Research Highlights (Required)

- Graph-based keyword spotting method using Hausdorff edit distance.
- Flexible: matching arbitrary graphs with any label alphabets.
- Efficient: quadratic time complexity with respect to graph size.
- Effective: outperforms other template-based spotting methods.
- Evaluation on four benchmark datasets including recent competition.

Graph-Based Keyword Spotting in Historical Manuscripts Using Hausdorff Edit Distance

Mohammad Reza Ameri[a,**], Michael Stauffer[b,c], Kaspar Riesen[b], Tien D. Bui[a], Andreas Fischer[d,e]

[a] *Concordia University, Computer Science and Software Engineering Department, 1455 de Maisonneuve Blvd West, Montréal,H3G 1M8, Canada*
[b] *University of Applied Sciences and Arts Northwestern Switzerland, Institute for Information Systems, 4600 Olten, Switzerland*
[c] *University of Pretoria, Department of Informatics, 0083 Pretoria, South Africa*
[d] *University of Fribourg, Department of Informatics, 1700 Fribourg, Switzerland*
[e] *University of Applied Sciences and Arts Western Switzerland, Institute for Complex Systems, 1705 Fribourg, Switzerland*

Abstract

Keyword spotting enables content-based retrieval of scanned historical manuscripts using search terms, which, in turn, facilitates the indexation in digital libraries. Recent approaches include graph-based representations that capture the complex structure of handwriting. However, the high representational power of graphs comes at the cost of high computational complexity for graph matching. In this article, we investigate the potential of Hausdorff edit distance (HED) for keyword spotting. It is an efficient quadratic-time approximation of the graph edit distance. In a comprehensive experimental evaluation with four types of handwriting graphs and four benchmark datasets (George Washington, Parzival, Botany, and Alvermann Konzilsprotokolle), we demonstrate a strong performance of the proposed HED-based method when compared with the state of the art, both, in terms of precision and speed.

1. Introduction

In recent years, there have been increased efforts worldwide by libraries and archives to digitize handwritten historical documents. For content-based searching and browsing in digital libraries, automatic handwriting recognition systems are needed. However, when compared to printed text, modeling and recognition of handwriting is far more challenging taking into account variable character shapes. When facing ancient scripts and languages with a lack of training data, automatic transcription is often not feasible. In such a situation, keyword spotting has been proposed as an alternative to index scanned manuscripts without performing a complete transcription (Manmatha et al., 1996).

Two general approaches to keyword spotting can be distinguished, viz. *template-based* and *learning-based* methods. While template-based methods match one or several instances of a keyword image directly with the scanned manuscript, learning-based methods aim to learn word or subword models from labeled training samples. Examples include learning with hidden Markov models (HMM) (Perronnin and Rodríguez-Serrano, 2009; Fischer et al., 2012; Rothacker and Fink, 2015), support vector machines (SVM) (Almazan et al., 2014), recurrent neural networks (RNN) (Frinken et al., 2012), and convolutional neural networks (CNN) (Sudholt and Fink, 2016). In general, learning-based methods are able to achieve a significantly better performance than template-based methods. However, they are less flexible because they require a considerable amount of labeled training data, similar to transcription systems.

In this article, we focus on template-based methods instead, which do not require any learning but can be applied even if only a single template image of the keyword is provided to the spotting system. This is particularly useful for historical manuscripts, which typically require human experts for obtaining labeled training data in a time-consuming and costly process.

Early approaches to template-based KWS include pixel-by-pixel matchings of word images (Manmatha et al., 1996) using affine transformations with the Scott and Longuet-Higgins algorithm (Scott and Longuet-Higgins, 1991). Later on, different feature descriptors have been investigated, including projection profiles (Rath and Manmatha, 2007), histograms of oriented gradients (HOG) (Rodríguez-Serrano and Perronnin, 2008; Terasawa and Tanaka, 2009; Rusiñol et al., 2015), and features extracted from unlabeled data by deep neural networks (Wicht et al., 2016), to name just a few. For coping with the variable

**Corresponding author: Tel.:+1-514-848-2424x7248; fax: +1-514-848-2830;
e-mail: mo_amer@encs.concordia.ca (Mohammad Reza Ameri)

width of the handwriting, a widely adopted approach is to use a sliding window for extracting a sequence of feature vectors from word images and match two sequences by means of dynamic time warping (DTW) (Rath and Manmatha, 2007). To avoid an explicit segmentation of the scanned document page into word images, segmentation-free methods have been proposed as well (Rusiñol et al., 2015).

A general limitation of feature vector descriptors is that they have to capture the complex structure of handwriting with a fixed number of real-valued features. Furthermore, they cannot represent binary relations between parts of the handwriting in a straight-forward way. Both limitations can be overcome by means of graph-based representations, which model parts of an object with nodes and relations between the parts with edges (Conte et al., 2004). In recent work, several graph-based methods have been proposed in the context of template-based keyword spotting, using keypoints as nodes (Howe, 2013; Wang et al., 2014b,a; Stauffer et al., 2016b) or basic strokes as nodes (Bui et al., 2015; Riba et al., 2015), and connecting them with edges if there is a connection in the image.

The main drawback of graphs, however, is that the high representational power comes at the cost of high computation complexity. Most of the aforementioned methods for graph-based keyword spotting are based on the well-known bipartite approximation (BP) (Riesen and Bunke, 2009) of graph edit distance (GED) (Bunke and Allermann, 1983). Although BP reduces the NP-complete problem of GED to a polyomial-time assignment problem, it still has a cubic time complexity with respect to the graph size, which imposes significant computational constraints for keyword spotting.

