Pattern Recognition 60 (2016) 596–612

Contents lists available at ScienceDirect

Pattern Recognition

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

Flexible Sequence Matching technique: An effective learning-free approach for word spotting

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

[a] Université François Rabelais, Laboratoire d'Informatique (LI EA 6300), Tours, France
[b] Computer Vision and Pattern Recognition Unit, Indian Statistical Institute, Kolkata, India

ARTICLE INFO

Article history:
Received 1 September 2015
Received in revised form
14 April 2016
Accepted 2 May 2016
Available online 24 May 2016

Keywords:
Flexible Sequence Matching (FSM)
Dynamic Time Warping (DTW)
Minimal Variance Matching (MVM)
Subsequence DTW (SSDTW)
Continuous Dynamic Programming (CDP)
Word spotting
Historical documents
Handwritten documents
Printed documents
George Washington dataset

ABSTRACT

In this paper, a robust method is presented to perform word spotting in degraded handwritten and printed document images. A new sequence matching technique, called the Flexible Sequence Matching (FSM) algorithm, is introduced for this word spotting task. The FSM algorithm was specially designed to incorporate crucial characteristics of other sequence matching algorithms (especially Dynamic Time Warping (DTW), Subsequence DTW (SSDTW), Minimal Variance Matching (MVM) and Continuous Dynamic Programming (CDP)). Along with the characteristics of multiple matching (many-to-one and one-to-many), FSM is strongly capable of skipping existing outliers or noisy elements, regardless of their positions in the target signal. More precisely, in the domain of word spotting, FSM has the ability to retrieve complete words or words that contain only a part of the query. Furthermore, due to its adaptable skipping capability, FSM is less sensitive to local variation in the spelling of words and to local degradation effects within the word image. The multiple matching capability (many-to-one, one-to-many) of FSM helps it addressing the stretching effects of query and/or target images. Moreover, FSM is designed in such a way that with little modification, its architecture can be changed into the architecture of DTW, MVM, and SSDTW and to CDP-like techniques. To illustrate these possibilities for FSM applied to specific cases of word spotting, such as incorrect word segmentation and word-level local variations, we performed experiments on historical handwritten documents and also on historical printed document images. To demonstrate the capabilities of sub-sequence matching, of noise skipping, as well as the ability to work in a multilingual paradigm with local spelling variations, we have considered properly segmented lines of historical handwritten documents in different languages and improperly as well as properly segmented words in printed and handwritten historical documents. From the comparative experimental results shown in this paper, it can be clearly seen that FSM can be equivalent or better than most DTW-based word spotting techniques in the literature while providing at the same time more meaningful correspondences between elements.

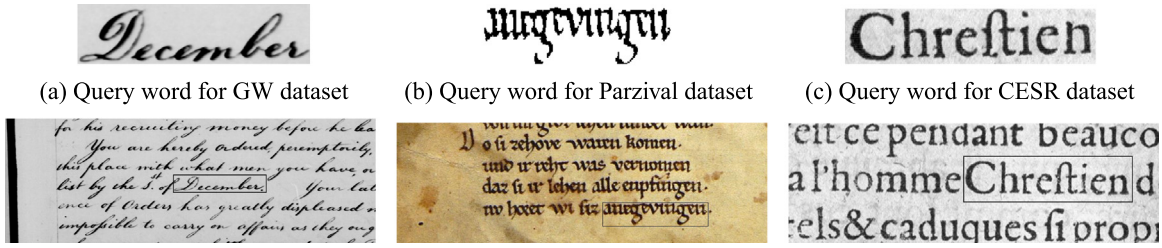
1. Introduction

Today's world of high quality document digitization has provided a stirring alternative to preserving precious ancient manuscripts. It has provided easy, hassle-free access of these ancient manuscripts for historians and researchers. Retrieving information from these knowledge resources is useful for interpreting and understanding history in various domains and for knowing our cultural as well as societal heritage. However, digitization alone cannot be very helpful until these collections of manuscripts can be indexed and made searchable. The performance of the available OCR engines highly dependent on the burdensome process of learning. Moreover, the writing and font style variability, linguistics and script dependencies and poor document quality caused by high degradation effects are the bottlenecks of such systems. The process of manual or semi-automatic transcription of the entire text of handwritten or printed documents for searching any specific word is a tedious and costly job. For this reason research has been emphasized on word spotting. This technique can be defined as the: "*localization of words of interest in the dataset without actually interpreting the content*" [1], and the result of such a search could look like the result shown in Fig. 1 (without transcription). These figures (Fig. 1) demonstrates a layman's view of the word spotting outcome of the system.[1]

* Corresponding authors.
E-mail addresses: tanmoy.mondal@etu.univ-tours.fr (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] For a detailed description of the datasets, please see the experimental evaluation Section 4. For a detailed description of the Parzival dataset, please see: ⟨http://www.iam.unibe.ch/fki/databases/iam-historical-document-database/parzival-database⟩.

http://dx.doi.org/10.1016/j.patcog.2016.05.011
0031-3203/



(a) Query word for GW dataset (b) Query word for Parzival dataset (c) Query word for CESR dataset

(d) Sample page of GW dataset (e) Sample page of Parzival dataset (f) Sample page of CESR dataset

Fig. 1. Example of word spotting outputs (marked by rectangular boxes in (d), (e), (f)) corresponding to queries ((a), (b), (c)) from document images extracted from 3 different datasets (see Section 4). The spotted query words on the complete document page are marked by a rectangular box. (a) Query word for GW dataset. (b) Query word for Parzival dataset. (c) Query word for CESR dataset. (d) Sample page of GW dataset. (e) Sample page of Parzival dataset. (f) Sample page of CESR dataset.

A popular way to categorize word spotting techniques is to consider those that are based on *query-by-example* and those that are based on *query-by-string*. In the former category, a region of a document is defined by the user, and the system should return all of the regions that contain the same text region, that is the same as the region defined by the user. These are often achieved by learning-free, image-matching-based approaches. For approaches that belong to the *query-by-string* category, queries of arbitrary character combinations can be searched. These approaches require a model for every character. Consequently, they are often achieved by learning-based approaches, such as HMM [2,3] or a Bidirectional Long Short-Term Memory (BLSTM) neural network [4]. These approaches allow us to obtain very good performance when the learning set is representative of the writing/font styles that are found in the document to be searched. The well-known drawback of learning-based approaches is the requirement of a set (most often enormous) of transcribed text line images for training, which could be costly to obtain for some of the historical datasets. Only very few approaches appear to be able to work with a low level of training data [5,6]. Moreover, the training (transcription of the learning set and learning of models) could have to be re-performed for new documents, depending on the variability of the writing/font styles. Thus, if neither the language nor the alphabet of a historical document are known or if creating a new learning set and retraining the system is necessary but not possible, a learning-free approach to word spotting might be the only available option. Consequently, a fair comparison between the two approaches is difficult to perform without including these criteria and we decided in this study to focus on learning-free approaches. These approaches can be further categorized depending on the level of segmentation.

1.1. Segmentation-based word spotting methods

The concept of word spotting as the task of detecting words in document images without actually understanding or transcribing the content, was initially the subject of experimentation by Manmatha et al. [1]. This approach relies on the segmentation of full document images into word images. A general and highly applicable approach for comparing word images is to represent them by a sequence of features, which are extracted by using a sliding window. These word images can be thought of as a 2D signal, which can be matched using dynamic programming [1,7] based approaches. Some methods that were oriented toward bitwise comparison of images were also investigated [8], as well as holistic approaches that describe a full image of words [9,10]. An approach based on low-dimensional, fixed-length representations of word images, which is fast to compute and fast to compare, is proposed in [11]. Based on the topological and morphological information of handwriting, a skeleton based graph matching technique is used in [12], for performing word spotting in handwritten historical documents. There have also been some attempts to spot words on segmented lines to avoid the problems of word segmentation. Indeed, depending on the document quality, line segmentation could be comparatively easier than word segmentation. The partial sequence matching property of CDP [13] is one possibility. Using over segmentation is also an alternative as in [14], where the comparison of sequences of primitives obtained by segmentation and clustering (corresponding to similar characters or pieces of characters) is investigated.

The necessity of proper word segmentation (or line segmentation in some cases) and the high computational complexity of matching are critical bottlenecks of most of the techniques in this category. Moreover, these techniques are prone to the usual degradation noise that is found in historical document images. Most of them cannot spot out-of-vocabulary words.

1.2. Segmentation free word spotting methods

In [15], the authors proposed another type of matching technique based on differential features to match only the informative parts of the words, which are detected by patches. A common approach for segmentation-free word spotting is to consider the task as an image retrieval task for an input shape, which represents the query image [16]. For example, a HOG descriptors based sliding window is used in [17] to locate the document regions that are the most similar to the query. In [18], by treating the query as a compact shape, a pixel-based dissimilarity is calculated between the query image and the full document page (using the Squared Sum Distance) for locating words. A heat kernel signature (HKS) based technique is proposed in [19]. By detecting SIFT based key points on the document pages and the query image, HKS descriptors are extracted from a local patch that is centered at the key points. Then, a method is proposed to locate the local zones that contain a sufficient number of matching key points that corresponds to the query image. Bag of visual words (BoVW) based approaches were also used to identify the zones of the image that share common characteristics with the query word. In [20], the Longest Weighted Profile based zone filtering technique is used from BoVW to identify the location of the query words in the document image. In [21], local patches powered by SIFT descriptors are described by a BoVW. By projecting the patch descriptors to a topic space with a latent semantic analysis technique and compressing the descriptors with a product quantization method, the approach can efficiently index the document information both in terms of memory and time. Overall, segmentation-free approaches can overcome the curse of the segmentation problems, but they have comparatively low accuracy (in comparison with segmentation-based and learning-based approaches) and a high computational burden, considering the full image

Table 1
Grouping of state of the art techniques of word spotting.

	Without learning	With learning
Segmentation free	[15–21]	
Line based	[13], [Proposed method]	[2–6]
Word based	[1,7–10,12], [Proposed method]	
Character based	[14]	

regions as an apparent candidate for matching.

The Table 1 summarizes several existing word spotting techniques, depending on the criteria mentioned above: learning or not; and level of segmentation. From this literature review, we noticed that there are still some important unresolved problems in this domain, e.g., word spotting independent of script and language variability, has not been properly addressed by the research community. Additionally, excluding learning based approaches, there are few research studies that could handle noise and degradation effects in historical documents [22,3]. In most languages, there are several variations or derivatives of words that could be interesting for the user. For example, the French word *cheval (horse)* can have derivatives such as "*chevalerie*", "*chevaux*", "*chevalier*". In old French, lexical variations also exist e.g., "*cheual*", "*chevaus*", and these variations do not change the meaning of the word; being able to retrieve these derivatives of the word that is being searched could be very useful. However, very little work is available in this direction as well as in the direction of skipping prefix and suffix of segmented words [13,7].

For this reason, in this paper, we propose a robust learning-free word spotting method by introducing a novel sequence matching technique, called Flexible Sequence Matching (FSM). The proposed FSM technique is designed to overcome the bottlenecks of the other sequence matching techniques that have been applied in the domain of word spotting, e.g., DTW [1], CDP [13], and some modified version of classical DTW [7]. The proposed algorithm is capable of handling (to some extent) the local degradations and noise that is present in the image by skipping it, if necessary. To an extent, it can also handle lexical variations or derivatives by skipping unnecessary portions such as prefixes or suffixes. FSM is flexible with regard to the word segmentation problem and does not truly depend on good segmentation: it can work on pieces of lines and/or on improperly segmented words (see Table 1). Finally, the approach can handle spelling (local) variations and word derivatives. These properties allow the algorithm to find meaningful correspondences between elements of query and targets, which is not often the case for classical sequence matching techniques. Finally, the architecture of FSM is designed in such a generalized manner that it can be easily modified into the architecture of other sequence matching techniques, e.g., Minimal Variance Matching (MVM) [23], DTW [1], Subsequence DTW (SSDTW) [24], and Continuous Dynamic Programming (CDP) [13]. To show the usefulness of FSM in the word spotting domain, a comparison of FSM with other similar sequence matching techniques such as DTW, SSDTW, MVM, Optimal subsequence bijection (OSB) [25], and CDP is performed.

The remainder of this paper is organized as follows. In Section 2, a comparative discussion of state-of-the-art sequence matching techniques is given. The proposed word spotting framework, along with descriptions of the used features are briefly mentioned in Section 4.1. The core idea of the paper, the *Flexible Sequence Matching* technique, is explained in Section 3, including it's theoretical description, pseudo-code and generalization properties (see Appendix E). The experimental evaluation is described in Section 4, and conclusions and future work are described in Section 5.

