineering 1 (3) (2007) 764–768.

Tanmoy Mondal: received B.Tech. degree in information technology from West Bengal University of Technology, Kolkata (India), in 2007 and the M.Tech. degree in mechatronics & robotics from Bengal Engineering and Science University, Kolkata (India) in 2009. Before joining as a PhD student at Poly-Tech Tours (France) in 2012, he worked at several industries and premier R&D centers as a researcher. After completing his PhD from Laboratoire d'Informatique, Poly-Tech, Tours (France) in 2015. Currently, he is doing Post-Doc at INSA, Lyon, France. His research interests include pattern recognition, image processing and analysis, and computer vision. His current research is mainly related to time series matching techniques and document image processing.

Nicolas Ragot: received his Ph.D. degree in computer science in 2003 from IRISA lab, Rennes University (France). Since 2005, he joined the Computer Science Lab (LI EA 6300) in the RFAI group of Université François-Rabelais, Tours (France), where he is an assistant professor at Poly-Tech Tours (French engineering school). His main research area is Pattern Recognition applied to Document Analysis. During the past 10 years, he worked mainly on online signature recognition, robust and adaptive OCR systems based on HMM, OCR control and defects detection (with French National Library-BnF). More recently he and Indian Statistical Institute-Kolkata received a 3 years grant from IFCPAR for project collaboration on robust and multilingual word spotting. He and his group were also involved in several National projects funded by government (ANR NAVIDOMAS, DIGIDOC etc.) as well as companies (ATOS Worldline, Nexter). His group has also received (during 2 years) Google Digital Humanities award to work on interactive layout analysis and the use of pattern redundancy for transcription and retrieval of old printed books.

Jean-Yves Ramel: received his Ph.D. in Computer Science (1996) from the RFV/LIRIS Laboratory in Lyon (France). From 1998 to 2002, he was working in the field of Man-Machine Interaction at INSA Lyon. Since 2002, he is working in the field of Pattern Recognition and Image Analysis at the Computer Sciences Laboratory (LI) of Tours (RFAI team) at Poly-Tech Tours (France). Since September 2007, he is Professor at the LI laboratory in the RFAI group.

Umapada Pal: received his Ph.D. in 1997 from Indian Statistical Institute. He did his Post Doctoral research at INRIA (Institut National de Recherche en Informatique et en Automatique), France. From January 1997, he is a Faculty member of Computer Vision and Pattern Recognition Unit of the Indian Statistical Institute, Kolkata and at present he is a Professor. His fields of research interest include Digital Document Processing, Optical Character Recognition, Biometrics, Word spotting etc. He has published 263 research papers in various international journals, conference proceedings and edited volumes. Because of his significant impact in the Document Analysis research, in 2003 he received ICDAR Outstanding Young Researcher Award from International Association for Pattern Recognition (IAPR). In 2008, 2011 and 2012, Dr. Pal received Visiting fellowship from Spain, France and Australia government, respectively. Dr. Pal has been serving as General/Program/Organizing Chair of many conferences including International Conference on Document Analysis and Recognition (ICDAR), International Conference on Frontiers of Handwritten Recognition (ICFHR), International Workshop on Document Analysis and Systems (DAS), Asian Conference on Pattern recognition (ACPR) etc. Also he has served as a program committee member of more than 50 international events. He has many international research collaborations and supervising Ph.D. students of many foreign universities. He is an associate Editor of the journal of ACM Transactions of Asian Language Information Processing (ACM-TALIP), Pattern recognition Letters (PRL), Electronic Letters on Computer Vision and Image Analysis (ELCVIA) etc. He has also served as a guest editor of several special issues. He is a Fellow of IAPR (International Association of Pattern Recognition).