In this article, we investigate the potential of a recently introduced, more efficient approximation of GED for keyword spotting, namely the Hausdorff edit distance (HED) (Fischer et al., 2015). It has a quadratic time complexity with respect to the graph size, similar to DTW, which has a quadratic time complexity with respect to the sequence length. Unlike DTW, HED is not constrained to sequence matching. Instead it is able to match arbitrary handwriting graphs without constraints on the graph structure and the label alphabets for nodes and edges.

A preliminary version of this article has been published as an extended abstract in the proceedings of the 18th International Graphonomics Society Conference (IGS2017) (Ameri et al., 2017). The present article substantially extends the conference paper with a more detailed description and discussion of the method, a more comprehensive experimental evaluation with three additional benchmark datasets, a study on the combination of HED and DTW, and an extended comparison with the current state of the art.

In the remainder, we first describe four graph-based handwriting representations in Section 2, introduce the HED-based keyword spotting system in Section 3, present our experimental evaluation on four benchmark datasets in Section 4, and conclude the article in Section 5 with an outlook to future lines of research.

2. Handwriting Graphs

The proposed template-based KWS approaches makes use of graphs for the representation and retrieval of word images as illustrated in Fig. 1. In the first step, handwritten document images are filtered, binarized and segmented into word images (detailed in Sect. 2.1). Based on single word images, graphs are represented by means of four different representations (see Sect. 2.2). Next, graphs are normalized by a z-score to minimize intrapersonal variations (see Sect. 2.3). Finally, the actual keyword spotting is performed by computing all graph dissimilarities (see Sect. 3.2 and 3.1) between a certain query graph q and all document graphs $g \in G$ to form a retrieval index (see Sect. 3.3).

In the following sections, the first three steps are described in greater detail, while the actual graph-based KWS is explained in Sect. 3.

2.1. Image Preprocessing

To remove noise, a *Difference of Gaussians* filtering is first applied to scanned document images (Fischer et al., 2010). Next, filtered document images are binarized by means of global thresholding. As the present KWS framework operates on segmented word images, document images are first automatically segmented into single word images by means of vertical projection profiles of a line of text. If necessary, the segmentation result is manually corrected. That is, segmentation errors are neglected and the accuracy can be seen as an upper-bound. Moreover, the skew, i.e. the inclination of the document, is estimated by means of the gradient of the lower baseline of a line of text and then corrected on single word images (Marti and Bunke, 2001). Optionally, word images are skeletonized by means of 3×3 thinning operator (Guo and Hall, 1989). The filtered and binarized word images are denoted by B, while skeletonized word images are denoted by S from now on.

2.2. Graph Extraction

Generally, a graph g is defined as a four-tuple $g = (V, E, \mu, \nu)$ where V and E are finite sets of nodes and edges, and $\mu : V \rightarrow L_V$ as well as $\nu : E \rightarrow L_E$ are labelling functions for nodes and edges, respectively. Graphs can be divided into *undirected* and *directed* graphs, where pairs of nodes are either connected by undirected or directed edges, respectively. Additionally, graphs are often distinguished into *unlabeled* and *labeled* graphs. In the latter case, both nodes and edges can be labeled with an arbitrary numerical, vectorial, or symbolic label from L_V or L_E, respectively. In the former case we assume empty label alphabets, i.e. $L_V = L_E = \{\}$.

Subsequently fourr different graph extraction methods are briefly explained and introduced, for further details we refer to (Stauffer et al., 2016a). All graph extraction methods result in nodes that are labeled with two-dimensional numerical labels, while edges remain unlabeled, i.e. $L_V = \mathbb{R}^2$ and $L_E = \{\}$.

- `Keypoint`: The first graph extraction algorithm makes use of characteristics points (so called keypoints) in skeletonized word images S. These keypoints are represented

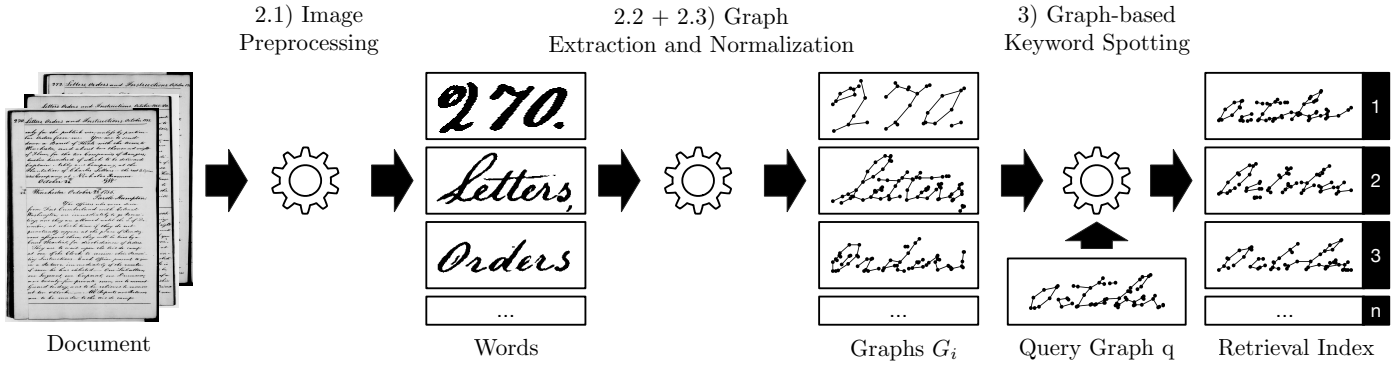

Fig. 1. Process of Graph-based Keyword Spotting of the Word "October"

as nodes that are labeled with the corresponding (x, y)-coordinates. Between pairs of keypoints (which are connected on the skeleton) further intermediate points are converted to nodes and added to the graph at equidistant intervals. Finally, undirected edges are inserted into the graph for each pair of nodes that is directly connected by a stroke.