2. Bird's eye view of sequence matching techniques

Many studies have been published in the literature [1,7], following an architecture for word image matching that is similar to the architecture that is based on classical DTW [24] (see also Section 4.1). The main idea of DTW is to calculate the distance between two time series by summing up the distances of their corresponding elements. DTW yields an optimal (order preserving) relationship $\mathbb{R}$ of all of the elements of sequence $x = \{x_1, x_2, x_3 \ldots. x_p\}$ to all of the elements of sequence $y = \{y_1, y_2, y_3 \ldots. y_q\}$. Dynamic programming is used to find the best corresponding elements (see Appendix B for more details). It has been shown in the literature that the DTW distance is superior to the Euclidean distance [26]. This proposition has also been proven for word spotting [27]. Nevertheless, there are some limitations to classical DTW, especially that each element of $x_{1..p}$ must correspond to some $y_{1...q}$, and vice versa (one-to-one, one-to-many and many-to-one matchings). This hard constraint often forces DTW to perform a correspondence with noisy elements, which could disturb the matching and also increase the final distance value. Hence, the final ranking of the elements for the task of word spotting can be disturbed. A critical problem arises, when sequence x corresponds to only a part y' of the sequence y: DTW cannot ignore the elements that do not belong to y'. To perform partial matching of the sequences, some research has been introduced in the literature, e.g., SSDTW [24], which is designed to find a continuous subsequence within a longer sequence that can optimally fit the shorter query sequence. Another DTW-based modification, called as DTW with corresponding window (DTW-CW) [23], can perform partial sequence matching, while using a sliding correspondence window of the same size as the query sequence and using DTW at each position. Although this approach can give better accuracy for subsequence matching, it is also very time consuming, due to it's architectural constraint. Moreover, deciding the threshold for sliding the corresponding window is highly data dependent and a cumbersome process. Continuous Dynamic Programming (CDP) [13] is another alternative to operate partial matching that is showing promising results. It is similar to DTW-CW but does not need to recompute DTW at each position. It is using a specific DP-path and the minimum of output function along target signal is giving candidate positions (see Appendix D). These partial matching algorithms have similar drawbacks to DTW, especially considering difficulty to find relevant correspondences because of inability to skip outliers inside matched part. Moreover correspondences between elements is not directly provided with output.

To handle noisy points in DTW, derivative DTW (DDTW) was proposed in [28]. This method is based on derivatives of feature elements, instead of the original feature sequence. An useful modification of the classical DTW technique is weighted DTW (WDTW) [29]. This approach penalizes points that have a higher phase difference between a reference point and a testing point, to prevent minimum distance distortions that are caused by outliers. In Weighted Derivative Dynamic Time Warping (WDDTW) [29], the penalization idea in WDTW is extended to DDTW to capture the benefits of both approaches. These above defined techniques, try to limit the effects of noisy elements in the sequence, but they are unable to completely skip the noisy elements. To address this restriction, some other sequence-matching algorithms were proposed.

To find an optimal correspondence between two sequences, a popular approach is the Longest Common Subsequence (LCSS) [30]. Given a query and a target sequence, LCSS determines their longest common subsequence, i.e., the sequence that best corresponds between the original sequences. The dissimilarity measure is based on the ratio between the length of the longest common subsequence and the length of the whole sequence. The elements of the subsequence do not need to be consecutive points, which could be a problem for word spotting. The order of the points is not rearranged, and some points can remain unmatched. To make LCSS efficient, one must also set the threshold that determines when the distances between corresponding points would be treated as being equal. Determining this threshold

value is a difficult task and it is highly data dependent [31].

To overcome this problem, Minimal Variance Matching (MVM) was proposed by Lateki et. al. [23]. MVM combines the strength of both DTW and LCSS, while overcoming their constraints. MVM calculates the sequence similarity directly based on the distances of corresponding elements. MVM also tries to find an optimal path including all the corresponding pairs but MVM is able to skip outliers in target during the matching process (see Appendix C). The notable difference between LCSS and MVM is that LCSS optimizes the length of the longest common subsequence, while MVM optimizes the sum of distances of corresponding elements (without any distance threshold). Moreover, LCSS can skip both query and target elements, whereas MVM can skip only the target sequence. Thus, MVM could be used only when the query sequence is smaller than the target sequence, and it is useful to find the query in a larger target sequence. For the case of word spotting, this case is the classical situation, if we consider that the query image is rightly selected and there is no disturbing noise present in the query image. However, when there are outliers in the query sequence and skipping those outliers from the query sequence is necessary, then the MVM properties remain limited. Finally, both algorithms (LCSS and MVM) are capable of performing one-to-one correspondences (many-to-one and one-to-many matchings are not allowed). Another algorithm, designed by Lateki et. al. [25], is Optimal Subsequence Bijection (OSB) [25]. The goal of OSB is to find subsequence x' from x and y' from y such that x' best matches y', by skipping some elements from x and y because both the sequences might contain some outlier elements. Thus, the properties of OSB is more similar to the properties of LCSS. To prevent skipping too many elements, the algorithm introduces a penalty for skipping.[2] Like MVM, OSB can perform only one-to-one correspondences, both of these algorithms cannot do many-to-one and one-to-many correspondences. But the noteworthy point is that OSB has a much higher computational complexity than MVM.

The Tables A1 and A2 in Appendix A, summarize the main properties of these different algorithms. We can see that none of them include at the same time the main properties that are necessary for word spotting, i.e., being able to perform multiple matches because signals (query and target images) could have local stretching/extension and because of varying writing/font styles; being able to operate partial matching by skipping irrelevant elements at the beginning and end of the target because of wrong segmentation; and being able to skip noisy elements inside the target because of lexical variations or degradations. For these reasons, we propose, in the following section, a novel sequence matching algorithm. This algorithm can successfully overcome most of the problems of the aforementioned individual sequence-matching algorithms and thus could be very useful for the word spotting domain as well as many other applications for which such flexibility is required.

3. Flexible sequence matching (FSM)

In this section, the complete mathematical structure of FSM is described. The main properties of FSM are summarized in Table A1 and A2. FSM creates a relation $\mathbb{R}$ from two finite sequences x (query) to y (target), of different lengths p and q: $x = (x_1, x_2, \ldots, x_p)$ and $y = (y_1, y_2, \ldots, y_q)$; $p \le q$.[3] The goal of the algorithm is to find $y'(y' \subset y)$ such that x best matches y'. The relationship $\mathbb{R}$, is performed on the set of indices $\{1, \ldots. p\} \times \{1, \ldots. q\}$, where, one-to-one, one-to-many and many-to-one mapping is possible. Then, based on the correspondence, a distance between the two sequences is computed.

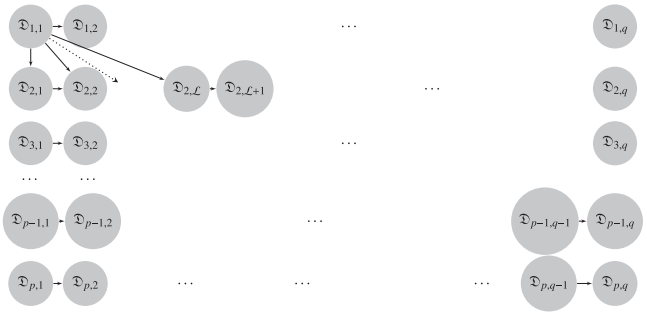


Fig. 2. DAG constructed using a distance matrix obtained from two sequences.

3.1. Theoretical description of FSM

The theoretical background of the FSM algorithm comes from DTW and MVM techniques. From DTW, we are mainly keeping its relational structure (one-to-one, many-to-one and one-to-many matching in order preserving manner). From MVM, we kept its algorithmic structure so that the participation of all the elements of y in the relation $\mathbb{R}$ could be relaxed. It means, if there are some noisy elements present in y then, they can be intelligently skipped by the algorithm. In the following, a stepwise description of FSM algorithm is illustrated. Note that, 1D time series signals are considered for ease of description but the extension to sequences of elements corresponding to vectors of same dimension, can be easily performed.

First, the difference matrix $\mathfrak{D}$ is calculated by using the Euclidean distance $\mathfrak{D}_{i,j} = \sqrt{(y_j - x_i)^2}$; $1 \le i \le p$; $1 \le j \le q$. According to [23], there is no restriction on using various distance measures and any of the distance measures can be considered. Here, $\mathfrak{D}$ can be used to generate a directed acyclic graph (DAG), denoted as $\mathbb{G}$, where each of the elements of $\mathfrak{D}$ can be considered to be a node. The links between each node and it's child nodes are obtained by solving a shortest path problem, i.e., by using the cost function $\mathcal{H}$ defined in (1), where $\mathfrak{D}_{u,k}$ represents the parent nodes, and $\mathfrak{D}_{i,j}$ represents the child nodes.[4]

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

$$\mathcal{L} = (k+1) + elasticity - |k-u|; \; elasticity = |q-p| \tag{2}$$

In (1), any node $\mathfrak{D}_{u,k}$ has the flexibility to be connected with another node $\mathfrak{D}_{i,j}$ in two possible ways. Here, (i) means that, a parent node can be connected with child nodes in the next row $(i = u+1)$, in the same column or at the right $(k \le j \le \mathcal{L})$, up to a given user-defined elastic limit $\mathcal{L}$, with respect to the position of parent node.[5] This elastic limit represents the number of jumps that could be performed inside the matched part of the target. By default, it is equal to the difference in size, between the query and target sequences (i.e. $|q-p|$) minus the number of jumps already used (i.e., the difference between the column and row indexes of the parent node: $|k-u|$).[6] The connection at the same column $(k=1)$ is mainly to ensure the many-to-one matching (link with the parent node and the child node in the same column). The condition (ii) says that the parent node can also be connected with the node next to it, in the same row. This particular connection, ensures the one-to-many matching facility. In this way, we can generate $\mathbb{G}$ (see Fig. 2).

Judging the intensity of the noise, that is present in the signal is a

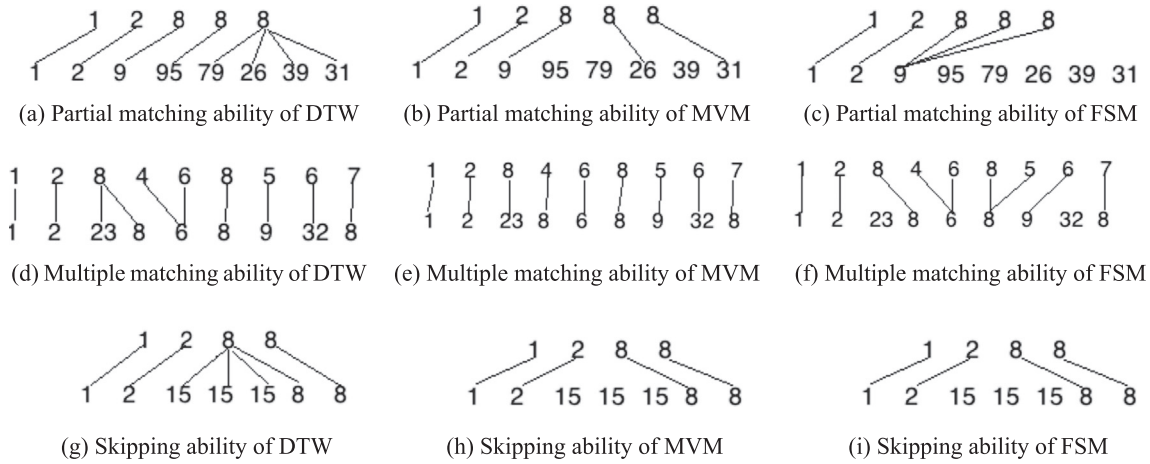
[2] Please note that, the MVM algorithm does not have any penalty for skipping.

[3] The length of the query sequence should always be less than or equal to the length of the target. For given application application, if this constraint is not satisfied, then we can consider the query as the target and vice versa, i.e. we simply reverse the order of the input arguments in the FSM function.

[4] Please note that the indexes of all of the matrix notations used in this paper, always start from 1.

[5] When $q=p$, the value of *elasticity* is taken to be 2 to maintain some skipping facility.

[6] This constraint is not mandatory but it limits the complexity of the algorithm.