- `Grid`: The second graph extraction algorithm is based on a grid-wise segmentation of binarised word images B into equally sized segments. For each segment, a node is inserted into the graph and labeled by the (x, y)-coordinates of its respective center of mass. Undirected edges are inserted between two neighboring segments that are actually represented by a node. Finally, the inserted edges are reduced by means of a *Minimal Spanning Tree* algorithm.

- `Projection`: The next graph extraction algorithm works on an adaptive and threshold-based segmentation of binarised word images B. Basically, this segmentation is computed on the horizontal and vertical projection profiles of B. The resulting segmentation is further refined in the horizontal and vertical direction by means of two distance-based thresholds. A node is inserted into the graph for each segment and labeled by the (x, y)-coordinates of the corresponding center of mass. Undirected edges are inserted into the graph for each pair of nodes that is directly connected by a stroke in the original word image.

- `Split`: The fourth graph extraction algorithm is based on an iterative segmentation of binarised word images B. That is, segments are iteratively split into smaller subsegments until the width and height of all segments are below certain thresholds. A node is inserted into the graph and labeled by the (x, y)-coordinates of the point on the stroke closest to the center of mass of each segment. For the insertion of the edges, the same procedure as for `Projection` is applied.

2.3. Graph Normalization

To mitigate the influence of intrapersonal writing variations, the resulting set of graphs is normalized with respect to the (x, y)-coordinates of their node labels $\mu(v)$. Formally, we use a z-score to derive normalized coordinates $(\hat{x}, \hat{y})$ by

$$\hat{x} = \frac{x - \mu_x}{\sigma_x} \text{ and } \hat{y} = \frac{y - \mu_y}{\sigma_y} \quad ,$$

where (μ_x, μ_y) and (σ_x, σ_y) represent the mean and standard deviation of all (x, y)-coordinates in the graph under consideration.

3. Graph-Based Keyword Spotting

For spotting keywords, a query graph q (used to represent a certain keyword) is pairwise matched against all document graphs $G = \{g_1, \ldots, g_N\}$. Generally, graphs can either be matched by means of exact or inexact approaches (Conte et al., 2004; Foggia et al., 2014; Riesen, 2015). In case of graph-based KWS, graphs are used to represent the inherent characteristic of handwriting, and thus, affected by (subtle) variations in both their structure and labels. For this reason, inexact graph matching can be applied only.

3.1. Graph Edit Distance

Several approaches have been proposed for inexact graph matching (Conte et al., 2004; Foggia et al., 2014; Riesen, 2015). Yet, graph edit distance (GED) is regarded as one of the most flexible and powerful paradigms (Bunke and Allermann, 1983). In particular, GED measures the amount of distortion needed to transform graph g_1 into graph g_2 using a sequence of edit operations like *insertions*, *deletions*, and *substitutions* of both nodes and edges (called *edit path* $\lambda(g_1, g_2)$ between g_1 and g_2).

To find the most suitable edit path, one commonly introduces a certain cost function $c(e)$ for every edit operation e. This cost function should correspond to the strength of a certain graph modification. Formally, the graph edit distance $d_{\text{GED}}(g_1, g_2)$, or d_{GED} for short, between g_1 and g_2 is given by

$$d_{\text{GED}}(g_1, g_2) = \min_{\lambda \in \Upsilon(g_1, g_2)} \sum_{e_i \in \lambda} c(e_i) \quad ,$$

where $\Upsilon(g_1, g_2)$ is the set of all edit paths between g_1 and g_2.

However, the exact computation of d_{GED} is exponential with respect to the number of nodes of the involved graphs. Formally, GED is an instance of a *Quadratic Assignment Problems (QAPs)* (Koopmans and Beckmann, 1957), which in turn belongs to the class of $\mathcal{NP}$-complete problems[1]. Hence, several

[1] That is, an exact and efficient algorithm for the graph edit distance problem can not be developed unless $\mathcal{P} = \mathcal{NP}$.

fast but suboptimal algorithms have been proposed in the last years (see (Foggia et al., 2014)).

In (Riesen and Bunke, 2009), the bipartite GED (commonly defined as BP) has been introduced. This algorithm with cubic time complexity reduces the QAP of GED to a *Linear Sum Assignment Problem* (*LSAP*) that can be optimally solved in cubic time (see (Burkard et al., 2009) for an exhaustive survey on LSAP solving algorithms). The optimal LSAP solution is eventually used to derive a suboptimal GED.

For the representation of domain knowledge, one commonly makes use of a certain cost model. In the current case, constant costs for both node and edge deletions/insertions are defined, i.e. $\tau_v \in \mathbb{R}^+$ and $\tau_e \in \mathbb{R}^+$, respectively. For the substation of nodes, we make use of a weighted Euclidean distance between the corresponding node labels, i.e. (x, y)-coordinates. Formally,

$$\sqrt{\alpha\,\sigma_x(x_i - x_j)^2 + (1-\alpha)\,\sigma_y(y_i - y_j)^2} \quad ,$$

where $\alpha \in [0, 1]$ denotes a parameter to weight the importance of the x- and y-coordinate of a node, while σ_x and σ_y denote the standard deviation of all node coordinates in the current query graph. Moreover, we make use of a weighting factor $\beta \in [0, 1]$ between the sum of all node and edge edit costs.

3.2. Hausdorff Edit Distance

Andreas: eine Spalte

3.3. Keyword Spotting Score

For building the KWS score, the graph edit distances d_{GED} between query graph q and all document graphs $G = \{g_1, \ldots, g_N\}$ is first normalized by the sum of the maximum cost edit path between q and g_i, i.e. the sum of the edit path that results from deleting all nodes and edges of q and inserting all nodes and edges in g_i. Formally,

$$r(q, g) = -\frac{d_{\mathrm{GED}}(q, g_i)}{(|V_q| + |V_{g_i}|)\,\tau_v + (|E_q| + |E_{g_i}|)\,\tau_e} \quad ,$$

where d_{GED} is either based on BP, BP2, or HED. If a query consists of a set of graphs $\{q_1, \ldots, q_t\}$ that represent the same keyword, the normalised graph edit distance is given by the minimal distance achieved on all t query graphs.

4. Experimental Evaluation

We evaluate the proposed HED-based method on four benchmark datasets for keyword spotting in historical manuscripts, which are described in Section 4.1. In several experiments, its performance is compared with other template-based reference methods, namely BP, BP2, and DTW, which are detailed in Section 4.2. Finally, we also put our method into context with learning-based approaches to keyword spotting.

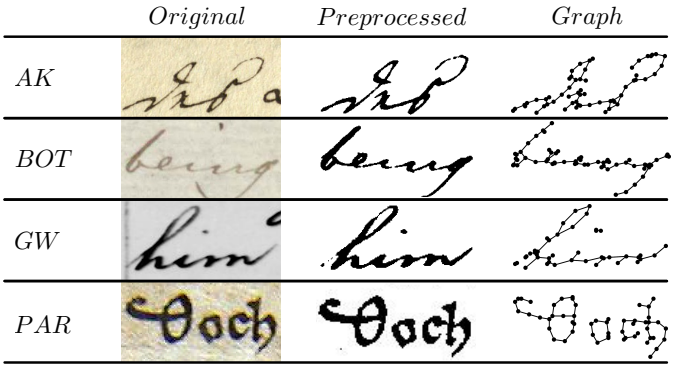

Fig. 2. Exemplary graph representations of the Alvermann Konzilsprotokolle (AK), Botany (BOT), George Washington (GW), and Parzival (PAR) dataset.

4.1. Datasets

For the experimental evaluation, we consider two well known manuscripts, viz. *George Washington (GW)*[2] and *Parzival (PAR)*[3], as well as two documents of a very recent KWS benchmark competition[4], viz. *Alvermann Konzilsprotokolle (AK)* and *Botany (BOT)*. GW consists of letters of George Washington and his associates during the American Revolutionary War in 1755. The letters are written in English and based on twenty pages with minor variations in writing and degradation. PAR is based on stories of the German poet Wolfgang von Eschenbach in the 13th century. The manuscript is written in Middle High German and based on 45 pages with low writing variations but markable signs of degradation. AK consists of minutes of formal meetings held by the central administration of the University of Greifswald in the period of 1794 to 1797. The notes are written in German and based on 18,000 pages with minor variations and signs of degradation. Finally, BOT is based on botanical records made in British India in the 18th and 19th century. The records are written in English and based on ten pages with high writing variation and markable signs of degradation.

On all four manuscripts, graphs are extracted by means of the graph representation formalisms proposed in Section 2. Note that for AK and BOT, only the two most promising graph representations, i.e. `Keypoint` and `Projection`, are considered. In Fig. 2, an exemplary word of each manuscript and the corresponding graph representation for `Keypoint` is given.

4.2. Reference Methods

In order to assess the potential of the proposed HED-based graph matching approach, we compare it with three related reference methods for matching graphs (BP and BP2) and sequences (DTW), respectively.

[2] George Washington Papers at the Library of Congress, 1741-1799: Series 2, Letterbook 1, pp. 270-279 & 300-309, `http://memory.loc.gov/ammem/gwhtml/gwseries2.html`

[3] Parzival at IAM historical document database, `http://www.fki.inf.unibe.ch/databases/iam-historical-document-database/parzival-database`

[4] Alvermann Konzilsprotokolle and Botany at ICFHR2016 benchmark database, `http://www.prhlt.upv.es/contests/icfhr2016-kws/data.html`

BP. First, the bipartite graph matching method (BP) proposed by Riesen and Bunke (2009) for approximating graph edit distance. It is widely used for graph-based pattern recognition (Stauffer et al., 2017) and has also been considered in a number of graph-based keyword spotting systems, including (Wang et al., 2014b; Bui et al., 2015; Riba et al., 2015; Stauffer et al., 2016b). BP reduces the problem of graph edit distance to a linear sum assignment problem (LSAP) and returns a valid – but not necessarily optimal – edit path between two graphs. The cost of this edit path, which is an upper bound of graph edit distance, can then be used to compute a keyword spotting score. The main constraint of BP is its cubic time complexity with respect to the graph size, which imposes computational limits regarding the size of the handwriting graphs as well as the number of handwriting graphs that can be matched.

BP2. Secondly, the recently introduced quadratic time variant of BP called BP2 (Fischer et al., 2017). It solves the bipartite matching problem in quadratic time and returns, similar to BP, a valid edit path between two graphs and thus an upper bound of graph edit distance.