(a) Partial matching ability of DTW (b) Partial matching ability of MVM (c) Partial matching ability of FSM

(d) Multiple matching ability of DTW (e) Multiple matching ability of MVM (f) Multiple matching ability of FSM

(g) Skipping ability of DTW (h) Skipping ability of MVM (i) Skipping ability of FSM

Fig. 3. Partial matching, many-to-one and one-to-many matching and noise skipping ability of DTW, MVM and FSM. (a) Partial matching ability of DTW. (b) Partial matching ability of MVM. (c) Partial matching ability of FSM. (d) Multiple matching ability of DTW. (e) Multiple matching ability of MVM. (f) Multiple matching ability of FSM. (g) Skipping ability of DTW. (h) Skipping ability of MVM. (i) Skipping ability of FSM.

difficult and puzzling problem. Indeed, deciding, whether a specific element should be considered to be an outliers (hence skipping it) or not is not straightforward. For example, in the case of LCSS, the user must define a threshold. In FSM, this type of threshold is not required, but the system cannot be allowed to skip outliers without any resistance. Otherwise, the process can frequently skip elements, instead of matching them with some acceptable dissimilarity cost. Thus, to limit skips, *skipCost* is introduced. This cost corresponds to a distance value that is higher than the cost that corresponds to an acceptable distance between corresponding elements. Due to the dependency of the *skipCost* value on the dataset, we estimate this value in the following way. We randomly choose two query sequences. For each, we randomly choose two corresponding targets that should match.[7] Then, the distance matrices between the chosen queries and targets are calculated. A vector $\mathfrak{M}_i$ is created to contain the average of the top m minimum values from each row i, $i = 1, \ldots, p$ of the corresponding matrices. The obtained vectors $\mathfrak{M}_i$ for each query and target pair are merged together and sorted in ascending order, giving $\mathfrak{M}^{merged}$. Then, only 90% of the elements of $\mathfrak{M}^{merged}$ are kept in $\mathfrak{M}^{merged'}$. These values correspond to the distances between the matching elements.[8] Finally, *skipCost* is computed based on the mean and standard deviation of $\mathfrak{M}^{merged'}$ (see (3)).

$$\forall\, i_{i=1,\ldots,p},\quad \mathfrak{M}_i = \text{mean}\left\{\text{top} - m\_\min_{j=1,\ldots,q}\left(\mathfrak{D}_{i,j}\right)\right\};\ \mathfrak{S}$$
$$= \text{skipCost} = \text{mean}\left(\mathfrak{M}_b^{merged'}\right) + 2 \times \text{std}\left(\mathfrak{M}_b^{merged'}\right) \tag{3}$$

To perform the matching, we must find the shortest path through $\mathbb{G}$ from a single initial node $\mathfrak{D}_{1,j}$, which leads to $\mathfrak{D}_{p,l}$; $l \in \{p, \ldots, q\}$. It is obtained due to a path cost matrix $\mathfrak{P}(i, j)$; $1 \le i \le p$; $1 \le j \le q$ and satisfies the following conditions:

(1) starts at the 1st row between column 1 and q; this starting point allows us to skip an unnecessary prefix.[9]
(2) ends at the last row p in a column $l \in \{p, \ldots, q\}$ to skip an unnecessary suffix.
(3) the shortest path between each pair of reachable nodes in $\mathbb{G}$ can be found from $\mathfrak{P}$.

The construction of $\mathfrak{P}(i, j)$ is given by (4a)–(4c). (4a) says that the 1st row of the distance matrix $\mathfrak{D}$ is copied to the path cost matrix $\mathfrak{P}$. Each cell that belongs to the i^{th} row ($i \ge 2$) is calculated by first choosing the possible parent nodes in a previous row $(i - 1)$ and in the columns k that range from $((i - 1) - elasticity)$ to $((i - 1) + elasticity)$ (see (4c)). Considering a parent node, its child nodes can belong only to the next row and in columns that range from the next column $(k + 1)$ until column $(k + 1) + elasticity - |k - (i - 1)|$. Other possible links are for many-to-one and one-to-many matchings: the vertical link (from node $(i - 1, j)$ to (i,j)) is responsible for many-to-one matching and the left link (from $(i, j - 1)$ to (i,j)) is responsible for one-to-many matching. Please note that in such cases, a small penalty ($\mathfrak{C} = mean(\mathfrak{M}_i^{merged'})$) is introduced to limit the numbers of multiple matchings. If for some application, multiple matching is not required, the last two terms of (4b) can be ignored.[10] The path cost $\mathfrak{P}(i, j)$ is updated only when there is a shorter path that comes from a parent node. The optimal structure condition guarantees that the returned matrix $\mathfrak{P}$, contains the cost of the shortest path that leads to every node under the previous constraints.

$$\mathfrak{P}(1, j) = \mathfrak{D}_{1,j} \text{ if } 1 \le j \le q \tag{4a}$$

$$\mathfrak{P}(i, j) = \begin{pmatrix} \min \begin{pmatrix} \{\mathfrak{P}(i-1, k) + \mathfrak{D}_{i,j} + (\mathfrak{S} \times (j - (k+1)))\} \\ \{\mathfrak{P}(i, j-1) + \mathfrak{C} + \mathfrak{D}_{i,j}\} \\ \{\mathfrak{P}(i-1, j) + \mathfrak{C} + \mathfrak{D}_{i,j}\} \end{pmatrix} \text{if } \mathfrak{L} \end{pmatrix} \tag{4b}$$

$$\mathfrak{L} = \begin{pmatrix} \{2 \le i \le p\} \\ \max[1, (i-1) - \text{elasticity}] \le k \le \min[q, (i-1) + \text{elasticity}] \\ k + 1 \le j \le \min(q, (k+1) + \text{elasticity} - \max(0, |k - (i-1)|)) \end{pmatrix} \tag{4c}$$

The distance between two comparable sequences can be obtained by obtaining the minimum value that is stored in the last row at the l^{th} column of the path cost matrix $\mathfrak{P}$, where $p \le l \le q$. To normalize the sequence dissimilarity value, we divide it by the number of corresponding pairs between the target and query sequence.

[7] Choosing such elements can be viewed as a calibration step that is not time consuming or difficult.

[8] The idea of choosing m minimum values is to have a cost that will not be null, when perfect matches are available between the query and target sequences. Here, $m=2$ is the minimum to consider, but experimentally, taking $m=5$ is less sensible and is in an acceptable range, with respect to the sizes of our sequences. It should be noted that this parameter value is not very important for the proposed algorithm.

[9] Please note that to allow substring matching, these skips at the beginning and end are not penalized, i.e., no skipCost is added

[10] One should note also that the elements of the path can be weighted as in DTW's variants. This was done in our implementation (see section below) but is not mentionned here for clarity.

3.2. Description of the Pseudo Code

As an output, the algorithm provides the path costs matrix $\mathfrak{P}$ along with two other matrices, ($\mathcal{R}$ and C), which are used to backtrack the shortest path, giving the closest correspondences between the query and target elements. The matrix $\mathcal{R}$ keeps track of the rows and C keeps track of the columns, and the vector $\mathfrak{W}$ merges these indexes to obtain the final path. The overall process is given by Algorithm 1. All of the cells of $\mathfrak{P}$ are initialized with infinity. Next, the first row of $\mathfrak{P}$ is obtained by (4a) (lines 2–3). The other cells are calculated (refer to lines 4–25) by iterating i over each row, while k iterates over the parent nodes in the row above $(i-1)$, and j keeps track of the child nodes for a given parent node. For each cell of each row, first the range of the parent nodes is calculated (lines 5 – 10). When $i=2$, the parent nodes can be at anywhere in the first line/row, to allow the skipping of the irrelevant part (at no cost) at the beginning of the target. Next, for each parent node from L (left) to R (right), the possible connections with child nodes are calculated by j, which is iterated between k and D according to (4c). Similar to DTW, FSM can perform multiple matches: many consecutive elements of the query to one element of the target (line 14) and one-to-many (lines 23–25). It can also skip some target elements. Recall here that the jump cost ($\mathcal{J}$) is based on the number of skips that are already taken (term $(j-(k+1))$ in line 19) and the skip cost ($\mathfrak{S}$). Notice also that this jumping facility is here weighted in the similar way as it is in CDP (2/3 of the cost for reaching parent node and 1/3 for matched element $\mathfrak{D}(i,j)$), since preliminary experiments have shown better results in such case. In contrast, when the parent node at $(i-1,k)$ is connected with the node at $(i,k+1)$ (diagonal link), which means that two consecutive elements of the query are matched with two consecutive elements of the target sequence, then no cost is added. The calculation of the warping path $\mathfrak{W}$ is given in lines 26–30. Initially, the index of the column, which ranges from p to q, and contains a minimum value in the last row of the path cost matrix is found. Then, back-tracking is performed through the while loop to obtain $\mathfrak{W}=\mathfrak{W}_1,\mathfrak{W}_2,\mathfrak{W}_3,\ldots,\mathfrak{W}_k,\ldots\mathfrak{W}_K$; $max(q,p)\le K\le q+p-1$, where $\mathfrak{W}_k$ contains an ordered set of indices of the corresponding elements. Finally, the distance $\bar{\mathfrak{d}}$ (see line 32 of Algorithm 1) is outputted.

Algorithm 1. Flexible Sequence Matching.

Input: $x_{1\ldots p}, y_{1\ldots q}, \mathfrak{D}_{pq}(double\times double), elasticity(=|q-p|), \mathfrak{S}(double), \mathfrak{C}(double)$
Output: $\mathfrak{P}_{pq}(double\times double), \mathfrak{W}(vector<(int,int)>), \bar{\mathfrak{d}}(double)$

1 $\mathfrak{P}\leftarrow\infty$ ▷ Initialize all the cells of $\mathfrak{P}$ matrix by ∞
2 **for** $j\leftarrow 1$ ***to*** q **do**
3 $\mathfrak{P}(1,j)\leftarrow\mathfrak{D}(1,j)$ ▷ Fill the 1^{st} row of $\mathfrak{P}$ with the 1^{st} row values at $\mathfrak{D}$ matrix
4 **for** $i\leftarrow 2$ ***to*** p **do**
5 **if** $i=2$ **then**
6 $R\leftarrow q$ ▷ Complete flexibility is given for choosing the 1^{st} node
7 $L\leftarrow 1$
8 **else**
9 $R\leftarrow min(q,(i-1)+elasticity)$
10 $L\leftarrow max(1,(i-1)-elasticity)$
11 **for** $k\leftarrow L$ ***to*** R **do**
12 $D\leftarrow((k+1)+elasticity)-max(0,\{k-(i-1)\})$
13 **for** $j\leftarrow k$ ***to*** D **do**
14 **if** $j=k$ **then**
15 $\mathscr{J}\leftarrow\mathfrak{C};\mathscr{W}\leftarrow 1$ ▷ Penalty and weights for vertical links
16 **else if** $j=k+1$ **then**
17 $\mathscr{J}\leftarrow 0;\mathscr{W}\leftarrow 1$ ▷ No penalty/weights for diagonal links
18 **else**
19 $\mathscr{J}\leftarrow 2/3\times\{j-(k+1)\}\times\mathfrak{S};\mathscr{W}\leftarrow 1/3\times\{j-(k+1)\};$ ▷ Penalty proportional to number of jumps
20 **if** $\mathfrak{P}(i,j)>(\mathfrak{P}(i-1,k)+\mathscr{W}\times\mathfrak{D}(i,j)+\mathscr{J})$ **then**
21 $\mathfrak{P}(i,j)\leftarrow(\mathfrak{P}(i-1,k)+\mathscr{W}\times\mathfrak{D}(i,j)+\mathscr{J})$ ▷ Link between two rows
22 $\mathscr{R}(i,j)\leftarrow i-1;\mathscr{C}(i,j)\leftarrow k$
23 **if** $\mathfrak{P}(i,j)>\mathfrak{P}(i,j-1)+\mathfrak{C}+\mathfrak{D}(i,j)$ **then**
24 $\mathfrak{P}(i,j)\leftarrow\mathfrak{P}(i,j-1)+\mathfrak{C}+\mathfrak{D}(i,j)$ ▷ Link from left node
25 $\mathscr{R}(i,j)\leftarrow i;\mathscr{C}(i,j)\leftarrow j-1$
26 $\mathfrak{I}=\underset{p\le t\le q}{\operatorname{argmin}}\,\mathfrak{P}(p,t)$ ▷ Column index (from index p to q), of last row with the minimum value
27 $\mathfrak{r}=p;\mathfrak{c}=\mathfrak{I};cnt=1$
28 **while** $((\mathfrak{r}\ge 1)\&(\mathfrak{c}\ge 1))$ **do**
29 $\mathfrak{W}_{cnt}=(\mathfrak{r},\mathfrak{c})$ ▷ Storing the cell indexes in the array for getting the warping path
30 $\mathfrak{t}=\mathscr{C}(\mathfrak{r},\mathfrak{c});\mathfrak{r}=\mathscr{R}(\mathfrak{r},\mathfrak{c});\mathfrak{c}=\mathfrak{t};cnt++$
31 $reverse(\mathfrak{W})$ ▷ The warping path vector is reversed to get elements in right order
32 i) $\bar{\mathfrak{d}}=\frac{\mathfrak{P}_{p,\mathfrak{I}}}{|\mathfrak{W}|}$; where $|\mathfrak{W}|=sizeof(\mathfrak{W})$ ii) $\bar{\mathfrak{d}}=\frac{\mathfrak{P}_{p,\mathfrak{I}}}{|\mathfrak{W}|+\{\mathfrak{W}(cnt-1,1)-1\}+\{q-\mathfrak{W}(1,1)\}}$ ▷ Depending on the requirement, the normalized final distance ($\bar{\mathfrak{d}}$) can obtained in two different ways.