DTW. Thirdly, the well-established Dynamic Time Warping (DTW) method for sequence matching, which is often used for keyword spotting in historical manuscripts (Rath and Manmatha, 2007; Terasawa and Tanaka, 2009; Frinken et al., 2012; Wicht et al., 2016). By moving a sliding window over the handwriting, a sequence of feature vectors is extracted. DTW finds an optimal alignment of two sequences along a common time axis such that the sum of feature vector distances is minimum. This sum of distances can then be used to compute a keyword spotting score. Note that sequences are a special case of graphs, in which the nodes are ordered and have at most one successor. Using dynamic programming, an optimal DTW alignment can be obtained in quadratic time with respect to the sequence length.

4.3. Experimental Setup

On all benchmark datasets, individual word images are considered for experimental evaluation. The word segmentation is manually corrected, hence the results obtained on these benchmarks can be seen as an upper bound on the spotting performance. In a real-world scenario, errors stemming from automatic word segmentation may decrease the end-to-end performance.

Experiments are conducted in two stages. First, during the validation stage, several system parameters are fine-tuned on a small validation set, which consists of 10 random instances of 10 manually selected keywords (with different word lengths) and 900 additional, randomly selected words (1,000 words in total). Secondly, during the testing stage, the optimized system is evaluated on the same training and test sets as used in (Fischer et al., 2012) for GW and PAR and (Pratikakis et al., 2016) for AK and BOT. All templates of a keyword present in the training set are used for keyword spotting. In Table 1 a summary of the datasets is presented.

To evaluate the keyword spotting performance, we consider Recall and Precision for each keyword query and compute the Mean Average Precision (MAP) over all queries using the `trec_eval`[5] software.

Table 1. Number of keywords and number of word images in the training and test sets of the four datasets.

Dataset	Keywords	Train	Test
GW	105	2,447	1,224
PAR	1,217	11,468	6,869
BOT	150	1,684	3,380
AK	200	1,849	3,734

Table 2. Mean average precision (MAP) for graph-based KWS systems on the George Washington (GW) and Parzival (PAR) datasets.

		GW		PAR	
	Method	MAP	±	MAP	±
BP	`Keypoint`	66.08		62.04	
	`Grid`	60.02		56.50	
	`Projection`	61.43		66.23	
	`Split`	60.23		59.44	
BP2	`Keypoint`	68.42	+2.33	55.03	−7.01
	`Grid`	62.10	+2.07	57.00	+0.50
	`Projection`	60.83	−0.60	63.35	−2.88
	`Split`	64.24	+4.02	68.69	+9.25
HED	`Keypoint`	69.28	+3.19	69.23	+7.19
	`Grid`	62.78	+2.75	60.74	+4.24
	`Projection`	66.71	+5.28	72.82	+6.59
	`Split`	65.12	+4.89	72.79	+13.35

4.4. Comparison with Graph Edit Distance Approximations

In the first experiment, we compare HED with other approximation methods of graph edit distance, namely BP and BP2. All three methods can be applied to any type of graph, without constraints on the graph structure or the node and edge label alphabets. The approximate graph edit distance is divided by the maximum graph edit distance to derive a normalized keyword spotting score, as described in Section 3.3.

We consider the four graph-based handwriting representations discussed in Section 2 and adopt optimal graph parameters from previous work (Stauffer et al., 2016a,b). Parameters of the keyword spotting system include the cost for node deletion/insertion τ_n and the cost for edge deletion/insertion τ_e. They are optimized over the range of $\tau_n, \tau_e \in \{1, 4, 8, 16, 32\}$ for each method individually on the validation set.

Table 2 presents the MAP results on the test set of GW and PAR for the three methods and the four graph representations. Confirming the observations in (Fischer et al., 2017), BP2 performs very similar to BP, outperforming BP in five out of eight cases. These results indicate that, in this scenario, the quadratic-time BP2 method is not only significantly more efficient than the cubic-time BP method but it can also achieve similar performance.

HED achieves the best results, outperforming BP in eight out of eight cases. Hence, it not only allows to reduce the computational complexity but also improves the keyword spotting performance. Unlike BP and BP2, HED allows multiple assignments among substructures in the handwriting graphs. We assume that this property of HED is beneficial in the context

[5] `http://trec.nist.gov/trec_eval`

Table 3. Mean average precision (MAP) for graph-based KWS systems on the Botany (BOT) and Alvermann Konzilsprotokolle (AK) datasets.

		BOT		AK	
	Method	MAP	±	MAP	±
BP	Keypoint	45.06		77.24	
	Projection	49.57		76.02	
BP2	Keypoint	50.94	+5.88	74.86	−2.38
	Projection	50.49	+0.92	75.46	−0.56
HED	Keypoint	51.74	+6.68	79.72	+2.48
	Projection	51.69	+2.12	81.06	+5.04

Table 4. Median and maximum number of nodes, mean runtime per graph pair in milliseconds for BP and HED, and speedup factor on the George Washington (GW) dataset.

Method	$\|V\|_{med}$	$\|V\|_{max}$	T_{BP}	T_{HED}	Speedup
Keypoint	74	366	303.0	3.2	95.3
Grid	90	509	707.9	6.1	116.0
Projection	74	391	344.1	3.9	88.1
Split	80	434	480.2	4.4	108.1

Table 5. Mean average precision (MAP) for graph-based KWS systems in comparison with three template-based reference systems on the George Washington (GW) and Parzival (PAR) dataset. The first, second, and third best systems are indicated by (1), (2), and (3).

Method		GW		PAR		Average	
Reference (Template)	DTW'08	63.39		47.52		55.46	
	DTW'09	64.80		73.49	(1)	69.15	(3)
	DTW'16	68.64	(2)	72.38	(3)	70.51	(2)
Graph (Template)	BP	66.08		66.23		66.16	
	BP2	68.42	(3)	68.69		68.55	
	HED	69.28	(1)	72.82	(2)	71.05	(1)

Table 6. Mean average precision (MAP) for the combination of DTW and HED on the George Washington (GW) dataset.

Method		MAP	±
Individual	DTW	64.00	
	HED	69.28	
Combined	DTW+HED	77.83	+8.55

of handwriting because it allows a kind of "warping" between characters of different size, similar to DTW (see Section 4.2) but in two dimensions rather than only one.

The results shown in Table 3 for the two other datasets, BOT and AK, confirm the findings. On these datasets, HED outperforms BP in four out of four cases.

Finally, Table 4 reports the speedup that can be achieved with the quadratic-time HED method when compared to the cubic-time BP method. On the GW dataset, the handwriting graphs have a median size between 74 and 90 and a maximum size between 366 and 509. For this graph size, HED-based keyword spotting is about hundred times faster than BP-based keyword spotting.

4.5. Comparison with Dynamic Time Warping

Table 5 shows a comparison with the state of the art for template-based keyword spotting using DTW. Three reference methods are considered for the GW and PAR benchmark datasets. DTW'08 (Rodríguez-Serrano and Perronnin, 2008) and DTW'09 (Terasawa and Tanaka, 2009) employ SIFT-like gradient features, while DTW'16 (Wicht et al., 2016) is based on convolutional neural network (CNN) features that are extracted from the datasets without supervision (without labeled training data) using deep belief networks. All results are taken from Wicht et al. (2016). For HED, we show the results for the best performing graph representations found in Section 4.4, which in most cases is Keypoint.

The results indicate that the template-based keyword spotting methods achieve performance results in the same ballpark. DTW'09 and DTW'16 tend to outperform BP and BP2, while HED achieves the overall best results on these benchmarks.

The strong performance of HED is rather astonishing when comparing the sophisticated CNN features of DTW'16 with the relatively simple coordinate labels used for the handwriting graphs. It underlines the representational power of graphs for capturing relevant structures of the handwriting.

Regarding runtime, HED has a quadratic time complexity with respect to the graph size and DTW has a quadratic time complexity with respect to the sequence length. In our experimental setting, the graph size is typically smaller than the sequence length. On the GW dataset, for example, the median graph size is 74, while the median sequence length is 134. In this scenario, HED also reduces the computational effort when compared with DTW.

4.6. Combination of HED and Dynamic Time Warping

In the next experiment, we investigate the potential of combining HED and DTW. Since the two methods are quite different, one matching two-dimensional graphs and the other matching one-dimensional sequences, they have complementary properties and thus a high potential to support each other in a multiple classifier system (MCS). In such an MCS setting, ideally, one method is able to correct errors of the other method (Kuncheva, 2004).

We have implemented our own DTW reference method, following the general ideas of Rath and Manmatha (2007) and using the features proposed by Marti and Bunke (2001). Image preprocessing includes skew and slant correction as well as height and width normalization. Afterwards, a sliding window of one pixel width extracts a sequence of nine geometric features. They are aligned by means of DTW using a Sakoe-Chiba band (Sakoe and Chiba, 1978) with a width of Ω percent to speedup the alignment and to exclude unusual warping paths. The parameter Ω is optimized on the validation set over a range of $\Omega \in \{0.20, 0.25, \ldots, 0.70\}$. The resulting cost of the warping path is normalized with the length of the warping path to obtain a keyword spotting score. This DTW system achieves a MAP of 64.00 on GW, which is comparable with the other reference methods listed in Table 5.

After normalizing the HED and the DTW scores to zero mean and unit standard deviation, they are combined with a weighted sum $hed + \omega \cdot dtw$. The weight ω is optimized on the validation set over a range of $\omega \in \{0.1, 0.2, \ldots, 2.0\}$.

Table 6 reports the combination result on the GW test set. Although DTW has a lower performance than HED, the combination leads to a significant increase in MAP by 8.55%, emphasizing the complementary properties of the two methods.

4.7. Comparison with Learning-Based Keyword Spotting

Table 7. Mean average precision (MAP) for graph-based KWS systems in comparison with three state-of-the-art learning-based reference systems on the Alvermann Konzilsprotokolle (AK) and Botany (BOT) datasets. The first, second, and third best systems are indicated by (1), (2), and (3).

Method		BOT		AK		Average	
Reference (Learning)	CVCDAG	75.77	(2)	77.91		76.84	(2)
	PRG	89.69	(1)	96.05	(1)	92.87	(1)
	QTOB	54.95	(3)	82.15	(2)	68.55	(3)
Graph (Template)	BP	49.57		77.24		63.41	
	BP2	50.94		75.46		63.20	
	HED	51.74		81.06	(3)	66.40	

Our proposed HED method follows the template-based approach to keyword spotting, which has minimum requirements regarding human interaction. Even if only a single template image of the keyword is provided to the system, it can search for it in a collection of scanned documents without requiring a human to annotate part of the collection. The low requirements of template-based keyword spotting are especially useful in the context of historical manuscripts, where obtaining labeled training data often requires human experts and thus becomes time-consuming and costly.