3.3. Properties of FSM

The warping path of FSM follows these constraints:

i. **Boundary condition**: FSM does not maintain the boundary condition of classical DTW and can perform partial sequence matching. The warping path can start from any specific column in the last row, ranging from the index p to q, i.e., $\mathfrak{W}_K = (p, t)$; $p \le t \le q$ and the warping path can end at any column, in the first row, i.e., $\mathfrak{W}_1 = (1, u)$; $1 \le u \le q$.
ii. **Continuity condition**: The continuity condition of classical DTW is also not exactly maintained by FSM: $\mathfrak{W}_k = (a, b)$ is followed by $\mathfrak{W}_{k+1} = (a', b')$; where $(a' - a) \in [0, 1]$ and $(b' - b) \in [0, 1, \ldots |p - q| + 1]$. Due to this characteristic, FSM can jump outliers from the target signal.
iii. **Monotonicity condition**: The monotonicity condition restricts the warping path from going back in time: $\mathfrak{W}_k = (a, b)$, $\mathfrak{W}_{k+1} = (a', b')$ is constrained by $(a' - a) \ge 0$ and $(b' - b) \ge 0$.

Additionally, it is very easy for the presented algorithm to be changed in such a way that it can behave as other algorithms such as DTW, MVM and SSDTW (details can be found in Appendix E).

3.4. Examples of matching with FSM

Using some toy examples, the behavior of FSM, compared to DTW and MVM is demonstrated in Fig. 3. Fig. 3(a), (b) and (c) is showing the results of matching the sequences: $Q = [1, 2, 8, 8, 8]$ and $T = [1, 2, 9, 95, 79, 26, 39, 31]$. It can be seen that for the case of DTW, the 1st, 2nd and 3rd elements of the query and target are correctly matched. However, the 4th element ('8') of the query is forced to be matched with the 4th element ('95') of the target. The last element of the query is also forced to be matched with the other remaining elements of the target (many-to-one matching), which significantly increases the final distance. The MVM algorithm can improve the matching process by skipping the outliers elements '95' and '79'. However, it's inability to have many-to-one matching and it's restriction on the numbers of matched elements (which should be equal to the number of elements in the query), compel the algorithm to match the 4th and 5th elements of Q with the 6th and 8th elements of T. In contrast, it can be easily visible that FSM can overcome these restrictions and provide correct correspondences. In such a case, small costs will minutely penalize the final distance for the one-to-many matchings.

In Fig. 3(d)–(f), the same trends can be seen. However, here we can see that even if the lengths of the target and query signals are same, the *elasticity* for the FSM algorithm is taken to be 2, which allows it to skip outliers inside the target. DTW cannot do that, and neither can MVM, in such a case. Fig. 3(g)–(i) illustrates the same possibilities when the query is smaller than the target. In such a case, MVM behaves the same as FSM. Again, DTW is forced to match all of the elements.

3.5. Time complexity of FSM

The worst case complexity of FSM is calculated by computing total number of possible parent nodes at each row of $\mathfrak{P}$ matrix. In worst case, we would have a subset of at most $\{(2 \times |q - p|) + 1\}$ parent nodes (see line 9, 10 of Algorithm 1) at each row of $\mathfrak{P}(i, j)(1 \le i \le p;\ 1 \le j \le q)$ matrix; which indeed can be represented as the vertices of DAG $\mathbb{G}$. Then for each of the parent nodes in row i is linked to at most $\mathcal{T}$ child nodes. Since there are total p rows, the algorithm complexity could be defined by: $[p \times \{(2 \times |q - p|) + 1\} \times \mathcal{T}]$.

$$\mathcal{T} = \left[\left\{(j+1) + |q-p|\right\} - (j+1)\right] + 1 - (j-i) + 1 + 1 = |q-p| + (j-i) + 3 \approx |q-p| + 3 \quad (5)$$

In Eq (5), the term $(j - i)$ can be ignored (it reduces the complexity) for performing worst case analysis. The term 3 comes from adding the diagonal link, link with the right child node on the same line as parent node and the child node just below parent node (see line 14 of Algorithm 1). The worst case complexity of FSM is $\Theta([|q - p| + 3] \times [p \times \{(2 \times |q - p|) + 1\}]) = \Theta(2.\ p \times ((|q - p| + 2)^2 - 1)) \approx \Theta(2.\ p \times (|q - p| + 2)^2)$. This assumption is also true : $\Theta((2.\ |q - p|^2).\ p)(\Theta(2.\ q^2.\ p))$. Furthermore the complexity can be reduced to linear $(\Theta((2.\ |q - p|^2).\ p) \approx \Theta(p))$ when $q \approx p$; i.e. matching whole target)). The time complexity of FSM can further be reduced by introducing an admissible lower bound. However, in this work we focus on demonstrating the utility of FSM; we will address speedup and index-ability of FSM in future work.

4. Experimental evaluation

The experiments of the word spotting application was performed on: *i) correctly segmented words*, *ii) incorrectly segmented words, i.e. pseudo-words*, *iii) correctly segmented lines*. Indeed, depending on the characteristics of the documents, it can be comparatively easier to perform word segmentation or line segmentation. For example, if inter-word gaps are not large enough or are variable (which is often the case in old historical documents), word segmentation can be very difficult. However, for handwritten documents with curved and touching lines, segmenting pieces of lines could be easier. The level of difficulty is highly writer and language dependent.[11] Thus, due to the pros and cons of the two segmentation techniques, we have performed experimental evaluations on all the aforementioned cases.

4.1. Word spotting framework

In this section, our complete word spotting framework is briefly explained in a stepwise manner. Because the main scope of this paper is a word image "*matching technique*", we have not provided the details about all of the steps of the word spotting process but only the main ones. This word spotting architecture is very common and has been used by several researchers [1,7]. It is illustrated in Fig. 4.

4.1.1. Image preprocessing and pseudo-word segmentation

First, we must pre-process each of the scanned document pages. To binarize the documents, we used the adaptive binarization technique proposed in [32]. Due to improper scanning, document images might be framed with unwanted text areas of the neighboring pages. Thus, after binarization, we apply the technique, described in [33], for obtaining proper text boundaries. Then, the text regions are segmented into either lines or pieces of lines, up to words or parts of words. Indeed, due to the properties of FSM, we do not require precise word segmentation. The textual elements obtained are called here "*pseudo-words*".

Word segmentation could be obtained by using Run Length Smoothing Algorithm (RLSA) based techniques. This can provide good results for high quality printed documents, but the results could be poor when inter-word spaces are variable which is often the case in handwritten documents as well as in historical printed

[11] Please note also that an advantage of line segmentation is that it can be possible to spot hyphenated words that span onto two lines.

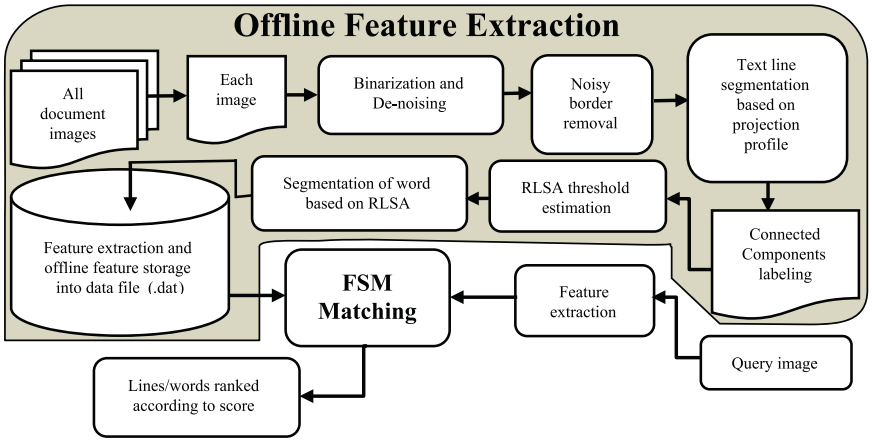


Fig. 4. (a) The block diagram of the proposed word spotting system.

documents. Thus, applying Horizontal RLSA (H-RLSA) with adaptive threshold [34] is a better solution. This threshold actually defines the average inter-character gap in a word, and it inherently depends on the considered dataset. To obtain it automatically, a preliminary text line segmentation is performed on the document pages. Proper segmentation of all of the text lines is not required in this case. We need to have only some prominent segmented text lines, which can help us to understand the inter-character gaps as well as the inter-word gaps in a text line. Thus, the basic text line segmentation algorithm based on the projection profile [34] is used here, and only the best segmented lines (based on the heights of the segmented text lines) are considered. Then, the average inter-word character gap is obtained from the histogram of gaps between connected components. Afterward, that H-RLSA is performed on the image by using the average character gaps as a threshold for obtaining the pseudo-words (words that are not necessarily well segmented).

4.1.2. Feature extraction

After obtaining pseudo-words (or lines), the next task is to extract features from them. Features are extracted from gray scale and also binary normalized images. In the following, we describe two different categories of features, namely column based features and a histogram of gradient-based features. Once the features are extracted, they are stored off-line for faster retrieval.

4.1.2.1. Column-based features. A set of statistical column based features, which have been used previously for handwriting recognition is broadly described in [35]. A subset of these features has been used by several researchers for word spotting [1,4]. One should notice that, these features are mostly based on the binary image, so they are not much sensitive to illumination variations, provided that the used binarization technique has taken into account the issues related to illumination variations, which is the case here with [32]. But some of these features are sensitive to the local noise, especially F_1, F_2, F_6. Although the column based features can be improved, selected, or outperformed in terms of accuracy by more complex features (e.g. gradient-based features [36], or graph similarity features [37]), due to their lower computational cost, they remain quite interesting and we chose not to focus on this point in this study.

Here, we have chosen 8 features, F_1, F_2, …, F_8 to describe each sequence using pixel column information. For an image with a width of N pixels, a sequence of N vectors in 8 dimensions is obtained by moving from left to right. A description of the features is given in Table 2. Among them, features $F1$ – $F6$ have been used

Table 2
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 greyscale 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 $1 \le n \le N$ $F7(n) = \begin{cases} \left[\frac{1}{\rho}\sum_{m=1}^{M} m \;\; \text{if } w_b(m,n)=1\right]; \lvert\rho \neq 0; \rho = \text{No. of foreground pixels at n}^{\text{th}} \text{ column;} \\ t \qquad \text{Obtained by interpolation;} \lvert\rho = 0 \end{cases}$
$F8$	Transition at C.G. obtained from $F7$ $F8(n) = \begin{cases} 1 \;\; w_b(F7(n),n)=0; \text{ and } w_b(F7(n-1),n)=1 \text{ or} \\ \quad w_b(F7(n),n)=1; w_b(F7(n-1),n)=0 \\ t \;\; \text{Obtained by interpolation} \end{cases}$

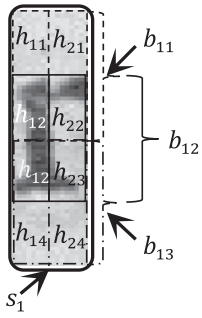


Fig. 5. Block normalization technique for SSHOG.

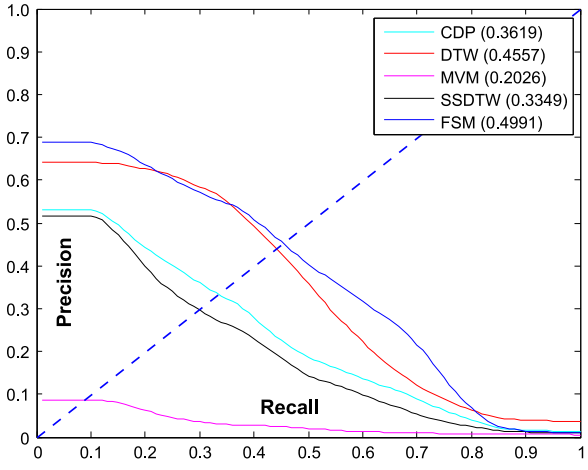


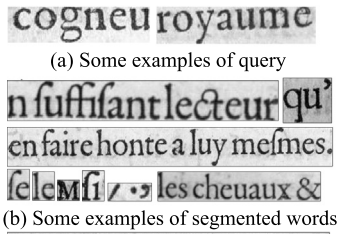
Fig. 6. Comparative performance of CDP, DTW, MVM, SSDTW and FSM on segmented words of GW dataset.

several times in the literature [1,4,36,38], but the features $F7$ and $F8$ are proposed in this work. $F7$, corresponds to the center of gravity of the foreground pixels inside a column. In the corresponding equation in Table. 2, w_b denotes the binarized version of the word image, m represents the row coordinate and n the column's coordinate. For columns that have no foreground pixels, $F7$ is calculated by nearest neighbor interpolation, with the help of the neighboring columns having foreground pixels. $F8$, which is calculated by using the location of the center of gravity (obtained from $F7$), is the number of transitions, from foreground to background (1–0) or from background to foreground (0–1) at these calculated centroid locations of pixels.

4.1.2.2. Slit style HOG based features. Slit style HOG (SSHOG) features were primarily introduced in [13]. We used the same features for some experiments in order to compare various approaches. HOG based features are less sensitive than column based features, considering noise and illumination variations because they do not need binarization and they are based on gradients of the image. For calculating this feature, a fixed sized window is slided over the image in a horizontal direction to extract the HOG features from each slit. For calculating the HOG descriptors, we need to divide the slit into smaller rectangular regions (called *cells*) along horizontal and vertical directions. A block is defined as a group of $h \times w$ cells. Blocks are slided over each slit that has the same width as the width of the block. The horizontal overlapping of the original HOG could be well realized by the sliding window and sequential representation of vectors. The relationships between slits, blocks and cells are shown Fig. 5. This figure shows, three blocks (b_{11}, b_{12}, b_{13}), each one composed of four cells (2×2), denoted by $h_{11}\ldots h_{24}$. Notice here that since SSHOG features are obtained from a slit of several pixels in width, skipping one element with FSM is corresponding of skipping one slit and thus several pixel columns. Such kind of side effect might need to be further studied.