However, if labeled training data can be made available to the system, learning-based approaches can profit from this knowledge and build more robust spotting systems. In Table 7, we compare our proposed template-based method with recent learning-based methods from the ICFHR2016 competition (Pratikakis et al., 2016), viz. CVCDAG (Almazan et al., 2014), PRG (Sudholt and Fink, 2016), and QTOB. CVCDAG is based on Pyramidal Histogram Of Characters (PHOC) features in conjunction with an SVM, PRG is based on the same features in conjunction with a Convolutional Neural Network (CNN), called PHOCNet, and QTOB is based on another CNN following a triplet network approach.

As expected, the learning-based methods achieve a higher performance in general and especially PRG significantly outperforms the proposed HED-based method. Nevertheless, it is interesting to observe that HED can keep up with the performance of QTOB and outperforms CVCDAG in one out of three cases, despite the fact that no learning has been performed for HED. This observation demonstrates the high potential of HED as a template-based keyword spotting method.

Note that template-based and learning-based methods have complementary properties and can be used together in the ditialization process of historical manuscripts. At the beginning, when no labeled data is available, template-based method can be used to cluster similar words that are then labeled conjointly and efficiently by a human expert. As soon as enough training samples become available, learning-based methods can be trained to perform a more accurate search. Finally, when enough labeled data is available to train robust character models, a full transcription can be attempted together with a word dictionary (Frinken et al., 2014).

5. Conclusion and Outlook

The HED-based keyword spotting approach presented in this article has demonstrated several promising properties. First, it approximates the graph edit distance and hence is *flexible* in the sense that it allows to represent handwriting with any type of graph, without constraints on the graph structure or the label alphabets for nodes and edges. Secondly, it can be computed in quadratic time with respect to the graph size and hence is *efficient* for matching large graphs and large numbers of graphs. Thirdly, the experimental evaluation on four benchmark datasets for keyword spotting in historical manuscripts have demonstrated that it is *effective* in terms of mean average precision and compares favourably with other template-based keyword spotting systems.

Unlike dynamic time warping, which considers handwriting as a sequence of feature vectors, HED considers the two-dimensional, global structure of the handwriting. The two perspectives are different and complementary, which could be demonstrated by combining the two methods into a multiple classifier system that outperformed the individual methods.

There are several promising lines of future research. First, the graph-based representations considered so far are rather close to the actual shape of the handwriting, representing parts of the ink with nodes that are labeled with coordinates and connecting them with edges when they are connected by the ink. It would be interesting to investigate other, potentially more abstract graph-based representations of handwriting. Secondly, the potential of multiple classifier system could be demonstrated. It may be rewarding to combine the HED spotting scores of different graph-based handwriting representations in order to improve the robustness of the spotting system. Finally, if labeled training data is available, an intriguing open question is how to perform machine learning on graph-based representations and graph matching in order to profit from the labeled data.

References

Almazan, J., Gordo, A., Fornes, A., Valveny, E., 2014. Word Spotting and Recognition with Embedded Attributes. IEEE Transactions on Pattern Analysis and Machine Intelligence 36, 2552–2566.

Ameri, M., Stauffer, M., Riesen, K., Bui, T., Fischer, A., 2017. Keyword Spotting in Historical Documents Based on Handwriting Graphs and Hausdorff Edit Distance, in: International Graphonomics Society Conference, pp. 105–108.

Bui, Q.A., Visani, M., Mullot, R., 2015. Unsupervised word spotting using a graph representation based on invariants, in: International Conference on Document Analysis and Recognition, pp. 616–620.

Bunke, H., Allermann, G., 1983. Inexact graph matching for structural pattern recognition. Pattern Recognition Letters 1, 245–253.

Burkard, R., Dell'Amico, M., Martello, S., 2009. Assignment Problems. Society for Industrial and Applied Mathematics. URL: http://epubs.siam.org/doi/book/10.1137/1.9781611972238, doi:10.1137/1.9781611972238.

Conte, D., Foggia, P., Sansone, C., Vento, M., 2004. Thirty Years Of Graph Matching In Pattern Recognition. International Journal of Pattern Recognition and Artificial Intelligence 18, 265–298.

Fischer, A., Indermühle, E., Bunke, H., Viehhauser, G., Stolz, M., 2010. Ground truth creation for handwriting recognition in historical documents, in: International Workshop on Document Analysis Systems, New York, New York, USA. pp. 3–10. URL: http://dl.acm.org/citation.cfm?id=1815330.1815331, doi:10.1145/1815330.1815331.

Fischer, A., Keller, A., Frinken, V., Bunke, H., 2012. Lexicon-free handwritten word spotting using character HMMs. Pattern Recognition Letters 33, 934–942.

Fischer, A., Riesen, K., Bunke, H., 2017. Improved quadratic time approximation of graph edit distance by combining Hausdorff matching and greedy assignment. Pattern Recognition Letters 87, 55–62.

Fischer, A., Suen, C.Y., Frinken, V., Riesen, K., Bunke, H., 2015. Approximation

of graph edit distance based on Hausdorff matching. Pattern Recognition 48, 331–343.

Foggia, P., Percannella, G., Vento, M., 2014. Graph Matching and Learning in Pattern Recognition in the last 10 Years. International Journal of Pattern Recognition and Artificial Intelligence 28, 1450001. URL: http://www.worldscientific.com/doi/abs/10.1142/S0218001414500013, doi:10.1142/S0218001414500013.

Frinken, V., Fischer, A., Baumgartner, M., Bunke, H., 2014. Keyword spotting for self-training of BLSTM NN based handwriting recognition systems, in: Pattern Recognition, pp. 1073–1082.

Frinken, V., Fischer, A., Manmatha, R., Bunke, H., 2012. A novel word spotting method based on recurrent neural networks. IEEE Transactions on Pattern Analysis and Machine Intelligence 34, 211–224.

Guo, Z., Hall, R.W., 1989. Parallel thinning with two-subiteration algorithms. Communications of the ACM 32, 359–373. URL: http://dl.acm.org/citation.cfm?id=62065.62074, doi:10.1145/62065.62074.

Howe, N.R., 2013. Part-structured inkball models for one-shot handwritten word spotting, in: Proceedings of the International Conference on Document Analysis and Recognition, ICDAR, pp. 582–586.

Koopmans, T.C., Beckmann, M., 1957. Assignment Problems and the Location of Economic Activities. Econometrica 25, 53. URL: http://www.jstor.org/stable/1907742{\%}5Cnhttp://www.jstor.org/stable/10.2307/1907742http://www.jstor.org/stable/1907742?origin=crossref, doi:10.2307/1907742.

Kuncheva, L.I., 2004. Combining Pattern Classifiers. John Wiley & Sons, Inc., Hoboken, NJ, USA.

Manmatha, R., Chengfeng Han, Riseman, E., 1996. Word spotting: a new approach to indexing handwriting, in: Computer Vision and Pattern Recognition, IEEE. pp. 631–637.

Marti, U.V., Bunke, H., 2001. Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition systems. International Journal of Pattern Recognition and Artificial Intelligence 15, 65–90.

Perronnin, F., Rodríguez-Serrano, J.A., 2009. Fisher Kernels for Handwritten Word-spotting, in: International Conference on Document Analysis and Recognition, pp. 106–110.

Pratikakis, I., Zagoris, K., Gatos, B., Puigcerver, J., Toselli, A.H., Vidal, E., 2016. ICFHR2016 Handwritten Keyword Spotting Competition (H-KWS 2016), in: International Conference on Frontiers in Handwriting Recognition, IEEE. pp. 613–618.

Rath, T.M., Manmatha, R., 2007. Word spotting for historical documents. International Journal of Document Analysis and Recognition 9, 139–152.

Riba, P., Llados, J., Fornes, A., 2015. Handwritten word spotting by inexact matching of grapheme graphs, in: International Conference on Document Analysis and Recognition, pp. 781–785.

Riesen, K., 2015. Structural Pattern Recognition with Graph Edit Distance. Advances in Computer Vision and Pattern Recognition, Springer International Publishing, Cham. URL: http://link.springer.com/10.1007/978-3-319-27252-8, doi:10.1007/978-3-319-27252-8.

Riesen, K., Bunke, H., 2009. Approximate graph edit distance computation by means of bipartite graph matching. Image and Vision Computing 27, 950–959.

Rodríguez-Serrano, J.A., Perronnin, F., 2008. Local gradient histogram features for word spotting in unconstrained handwritten documents, in: International Conference on Frontiers in Handwriting Recognition, pp. 7–12.

Rothacker, L., Fink, G.A., 2015. Segmentation-free query-by-string word spotting with Bag-of-Features HMMs, in: International Conference on Document Analysis and Recognition, IEEE. pp. 661–665.

Rusiñol, M., Aldavert, D., Toledo, R., Lladós, J., 2015. Efficient segmentation-free keyword spotting in historical document collections. Pattern Recognition 48, 545–555.

Sakoe, H., Chiba, S., 1978. Dynamic programming algorithm optimization for spoken word recognition. IEEE Transactions on Acoustics, Speech, and Signal Processing 26, 43–49.

Scott, G.L., Longuet-Higgins, H.C., 1991. An Algorithm for Associating the Features of Two Images. Proceedings of the Royal Society B: Biological Sciences 244, 21–26.

Stauffer, M., Fischer, A., Riesen, K., 2016a. A Novel Graph Database for Handwritten Word Images, in: International Workshop on Structural, Syntactic, and Statistical Pattern Recognition, pp. 553–563.

Stauffer, M., Fischer, A., Riesen, K., 2016b. Graph-based Keyword Spotting in Historical Handwritten Documents, in: International Workshop on Structural, Syntactic, and Statistical Pattern Recognition, pp. 564–573.

Stauffer, M., Tschachtli, T., Fischer, A., Riesen, K., 2017. A Survey on Applications of Bipartite Graph Edit Distance, in: Graph-Based Representations in Pattern Recognition, pp. 242–252.

Sudholt, S., Fink, G.A., 2016. PHOCNet: A Deep Convolutional Neural Network for Word Spotting in Handwritten Documents, in: International Conference on Frontiers in Handwriting Recognition, IEEE. pp. 277–282.

Terasawa, K., Tanaka, Y., 2009. Slit Style HOG Feature for Document Image Word Spotting, in: International Conference on Document Analysis and Recognition, pp. 116–120.

Wang, P., Eglin, V., Garcia, C., Largeron, C., Llados, J., Fornes, A., 2014a. A Coarse-to-Fine Word Spotting Approach for Historical Handwritten Documents Based on Graph Embedding and Graph Edit Distance, in: International Conference on Pattern Recognition, IEEE. pp. 3074–3079.

Wang, P., Eglin, V., Garcia, C., Largeron, C., Llados, J., Fornes, A., 2014b. A Novel Learning-Free Word Spotting Approach Based on Graph Representation, in: International Workshop on Document Analysis Systems, pp. 207–211.

Wicht, B., Fischer, A., Hennebert, J., 2016. Deep Learning Features for Handwritten Keyword Spotting, in: International Conference on Pattern Recognition, pp. 3423–3428.