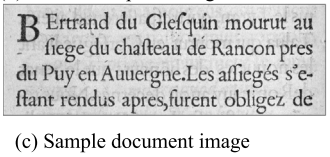
4.2. Results on segmented words

This experiment is performed with the George Washington historical dataset that contains 20 pages of letters, orders, and instructions written in 1755 by *George Washington* and some of his associates. Thus, obviously it inhibits some variations in writing style. The quality of the scanned pages varied from clean to difficult-to-read by humans (see Fig. 8(b)). This specific dataset was made by considering 10 better quality pages among the 20. We consider 14 query images (see Fig. 8(b)) and 2381 properly segmented target words coming from the 10 pages. To perform the experiment, we used the architecture and framework described in Section 4.1 with the column-based features (see Table 2). Please note that, due to improper segmentation, some of the target words can be smaller than considered query word. In such cases, that

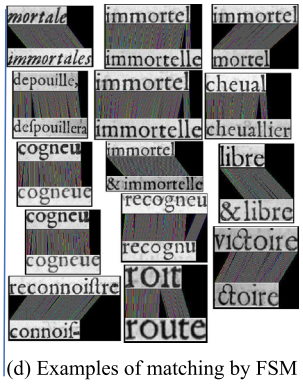


(a) Some examples of query

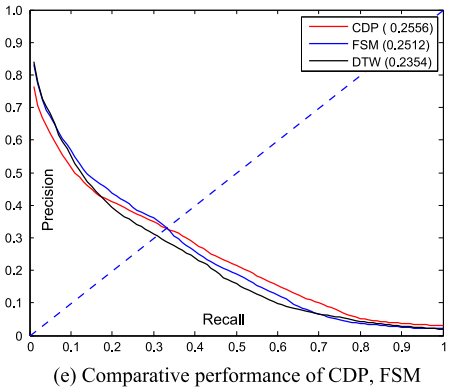
(b) Some examples of segmented words



(c) Sample document image



(d) Examples of matching by FSM



(e) Comparative performance of CDP, FSM and DTW

Fig. 7. (a,b,c) Some visual examples of the CESR dataset along with correspondences provided by FSM (d) The top images are queries, and the bottom ones are targets. (e) Comparative results of DTW, CDP and FSM.

Table 3
Statistics and results of CESR dataset by groups of queries. $\breve{N}^{\circ}$=No. of occurrences of the query word. $\breve{C}$ = mAP of CDP; $\breve{F}$ = mAP of FSM; TW=Total words in target set; $\bar{AC}$=average mAP of CDP; $\bar{AF}$=average mAP of FSM.

$\breve{Q}^{\circ}$	$\breve{N}^{\circ}$	$\breve{C}$	$\breve{F}$
libre	19	0.712	0.661
liberte	2	0.308	0.308
librement	3	0.189	0.176
TW = 3192; $\bar{AC}$ = 0.403; $\bar{AF}$ = 0.382			
cheualier	4	0.501	0.494
cheual	38	0.509	0.443
cheualerie	1	0.451	0.445
cheuallier	1	0.421	0.516
cheuaus	4	0.395	0.436
cheuaux	8	0.400	0.333
TW = 7570; $\bar{AC}$ = 0.446; $\bar{AF}$ = 0.445			
victoire	23	0.646	0.575
victo	1	0.593	0.510
victoi	2	0.542	0.443
victoire	23	0.646	0.575
victorieus	5	0.375	0.425
victorieux	6	0.518	0.541
TW = 8220; $\bar{AC}$ = 0.553; $\bar{AF}$ = 0.512			
mortel	11	0.363	0.352
immortales	1	0.109	0.107
immortalit	8	0.301	0.283
immortel	14	0.172	0.154
immortelle	10	0.194	0.162
immortels	2	0.197	0.171
mort	28	0.071	0.087
mortale	4	0.134	0.140
mortales	1	0.107	0.120
mortalia	5	0.090	0.075
mortalibus	2	0.078	0.051
mortalite	1	0.225	0.225
morte	10	0.047	0.049
mortelle	9	0.285	0.285
mortelles	8	0.236	0.214
mortels	4	0.199	0.190
mortem	5	0.025	0.018
mortes	1	0.040	0.081
morgue	2	0.018	0.015
morti	1	0.176	0.160
mortis	3	0.032	0.028
morts	3	0.065	0.080
mortua	1	0.022	0.016
mortuum	1	0.079	0.059
mortuus	1	0.021	0.016
mortz	3	0.060	0.080
TW = 13518; $\bar{AC}$ = 0.129; $\bar{AF}$ = 0.124			

$\breve{Q}^{\circ}$	$\breve{N}^{\circ}$	$\breve{C}$	$\breve{F}$
roy	119	0.272	0.304
royales	1	0.042	0.029
royaume	2	0.027	0.035
royaumes	4	0.038	0.037
royaus	1	0.082	0.051
royaute	2	0.116	0.068
royaux	1	0.107	0.060
royne	8	0.252	0.116
roys	17	0.173	0.145
rois	16	0.021	0.044
roistre	1	0.012	0.012
roit	20	0.017	0.038
TW = 20696; $\bar{AC}$ = 0.097; $\bar{AF}$ = 0.078			
reconneut	1	0.045	0.117
reconnoi	1	0.047	0.130
reconnoissable	2	0.343	0.344
reconnoissance	6	0.318	0.358
reconnoissant	1	0.334	0.328
reconnoissent	1	0.307	0.112
reconnoissoien	1	0.219	0.258
reconnoissoit	2	0.291	0.343
reconnoissons	3	0.344	0.333
reconnoistre	7	0.140	0.174
reconnoit	5	0.052	0.184
reconnoitre	4	0.162	0.207
reconnoy	3	0.088	0.084
reconnu	2	0.044	0.172
reconnue	1	0.037	0.097
TW = 6117; $\bar{AC}$ = 0.185; $\bar{AF}$ = 0.216			
reconoissent	1	0.092	0.116
reconoissoit	1	0.104	0.100
reconoitre	1	0.055	0.050
recogneu	1	0.143	0.134
recognois	1	0.328	0.352
recognoissan	1	0.312	0.251
recognoissance	3	0.330	0.318
recognoissant	1	0.297	0.249
recognoisse	1	0.439	0.413
recognoissons	1	0.426	0.378
recognoistre	8	0.118	0.139
recognoit	1	0.343	0.327
recognoitre	2	0.150	0.142
recognu	1	0.148	0.137
TW = 3524; $\bar{AC}$ = 0.235; $\bar{AF}$ = 0.221			

$\breve{Q}^{\circ}$	$\breve{N}^{\circ}$	$\breve{C}$	$\breve{F}$
connoi	4	0.019	0.035
connois	5	0.312	0.329
connoissan	11	0.394	0.396
connoissance	34	0.380	0.382
connoissans	2	0.175	0.181
connoissant	4	0.401	0.324
connoisse	1	0.429	0.407
connoissent	4	0.355	0.365
connoissoit	3	0.380	0.414
connoist	1	0.276	0.371
connoistre	12	0.176	0.197
connoit	5	0.035	0.053
connoitre	3	0.043	0.047
connoy	5	0.021	0.029
connu	3	0.018	0.028
connue	1	0.031	0.029
connues	4	0.029	0.028
connus	1	0.018	0.023
TW = 13431; $\bar{AC}$ = 0.194; $\bar{AF}$ = 0.202			
cognoissons	1	0.505	0.496
cognoistre	12	0.583	0.543
cognoit	6	0.147	0.112
cognoitre	2	0.371	0.441
cognosci	1	0.142	0.296
cogneu	3	0.098	0.102
cogneue	2	0.065	0.086
cogneut	1	0.121	0.095
cognoi	1	0.094	0.086
cognois	5	0.519	0.513
cognoisoit	1	0.569	0.518
cognoissan	2	0.516	0.505
cognoissance	29	0.528	0.473
cognoissances	1	0.500	0.405
cognoissant	1	0.486	0.460
cognoissent	6	0.492	0.438
cognoissions	1	0.470	0.479
cognoissoint	1	0.499	0.465
TW = 11024; $\bar{AC}$ = 0.373; $\bar{AF}$ = 0.362			
despouille	4	0.721	0.656
despouiller	5	0.785	0.742
despouillera	1	0.763	0.792
depouille	1	0.754	0.746
depouiller	1	0.392	0.358
TW = 1647; $\bar{AC}$ = 0.683; $\bar{AF}$ = 0.659			

(a) Query images used for GW dataset

(b) Sample page from the GW dataset

(c) Query images used for Japanese dataset

(d) Sample page from the Japanese dataset

Fig. 8. Query images and sample scanned page images from GW and Japanese datasets. (a) Query images used for GW dataset. (b) Sample page from the GW dataset. (c) Query images used for Japanese dataset. (d) Sample page from the Japanese dataset.

particular small target word is considered as query and the original query word is considered as target word.[12] The performances are evaluated using the mean average precision (mAP)[13] for all of the selected 14 query words as well as Precision/Recall curves based on number of retrieved elements considered. The results, shown in Fig. 6, demonstrates that FSM significantly outperforms all other algorithms.

4.3. Results on segmented pseudo-words

The application of FSM on segmented pseudo-words is evaluated by an experiment on a dataset called CESR. This dataset comes from the resources of the Centre d'Etude Suprieure de la Renaissance,[14] through the BVH (Bibliothèques Virtuelles Humanistes[15]) project. The CESR has a collection of precious historical books, that date from the middle of the XIV century to the beginning of the XVII century. The dataset was composed from the two volumes of *Essais de messire Michel Seigneur de Montaigne, Chevalier de l'order du Roy, & Gentil-homme ordinaire de Sa Chambre*. This first edition was published in Bordeaux in 1580 by S. Millanges[16]. The languages used in the books are Latin or French. All the pages were scanned with a resolution of 312 × 312 dpi and were saved in grey-scale format (see Fig. 7). Vol.I contains 520 pages and Vol.II contains 676 pages. To build the dataset, we selected 123 relevant queries from the user's point of view, such as those that could have lexical variations. Each query and its derivatives are grouped for evaluation: each image in a group is considered to be a query, and then, all of the other words in the group are considered to be relevant targets and must be retrieved also. For each query (and its derivatives), the research is performed with the target words, segmented from the set of full page images, in which at least one target appears. The total number of target words in each group varies from 1700 to 21000 approximatively. The statistics are given in Table 3.

4.3.1. Experimental Setup

To perform experiments with this dataset, we used the architecture and framework, described in Section 4.1. The column-based features (see Table 2) are used. Very small segmented words (e.g., one or two characters) are pruned by considering their width with respect to the width of the original query word, which allows faster experiments We heuristically decided that, if the width of any target word is less than 35% of the width of the query word, then it is not considered for matching.

[12] In such cases, the 2nd way of normalization is used for FSM, to avoid wrongly segmented small words from appearing at top ranked positions in overall nearest neighbor ranking process. Refer to Algorithm 1, line 32.

[13] ⟨https://en.wikipedia.org/wiki/Information_retrieval#Mean_average_precision⟩.

[14] ⟨http://www.cesr.univ-tours.fr/⟩.

[15] See Bibliothèques Virtuelles Humanistes Project: ⟨http://www.bvh.univ-tours.fr/presentation_en.asp⟩.

[16] Please see the following links for more details about the book: ⟨https://www.lib.virginia.edu/rmds/collections/gordon/literary/montaigne/bibliography.html⟩ and ⟨http://www.search.lib.virginia.edu/catalog/u50318⟩.

Table 4
Statistics of query words in GW and Japanese datasets.

	$\breve{Q}^\circ$	$\widetilde{N}^\circ(\hat{N}^\circ)$	$\bar{C}$	$\bar{F}$
GW Dataset	1755	37 (0)	0.701	0.591
	Company	20 (1)	0.737	0.661
	Cumberland	13 (1)	0.818	0.843
	December	26 (0)	0.755	0.760
	Fort	22 (0)	0.689	0.660
	Instructions	21 (1)	0.955	0.959
	Letters	22 (0)	0.841	0.883
	October	15 (1)	0.705	0.713
	Orders	33 (0)	0.601	0.568
	Recruits	12 (0)	0.837	0.746
	Sergeant	12 (0)	0.731	0.800
	Virginia	14 (0)	0.515	0.494
	Winchester	15 (4)	0.693	0.676
	Captain	27 (1)	0.660	0.648
	Regiment	17 (0)	0.546	0.420
Japanese	A.Matazaemon	154 (10)	0.841	0.808
	B. Uriyamusu	68 (5)	0.821	0.764
	C.InoueTomizou	22 (3)	0.722	0.721
	D.IshizukaKanzou	21 (4)	0.603	0.598

$\widetilde{N}^\circ$ = No. of full words; $\bar{C}$ = mAP of CDP
$\hat{N}^\circ$ = No. of hyphenated words; $\bar{F}$ = mAP of FSM

4.3.2. Results

Overall, the results of FSM and CDP are better than those of DTW (see Fig. 7(e)). The overall difference between CDP and FSM seems not very significant (less than 0.5%) but it seems that FSM has a higher precision (nearly 1%) at first ranks. Looking at more detailed results (see Table 3), we can see that FSM outperforms or equals CDP in nearly half the cases (see the highlighted rows). These close results for all algorithms might be explained by the fact that the dataset is very difficult (low mAP) and so differences cannot be well highlighted. Nevertheless, we can see in Fig. 7(d) that matching outputs of FSM are meaningful and allow to skip irrelevant or noisy parts of target, which is not directly possible with CDP.

4.4. Results on segmented lines

The experiments in this category are performed with two handwritten historical datasets: *George Washington (GW)* and *Japanese* datasets. The full *George Washington* dataset is used and contains 675 text lines. The correctly segmented lines are obtained from [13]. The *Japanese* dataset comes from the manuscripts of "*Akoku Raishiki (The diary of Matsumae Kageyu)*". It is composed of 92 scanned images (a total of 1576 segmented lines) scanned in 72×72 ppi (see Fig. 8(d)). It is obtained from [13].

4.4.1. Experimental Setup

To perform experimentation with the *George Washington* dataset, 15 query images are used (same as previously). These are the same, as the one used in [15,13] (see Fig. 8(a) and Table 4). For the case of the *Japanese* dataset, the four query images (taken from [13]) are shown in Fig. 8(c). As mentioned above, we used the same segmented lines as those used in [13]. SSHOG features are used in this case. Their values for these two datasets are the same as in [13]. Consequently, the only differences in performances between the considered approaches presented here, come from the matching techniques.[17]

4.4.2. Results

The P-R curves as well as the mAP for each algorithm are given in Fig. 9. Fig. 9(a) demonstrate first the ability of FSM to behave similar to other algorithms (MVM and SSDTW here), as shown in Appendix E. Indeed, we can see that FSM turned into MVM (MVM-FSM) and FSM turned into SSDTW (SSDTW-FSM) give exactly same results as the original algorithms on the GW dataset. Next, it can be visible from Fig. 9(b) and (c), that FSM outperforms other classical sequence matching techniques, except CDP. Nevertheless, the difference in mAP is minor (0.025 and 0.024 for the GW and Japanese datasets respectively). Most likely, the specific DP path associated with SSHOG features for CDP are responsible. Indeed, we can also observe that the result of the M-CDP algorithm (CDP using the classical DTW DP-path) is much less accurate than the original CDP as well as FSM that has some similarities (notice also that FSM turned into modified CDP (M-CDP-FSM) achieves the same performances as M-CDP). Considering FSM, maybe the skipping facility should be adapted to the specificities of the SSHOG features (as mentionned before, skipping one element with SSHOG is in fact same as skipping several pixel columns). As a counterpart, thanks to this skipping ability, FSM can achieve better localization of the spotted words and also a better correspondence between target and query elements, which could be useful for the user as feedback to understand the retrieval process.

5. Conclusion and future works

In this paper, we presented a new robust sequence-matching algorithm called the FSM algorithm, which can be easily modified into the architecture of other sequence matching techniques, e.g., DTW, SSDTW, MVM, and CDP. Specifically, FSM also includes the ability to skip outliers from any position of the target sequence. This, coupled with the facilities for many-to-one and one-to-many matching, makes the proposed FSM algorithm robust, general (it can be easily modified to behave differently) and applicable for various domains of time series sequence matching, e.g., finance, video retrieval, shape matching [23,29,24,25].

We have demonstrated the usefulness of the proposed FSM algorithm in the domain of word spotting for segmented lines as well as correctly/incorrectly segmented words, where partial matching of query as well as searching for derivatives of query words are needed. In such situation, FSM is among the best sequence matching methods. Only CDP is outperforming it on some datasets, especially when SSHOG features are used instead of common column-based features. This might be explained by the relationship between the feature extraction level and the skipping ability, which must be further studied. Other limitations of the proposed system are the same as for any other sequence matching technique, including difficulties to handle font and script variabilities, being able to spot arbitrary keywords, etc. In such directions, more experimental investigations are needed for finding a robust set of features that can account for the writer independence and font variability of handwritten and printed

[17] Nevertheless, the ground truth is minutely different in our case, which could explain some of the differences with regard to CDP in the original paper [13]. Indeed, the ground truth data contains the image name and location of each query word, and a single query word can appear several times in the same line. However, only the CDP algorithm can spot multiple query words in a single line, if a good distance threshold (difficult to estimate and highly data dependent) can be estimated. For a fair comparison with other algorithms, and to avoid manual determination of such a threshold, we performed some duplication of lines that have multiple occurrences of query words with only one different occurrence retained in each line. In the case of the occurrence of hyphenated words in consecutive two lines, we merged these two line images and corresponding feature vectors for matching.

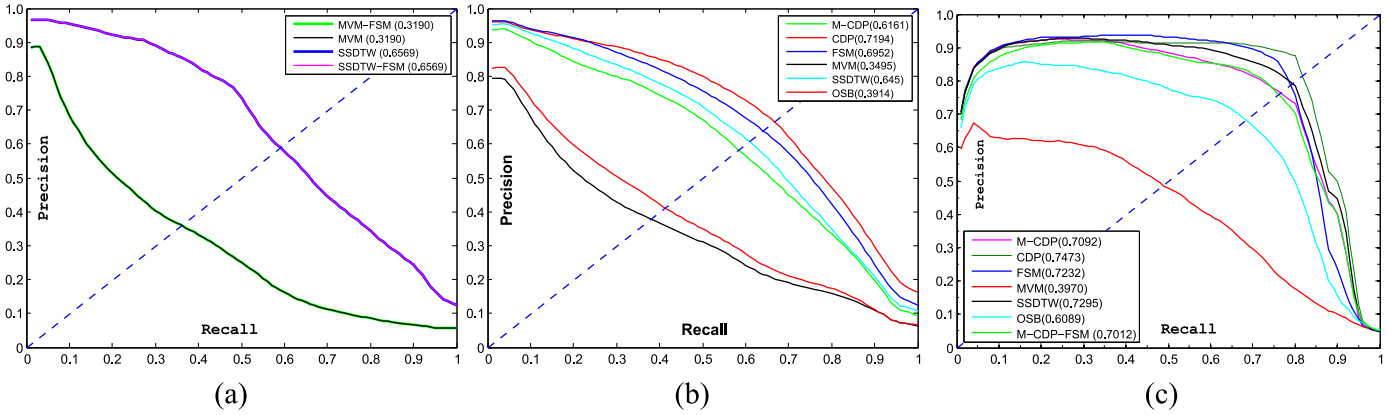


Fig. 9. (a) Performance comparison of MVM with MVM-FSM and of SSDTW with SSDTW-FSM (see the digitized version of the image, because the curves are completely overlapping). (b) Performance comparison of CDP, M-CDP, FSM, MVM, and SSDTW on the GW dataset. (c) Performance comparison of CDP, M-CDP, M-CDP-FSM, FSM, MVM, SSDTW, and OSB on the Japanese dataset (Please see Appendix E for details on MVM-FSM, SSDTW-FSM, M-CDP, M-CDP-FSM).

manuscripts. We also plan to perform more experiments to evaluate the robustness of FSM algorithm in comparison with other approaches on additional larger and multilingual datasets. We would also like to work on the reduction of time complexity of the FSM algorithm. Finally, the characteristics of FSM can be helpful in other domains of time series, e.g., finance, video retrieval, and shape matching [23,29,24,25], and we would like to investigate these fields.

Conflict of interest

none declare

Acknowledgements

This work has been supported by the Indo-French Center for Promotion of Advanced Research (IFCPAR/CEFIPRA) under project no 4700-IT1. We would like to thank Dr. Kengo Terasawa (Department of Media Architecture, Future University-Hakodate, Japan) for providing us the segmented lines, ground truth and feature vectors [13] of the George Washington and Japanese datasets.

Appendix A. Comparison of properties and time complexities of sequence matching techniques

In this section, readers will find in Table A1, a summary of main properties (continuity, boundary conditions, type of allowed correspondences, etc.) of common sequence matching techniques. Table A2 is dedicated to time complexities of these algorithms.

Appendix B. Dynamic Time Warping (DTW)

To align two finite sequences: $x = (x_1, x_2, \ldots, x_p)$ and $y = (y_1, y_2, \ldots, y_q)$; using DTW, we construct an $p \times q$ distance matrix $(\mathfrak{D}_{(i,j)} = dist(x_i, y_j))$. The best warping path $(W = w_1, w_2, \ldots, w_K;\ max(p, q) \le K \le p + q - 1)$ between these sequences defines an optimal mapping between x and y. The k^{th} element of W is defined as $w_k = (i, j)_k$. The optimal warping path minimizes the warping cost (6). This optimal path can be found by using dynamic programming (7), where $\mathfrak{P}_{(1,1)} = \mathfrak{D}_{(1,1)}$; $\mathfrak{P}_{(i,0)} = \mathfrak{P}_{(i-1,0)} + \mathfrak{D}_{(i,0)}$; $\underset{1<i\le p}{}$ $\mathfrak{P}_{(0,j)} = \mathfrak{P}_{(0,j-1)} + \mathfrak{D}_{(0,j)}$ $\underset{1<j\le q}{}$. The optimal distance $(DTW(x, y))$ is obtained from the cell $\mathfrak{P}_{(p,q)}$.

$$DTW(x, y) = min\sqrt{\sum_{k=1}^{K} \mathfrak{D}_{(w_k)}} \tag{6}$$

$$\mathfrak{P}_{(i,j)} = \mathfrak{D}_{(i,j)} + min\begin{Bmatrix} \mathfrak{P}_{(i,j-1)} \\ \mathfrak{P}_{(i-1,j-1)} \\ \mathfrak{P}_{(i-1,j)} \end{Bmatrix}_{i=1,2,\ldots,p;\ j=1,2,\ldots,q} \tag{7}$$

Appendix C. Minimal Variance Matching (MVM)

This algorithm is designed to handle partial sequence matching of two sequences: $x = (x_1, x_2, \ldots, x_p)$ and $y = (y_1, y_2, \ldots, y_q)$; $p \le q$. 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:

$$\mathcal{H}(\mathfrak{D}_{k,l}, \mathfrak{D}_{i,j}) = \begin{cases} \mathfrak{D}_{i,j} & \text{if } i = k + 1 \text{ and } l + 1 \le j \le \mathcal{L} \\ \infty & \text{otherwise} \end{cases} \tag{8}$$

$$\mathcal{L} = (l + 1) + \text{elasticity} - |j - i|;\ \text{elasticity} = |q - p| \tag{9}$$

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

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

$$\mathfrak{L} = 2 \le i \le m;\ i \le k \le i + (q - p);\ k + 1 \le j \le k + 1 + (q - p) \tag{10}$$

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 $j = 1\ldots. (q - p) + 1$; ii) ending at some node in the last row $\mathfrak{D}_{p,j}$ for $j = p\ldots. q$. To find it, we just need to check the minimum value in $\mathfrak{P}$ at ending positions and then to back track to obtain the path.

Table A1
Properties of main sequence matching techniques.

$\mathcal{MN}$	$\mathcal{CS}$	$\mathcal{BC}$	$\mathcal{OS}$
DTW, DDTW, WDTW, WDDTW	Each query element has to be matched with target elements without skipping any elements from target and query sequence (and vice versa)	Correspondence involves every elements of the target and query sequence, including 1st and last elements	**NC:** One-One, Many-One and One-Many; **Pros:** These algorithms can handle the presence of stretching and contraction in the signals. DDTW uses derivative of features and is helpful in the presence of outliers In WDTW, sequence elements are weighted based on phase difference. The characteristics of DDTW and WDTW is merged together in WDDTW. WDTW and WDDTW are useful for handling the phase difference in the query and target signals; **Cons:** These algorithms cannot skip outliers present at any position of the query and target signals. Tuning the parameters for WDTW and WDDTW is difficult and only can be done heuristically
SSDTW, CDP	Partial sequence matching is possible. Skipping outliers only at the end and beginning of the target signal is possible	Every query elements are involved but the correspondence with target signal can start and end at any position of the target sequence	**NC:** One-One, Many-One and One-Many; **Pros:** Both of these techniques can skip outliers at the end and at the beginning of the target signal only. Due to the associated weights with DP paths, and the specific DP path, CDP performs best; **Cons:** Unlike SSDTW and other algorithms, no element-wise correspondence can be provided for CDP
LCSS	Skipping of query and target elements are possible (no skipping penalty)	Correspondence between target and query signal can start and end at any position of the query and target sequence	**NC:** One-One; **Pros:** Can skip outliers from any positions of query and target signals; **Cons:** Stretching and contraction present in the signals can not be handled and a threshold is needed to define when two elements are similar, which is very difficult to decide
MVM	Each query element has to be matched with one target element but not vice versa. Skipping is possible at any position of target signal, i.e. at the beginning, end and inside matched sequence	Every elements of query signal are involved but correspondences with target signal can start and end at any position	**NC:** One-One; **Pros:** Outliers present at any position of target signal can be skipped hence partial sequence matching is possible; **Cons:** Query sequence has to be smaller than target and there is no skipping penalty. Can't handle stretching and contraction of signals
OSB	Skipping of elements at any position of query and target signals are possible (penalty is added for skipping)	Same as LCSS	**NC:** One-One; **Pros:** Outliers present at any position of query and target signals can be skipped hence partial sequence matching is possible. Skipping penalty is present; **Cons:** Can't handle stretching and contraction of signals. Very high computational complexity
FSM	Each query elements has to be matched with some target elements but not vice versa. Skipping is possible at any position of target signal	Same as MVM	**NC:** One-One, Many-One and One-Many; **Pros:** Outliers can only be skipped from any positions of the target signal hence partial sequence matching is possible. Skipping penalty is present. Can handle stretching and contraction of target and query signals; **Cons:** Query length has to be less than or equal to target length. Outliers in query can't be skipped

$\mathcal{MN}$ = Method Name ; $\mathcal{CS}$ = Continuity (Skipping); $\mathcal{BC}$ = Boundary Condition; **NC**=Nature or correspondence $\mathcal{OS}$ = Other specificities.

Table A2
Time complexities of main sequence matching techniques.

Method Name	**Time complexity** (where, p and q represents the sizes of sequences, to matched)
DTW [24]	The complexity of DTW is $O(pq)$. It can be reduced to $O(p)$ with Sakeo-Chiba [24] and Itekura bands [24]
DDTW [28]	Same as DTW. There are some added constant factors because of derivative estimating step but it is negligible
WDTW [29]	Same as DTW. There are weight factors to be calculated and multiplied with each cell, but the time complexity for that is negligible
WDDTW [29]	Same as DTW
SSDTW [24]	Same as DTW
LCSS [26]	Same as DTW
MVM [23]	The complexity of MVM is $O(pq^2)$. But if "*Corresponding window bound*" (refer to [23]) is introduced, the complexity can be reduced to $O(pq)$. Moreover, when the query sequence is to be matched with complete target sequence, the complexity is reduced to $O(p)$
OSB [25]	The time complexity of OSB is $O(p^2q^2)$. By imposing warping window restriction on OSB, and by limiting the number of elements that can be jumped, we can reduce the time complexity of OSB up-to $O(p)$
FSM [Proposed]	The time complexity of FSM is $\Theta(3pq^2)$ (see Section 3.5)

Appendix D. Continuous Dynamic Programming algorithm

Let's consider two finite sequences x (query) and y (target), of lengths p and q, respectively (see Section 3). For details on CDP,[20] please see [39]. The path cost matrix ($\mathfrak{P}$) is obtained in the following way:

$$\mathfrak{P}(j,i)=\begin{cases}3\times\mathfrak{D}(j,1) & \text{if } i=1\\ \min\begin{pmatrix}\mathfrak{P}(j-2,1)+2\times\mathfrak{D}(j-1,2)+\mathfrak{D}(j,2)\\ \mathfrak{P}(j-1,1)+3\times\mathfrak{D}(j,2)\\ \mathfrak{P}(j,1)+3\times\mathfrak{D}(j,2)\end{pmatrix} & \text{if } i=2\\ \min\begin{pmatrix}\mathfrak{P}(j-2,i-1)+2\times\mathfrak{D}(j-1,i)+\mathfrak{D}(j,i)\\ \mathfrak{P}(j-1,i-1)+3\times\mathfrak{D}(j,i)\\ \mathfrak{P}(j-1,i-2)+3\times\mathfrak{D}(j,i-1)+3\times\mathfrak{D}(j,i)\end{pmatrix} & \text{if } 3\le i\le p\end{cases} \tag{11}$$

The output is obtained as $A(j)=\frac{1}{3.p}\mathfrak{P}(j,p)$. 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 the specific initial condition of $\mathfrak{P}(-1,i)=\mathfrak{P}(0,i)=\infty$.[18] As it can be visible from (11), more resistance is present for the diagonal link of the DP path compared to other two links. We investigated the effectiveness of this particular DP (see (11)) path and we found that it has some positive influence on the performances for word spotting, compared to the traditional DP Path (see (12)). In such case, the output is obtained by $A(j)=\frac{1}{p}\mathfrak{P}(j,p)$. Please see Section 4.4.2 and Fig. 9 where CDP with original DP Path and its modified version with DTW DP Path (*Modified CDP*: M-CDP) are compared.

$$\mathfrak{P}(j,i)=\begin{cases}\infty & \text{if } (1\le i\le p; j=1,2\\ \mathfrak{D}(j,i) & \text{if } i=1;\ 3\le j\le q\\ \min\begin{pmatrix}\mathfrak{P}(j-1,i-1)+\mathfrak{D}(j,i)\\ \mathfrak{P}(j-1,i)+\mathfrak{D}(j,i)\\ \mathfrak{P}(j,i-1)+\mathfrak{D}(j,i)\end{pmatrix} & \text{if } 1<i\le p;\ 3\le j\le q\end{cases} \tag{12}$$

Appendix E. Generalization property of FSM

The architecture of FSM is general enough so that, with little modifications, it can be easily tuned to perform as other useful sequence matching algorithms.

E.1. Attaining DTWs behavior from FSM

The architecture of FSM can be easily tuned to perform as DTW, which can give completely identical results (see Fig. 9 (a)). The main idea here is to follow the same DP path as DTW and to restrain the skipping facility of FSM (see Algorithm 2 below.[19]

Algorithm 2. FSM as DTW.

Input: $x_{1\ldots p}, y_{1\ldots q}, \mathfrak{D}_{pq}, \mathfrak{S}, \mathfrak{C}$
Output: $\mathfrak{P}_{pq}, \mathfrak{W}$

1.1 $\mathfrak{P}\leftarrow\infty$
1.2 $elasticity\leftarrow 0; \mathfrak{S}=0; \mathfrak{C}=0$ ▷ No skipping
2.1 $\mathfrak{P}(1,1)\leftarrow\mathfrak{D}(1,1)$
2.2 **for** $j\leftarrow 2$ ***to*** q **do**
3 $\quad\mathfrak{P}(1,j)\leftarrow\mathfrak{D}(1,j)+\mathfrak{P}(1,j-1)$
4 **for** $i\leftarrow 2$ ***to*** p **do**
11 $\quad$**for** $k\leftarrow 1$ ***to*** q **do**
12 $\quad\quad D\leftarrow(k+1)$
13 $\quad\quad$**for** $j\leftarrow k$ ***to*** D **do**
$\quad\quad\quad$.... DTW equations ...
25
26 ▷ Warp. path starts from bottom right most cell
27 $\mathfrak{r}=q; \mathfrak{c}=p; cnt=1$
.....
32 ....

Algorithm 3. FSM as MVM.

Input: $x_{1\ldots p}, y_{1\ldots q}, \mathfrak{D}_{pq}$
Output: $\mathfrak{P}_{pq}, \mathfrak{W}$

1.1 $\mathfrak{P}\leftarrow\infty$
1.2 $elasticity\leftarrow|q-p|$
2 **for** $j\leftarrow 1$ ***to*** q **do**
3 $\quad\mathfrak{P}(1,j)\leftarrow\mathfrak{D}(1,j)$
4 **for** $i\leftarrow 2$ ***to*** p **do**
9 $\quad R\leftarrow max(1,\{(i-1)+elasticity\})$
10 $\quad L\leftarrow min(q,(i-1))$
11 **for** $k\leftarrow L$ ***to*** R **do**
12 $\quad D\leftarrow(k+1+elasticity)-(|k-(i-1)|)$
13 $\quad$**for** $j\leftarrow(k+1)$ ***to*** D **do**
20 $\quad\quad$**if** $\mathfrak{P}(i,j)>\mathfrak{P}(i-1,K)+\mathfrak{D}(i,j)$ **then**
21 $\quad\quad\quad\mathfrak{P}(i,j)\leftarrow\mathfrak{P}(i-1,K)+\mathfrak{D}(i,j)$
22 $\quad\quad\quad\mathscr{R}(i,j)\leftarrow i-1; \mathscr{C}(i,j)\leftarrow k$
26 $\mathfrak{I}=\underset{p\le t\le q}{\operatorname{argmin}}\,\mathfrak{P}(p,t)$
27 $\mathfrak{r}=p; \mathfrak{c}=\mathfrak{I}; cnt=1$
.....

The required changes are following : (***i***) The parent nodes are looped through all the columns of the array (line 11) and the elasticity of skipping of child nodes are completely constrained (assigned zero). In DTW, there are no possibility of skipping and the DP path is constructed through all the cells of every row of the dissimilarity matrix ($\mathfrak{D}$). (***ii***) Consider the sequence dissimilarity value from bottom right most cell of path cost matrix and backtrack the warping path from this cell and up to the top left most cell (line 27 of Algorithm 2). The final dissimilarity value is normalized through dividing the path cost value at the bottom right cell of $\mathfrak{P}$ matrix by total number of correspondence between query and target sequence. The number of entries in warping path $\mathfrak{W}$, gives total number of correspondence.

E.2. Attaining MVM behavior from FSM (MVM-FSM)

Attaining the MVM architecture from FSM, is quite easy, since FSM is developed on the basis of MVM. Interested readers are requested to see [23] for detailed description on MVM. The required changes (that gives same results as original MVM, see Fig. 9(a)) are as follows (see Algorithm 3): i) Restrict the limit of parent node (replace line 5–10 by line 9–10 of Algorithm 3). ii) For each parent node, the child node always start from diagonal position (line 12). iii) The portion, responsible for horizontal link of the dynamic programming (DP) path for FSM is removed here (lines 23–25 of Algorithm 1 are removed here). The final dissimilarity value is normalized here by dividing through the total number of query elements.

[18] For implementation, two more rows are added at the beginning of path cost matrix $\mathfrak{P}$ and initialized with ∞, so the size of path cost matrix becomes $\{(p+2)\times q\}$ (to get the final corresponding indexes of target sequence, we subtract the resultant indexes by 2). The above mentioned DP path formation (see (11)) starts from i=3 which corresponds to the 1st ($i=1$) row of the distance matrix $\mathfrak{D}$. Another way to achieve the particular initial condition is to initialize the first two rows of the path cost matrix with ∞ (no extra rows are added) and to follow the same aforementioned DP path (see (11)) from third row onwards. In case of our application, where the length of target and query sequence is quite large and always greater than two, these two different initialization processes give identical results.

[19] Please note here that line numbering in these modified versions of FSM algorithm keep same line numbering as in original algorithm. It means for example that lines 1.2 is added and that lines between 4 and 11 are removed).

E.3. Attaining CDP behavior from FSM (M-CDP-FSM)

CDP[20] architecture [39] can also be achieved by the following modifications of FSM algorithm. The original DP path of CDP can not be realized in this adaptive version of the algorithm. So to realize the CDP architecture, (refer Appendix D), the complex DP path of original CDP is modified to a simpler DP path of classical DTW. This results to a small decrease of accuracy compared to original CDP but it allows us to compare modified CDP (M-CDP) alike FSM, based on classical DTW's DP path (see Fig. 9(c)). The proposed modifications are mentioned in Algorithm 4. (***i***) Query and target elements are reversed (target is now row-wise) and parents are iterated for all query elements (line 12). (***ii***) Lines 14 to 25 are almost the same but it includes initialization of first column and jumpcost $\mathcal{J}$ is removed). (***iii***) The variable $\mathfrak{I}$, mentioned in line 32, stores the normalized distances. The normalized distances are obtained by dividing the path cost value with the length of reference sequence (p). The minimum value of $\mathfrak{I}$ gives the indexes of sub-sequence (of target signal), which has optimal match with reference sequence and the final dissimilarity cost between the target and query sequences.

Algorithm 4. FSM as CDP.

Input: $x_{1\ldots p}, y_{1\ldots q}, \mathfrak{D}_{pq}, \mathfrak{S}, \mathfrak{C}$
Output: $\mathfrak{P}_{pq}, \mathfrak{W}$

1.1 $\mathfrak{P} \leftarrow \infty$
1.2 $elasticity \leftarrow 0; \mathfrak{S} \leftarrow 0; \mathfrak{C} \leftarrow 0$
2 **for** $j \leftarrow 1$ ***to*** p **do**
3 $\quad \mathfrak{P}(1, j) \leftarrow \mathfrak{D}(1, j)$
4 **for** $i \leftarrow 2$ ***to*** q **do** // Target elements are considered row-wise in path cost matrix.
11 $\quad$ **for** $k \leftarrow 1$ ***to*** p **do** // Query elements are considered column-wise[21]. The loop for -
12 $\quad\quad D \leftarrow min(k+1, p)$ // - parent nodes are iterated for all query elements.
13 $\quad\quad$ **for** $j \leftarrow k$ ***to*** D **do** // Calculation of child nodes are same as DTW
14 $\quad\quad\quad$ **if** $j = 1$ **then**
15 $\quad\quad\quad\quad \mathfrak{P}(i, j) \leftarrow \mathfrak{D}(i, j)$ // 1^{st} column is copied from distance matrix.
16 $\quad\quad\quad$ **else**
20 $\quad\quad\quad\quad$ **if** $\mathfrak{P}(i, j) > (\mathfrak{P}(i-1, k) + \mathfrak{D}(i, j))$ **then**
21 $\quad\quad\quad\quad\quad \mathfrak{P}(i, j) \leftarrow (\mathfrak{P}(i-1, k) + \mathfrak{D}(i, j))$
22 $\quad\quad\quad\quad\quad \mathscr{R}(i, j) \leftarrow i-1; \mathscr{C}(i, j) \leftarrow k$
23 $\quad\quad\quad\quad$ **if** $\mathfrak{P}(i, j) > \mathfrak{P}(i, j-1) + \mathfrak{C} + \mathfrak{D}(i, j)$ **then**
24 $\quad\quad\quad\quad\quad \mathfrak{P}(i, j) \leftarrow \mathfrak{P}(i, j-1) + \mathfrak{C} + \mathfrak{D}(i, j)$
25 $\quad\quad\quad\quad\quad \mathscr{R}(i, j) \leftarrow i\ \mathscr{C}(i, j) \leftarrow j-1$
32 $\quad \mathfrak{I}(i, 1) = \mathfrak{P}(i, k)/k$

E.4. Attaining Subsequence DTW architecture from FSM

Let $x = (x_1, x_2, \ldots..x_p)$ and $y = (y_1, y_2, \ldots. y_q)(q \geqslant p)$; be the sequences of features. The goal is to find continuous indices from a^* to b^* of y, so that $y(a^*: b^*) = (y_{a^*}, y_{a^*+1}, \ldots., y_{b^*})$ with $1 \leq a^* \leq b^* \leq q$ that minimizes the DTW distance to x over all possible subsequences of y. In other words: $(a^*, b^*) = \underset{(a^*,b^*):1\leq a^*\leq b^*\leq q}{\text{argmin}} (DTW(x, y(a^*: b^*)))$. To achieve the SSTDW architecture from FSM, we modify the definition of the accumulated cost matrix $\mathfrak{P}(i, j)$ by setting $\mathfrak{P}(i, 1) = \sum_{p=1}^{i} \mathfrak{D}(x_p, y_1)$ and $\mathfrak{P}(1, j) = \sum_{q=1}^{j} \mathfrak{D}(x_1, y_q)$ for $j \in [1: q]$. The remaining values of $\mathfrak{P}_{i,j}$ are defined recursively for $i \in [2: p]$ and $j \in [2: q]$. The index b^* can be determined from the path cost matrix $\mathfrak{P}(i, j)$ by : $b^* = \underset{j\in[p:q]}{\arg\min} \mathfrak{P}(p, j)$. The index a^* is obtained by generating the optimal warping path between x and the subsequence $y(a^*: b^*)$ by back tracking process. The optimal warping path starts with $\mathfrak{p}_1 = (p, b^*)$ and ends at $\mathfrak{p}_L = (1, a^*)$; $a^* \in [1: q]$. Let $\mathfrak{p}^* = (\mathfrak{p}_1, \mathfrak{p}_2, \ldots., \mathfrak{p}_L)$ denote the resulting warping path. Then $p_L = (1, a^*)$; $a^* \in [1: q]$ is the last entry of $\mathfrak{p}^*$ array, which keeps the $(i, j)^{th}$ indexes of back tracked warping path. So, all the elements of y to the left of y_{a^*} and to the right of y_{b^*} are ignored for the alignment without any additional costs. The final dissimilarity measure $(\triangle(x, y))$ is calculated between x and the subsequence $y(a^*: b^*)$ of y; $b^* - a^* = p$;. For generating *sub-sequence DTW* architecture from FSM, we simply follow the same technique mentioned in Section E.1 by keeping into mind the aforementioned special constraint of *sub-sequence DTW*. The process of calculation of warping path is same as FSM and the rest of the part before the warping path calculation portion would remain same as in Algorithm 2. The final dissimilarity value: $\triangle(x, y)$ is normalized by dividing through the length of the warping path (same as in Algorithm 2).

Appendix F. Supplementary data

Supplementary data associated with this article can be found in the online version at http://dx.doi.org/10.1016/j.patcog.2016.05.011.

References

[1] H. Manmatha, R. Chengfeng, E. Riseman, Word spotting: a new approach to indexing handwriting, in: CVPR, IEEE Comput. Soc. Press, 1996, pp. 631–637.
[2] F. Perronnin, J.a. Rodriguez-Serrano, Fisher kernels for handwritten word-spotting, ICDAR (2009) 106–110.

[20] The implementation of CDP, used here is taken from: ⟨http://www.diva-portal.org/smash/get/diva2:347724/FULLTEXT01.pdf⟩. Page no. 86. Corresponding code can be found at: ⟨http://continuousdynamicprog.blogspot.in/⟩.

[3] A. Fischer, V. Frinken, H. Bunke, C.Y. Suen, Improving HMM-based keyword spotting with character language models, ICDAR (2013) 506–510.
[4] 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.
[5] N.R. Howe, Part-structured inkball models for one-shot handwritten word spotting, ICDAR (2013) 582–586.
[6] L. Rothacker, M. Rusinol, G.a. Fink, Bag-of-features hmms for segmentation-free word spotting in handwritten documents, ICDAR (2013) 1305–1309.
[7] M. Meshesha, C.V. Jawahar, Matching word images for content-based retrieval from printed document images, IJDAR 11 (1) (2008) 29–38.
[8] S. Srihari, H. Srinivasan, P. Babu, C. Bhole, Spotting words in handwritten Arabic documents, in: *Proc. SPIE* 6067, Document Recognition and Retrieval XIII, 2006, 606702.
[9] T. Adamek, N.E. O'Connor, A.F. Smeaton, Word matching using single closed contours for indexing handwritten historical documents, IJDAR 9 (2–4) (2006) 153–165.
[10] H. Cao, V. Govindaraju, Template-free word spotting in low-quality manuscripts, ICPR (2007) 1–5.
[11] J. Almazan, A. Gordo, A. Fornes, E. Valveny, Handwritten word spotting with corrected attributes, ICCV (2013) 1017–1024.
[12] P. Wang, V. Eglin, C. Garcia, C. Largeron, J. Llados, A. Fornes, A novel learning-free word spotting approach based on graph representation, in: Proceedings of the 11th IAPR International Workshop on Document Analysis Systems, IEEE, 2014, pp. 207–211.
[13] K. Terasawa,Y. Tanaka, H.O.G. Slit-Style feature for document image word spotting, ICDAR, 2009, 116–120.
[14] U. Roy, N. Sankaran, K.P. Sankar, C. Jawahar, Character N-gram spotting on handwritten documents using weakly-supervised segmentation, in: ICDAR, IEEE, 2013, pp. 577–581.
[15] Y. Leydier, F. Lebourgeois, H. Emptoz, Text search for medieval manuscript images, Pattern Recognit. 40 (12) (2007) 3552–3567.
[16] R.F. Moghaddam, M. Cheriet, Application of multi-level classifiers and clustering for automatic word spotting in historical document images, ICDAR (2009) 511–515.
[17] J. Almazán, A. Gordo, A. Fornés, E. Valveny, Efficient exemplar word spotting, in: Proceedings of the British Machine Vision Conference, 2012, pp. 67.1–67.11.
[18] N. Vasilopoulos, E. Kavallieratou, A Classification-free Word-Spotting System, in: *Proc. SPIE 8658*, Document Recognition and Retrieval XX, vol. 8658, 2013.
[19] X. Zhang, C.L. Tan, Segmentation-Free Keyword Spotting for Handwritten Documents Based on Heat Kernel Signature, in: ICDAR, IEEE, 2013, pp. 827–831.
[20] V. Dovgalecs, A. Burnett, P. Tranouez, S. Nicolas, L. Heutte, Spot It! Finding Words and Patterns in Historical Documents, in: ICDAR, IEEE, 2013, pp. 1039–1043.
[21] M. Rusiñol, D. Aldavert, R. Toledo, J. Lladós, Efficient segmentation-free keyword spotting in historical document collections, Pattern Recognit. 48 (2) (2015) 545–555.
[22] J. Chan, C. Ziftci, D. Forsyth, Searching Off-line Arabic Documents, in: CVPR, vol. 2, 2006, pp. 1455–1462.
[23] L.J. Latecki, V. Megalooikonomou, Q. Wang, D. Yu, An elastic partial shape matching technique, Pattern Recognit. 40 (11) (2007) 3069–3080.
[24] M. Müller, Information Retrieval for Music and Motion, Springer-Verlag, 2007, pp. 69–85.
[25] L.J. Latecki, Q. Wang, S. Koknar-Tezel, V. Megalooikonomou, Optimal subsequence bijection, ICDM (2007) 565–570.
[26] M. Vlachos, G. Kollios, D. Gunopulos, Discovering similar multidimensional trajectories, in: ICDE, IEEE Comput. Soc, 2002, pp. 673–684.
[27] T.M.R.R. Manmatha, Indexing of Handwritten Historical Documents – Recent Progress, Proceedings of Symposium on Document Image Understanding Technology (SDIUT), 2003, pp. 77–85.
[28] E.J. Keogh, M.J. Pazzani, Scaling up dynamic time warping for datamining applications, Proceedings of KDD, 2000, pp. 285–289.
[29] Y.S. Jeong, M.K. Jeong, O.A. Omitaomu, Weighted dynamic time warping for time series classification, in: Pattern Recognition, vol.44, Elsevier, 2011, pp. 2231–2240.
[30] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, E.J. Keogh, E.K. Michail Vlachos, Marios Hadjieleftheriou, Dimitrios Gunopulos, Indexing multi-dimensional time-series with support for multiple distance measures, in: KDD, ACM Press, New York, New York, USA, 2003, pp. 216–225.
[31] G. Das, D. Gunopulos, H. Mannila, Principles of Data Mining and Knowledge Discovery, vol. 1263 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 1997.
[32] B. Gatos, I. Pratikakis, S. Perantonis, Adaptive degraded document image binarization, PR 39 (3) (2006) 317–327.
[33] N. Stamatopoulos, B. Gatos, T. Georgiou, Page frame detection for double page document images, DAS, 2010, pp. 401–408.
[34] N. Nikolaou, M. Makridis, B. Gatos, N. Stamatopoulos, N. Papamarkos, Segmentation of historical machine-printed documents using Adaptive Run Length Smoothing and skeleton segmentation paths, Image Vis. Comput. 28 (4) (2010) 590–604.
[35] U. Marti, H. Bunke, Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition system, IJPRAI 15 (2001) 65–90.
[36] J.a. Rodríguez-Serrano, F. Perronnin, Handwritten word-spotting using hidden Markov models and universal vocabularies, Pattern Recognit. 42 (9) (2009) 2106–2116.
[37] A. Fischer, K. Riesen, H. Bunke, Graph Similarity Features for HMM-Based Handwriting Recognition in Historical Documents, in: ICFHR, IEEE, 2010, pp. 253–258.
[38] A. Fischer, A. Keller, V. Frinken, H. Bunke, Lexicon-free handwritten word spotting using character HMMs, Pattern Recognit. Lett. 33 (7) (2012) 934–942.
[39] R. Oka, Spotting method for classification of real world data, Comput. J. 41 (8) (1998) 1–6.

Tanmoy Mondal: 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 from Bengal Engineering and Science University, Kolkata, in 2009. Before joining as a PhD student in Laboratoiré d'Informatique, Université François Rabelais, Tours, France in 2012, he has worked at several industries and scientific laboratories as a researcher. He has received his PhD degree in computer science in 2015. 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: 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 associate professor at Polytech 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 is group were also involved in several National projects funded by government (ANR NAVIDOMAS, DIGIDOC) as well as companies (ATOS Worldline, Nexter). His group also received during 2 years a 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: Jean-Yves RAMEL received his PhD degree in Computer Sciences in 1996 from the National Institute of Applied Sciences of Lyon (INSA Lyon France). After being an Assistant Professor at the Industrial Engineering Department of the National Institute of Applied Sciences of Lyon from 1998 to 2002; Jean-Yves Ramel is currently a Full Professor at the Computer Science Department of Polytech Tours (French engineering school). He is also a researcher of the Computer Science Laboratory of Tours (EA 6300) where he is managing the Image Analysis and Pattern Recognition Group. His current research fields are interactive methods for image analysis and classification and structural pattern recognition. Jean-Yves RAMEL is an active member of the Pattern Recognition and Image Analysis French and International communities (AFRIF, GRCE, IAPR).

Umapada Pal: 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 300 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 2005–2006 Dr. Pal has received JSPS fellowship from Japan government. 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. Also he has served as a guest editor of several special issues. He is a Fellow of IAPR (International Association of Pattern Recognition).