# Speeding-Up Graph-based Keyword Spotting in Historical Handwritten Documents

Michael Stauffer[1,4], Andreas Fischer[2,3], and Kaspar Riesen[1]

[1] University of Applied Sciences and Arts Northwestern Switzerland, Institute for Information Systems, Riggenbachstr. 16, 4600 Olten, Switzerland
{michael.stauffer,kaspar.riesen}@fhnw.ch
[2] University of Fribourg, Department of Informatics, 1700 Fribourg, Switzerland
andreas.fischer@unifr.ch
[3] University of Applied Sciences and Arts Western Switzerland, Institute for Complex Systems, 1705 Fribourg, Switzerland
[4] University of Pretoria, Department of Informatics, Pretoria, South Africa

**Abstract.** The present paper is concerned with a graph-based system for Keyword Spotting (KWS) in historical documents. This particular system operates on segmented words that are in turn represented as graphs. The basic KWS process employs the cubic-time bipartite matching algorithm (BP). Yet, even though this graph matching procedure is relatively efficient, the computation time is a limiting factor for processing large volumes of historical manuscripts. In order to speed up our framework, we propose a novel fast rejection heuristic. This heuristic compares the node distribution of the query graph and the document graph in a polar coordinate system. This comparison can be accomplished in linear time. If the node distributions are similar enough, the BP matching is actually carried out (otherwise the document graph is rejected). In an experimental evaluation on two benchmark datasets we show that about 50% or more of the matchings can be omitted with this procedure while the KWS accuracy is not negatively affected.

**Keywords:** Handwritten Keyword Spotting, Bipartite Graph Matching, Fast Rejection, Filtering Graph Matching

## 1 Introduction

An automatic full transcriptions of historical handwritten documents is often negatively affected by both the degenerative conservation state of scanned documents and different writing styles. Thus, *Keyword Spotting (KWS)* as a more error-tolerant, flexible, and suitable approach has been proposed [1–4]. KWS refers to the task of retrieving any instance of a given query word in a document. This task is of high relevance due to a global trend towards digitalisation of paper-based archives and libraries. Similar to handwriting recognition, textual KWS can be divided into two different approaches *online* and *offline* KWS, respectively. The former has access to temporal information, while the latter is limited to spatial information only. The focus of this paper is on historical

documents, and thus, offline KWS, referred to as KWS from now on, can be applied only.

KWS approaches can be divided into *template-based* or *learning-based* algorithms. Template-based matching algorithms such as for example *Dynamic Time Warping (DTW)* [2, 5, 6] directly match sample images of the keyword with document images. Learning-based algorithms [3, 4, 7], on the other hand, derive character or word models from learning samples. The latter typically achieve higher accuracies than template-based approaches but are limited by the need for a considerable amount of learning samples. Template-based approaches, in contrast, require only one or a few keyword instances and are thus more flexible. In this paper, we focus on template-based KWS using different graph representations of handwritten words.

Even though graphs gained noticeable attention in diverse applications [8, 9], we observe only limited attempts where graphs are used to represent handwriting for KWS [10–14]. This is particularly interesting as graphs are, in contrast with feature vectors, flexible enough to adapt their size to the size and complexity of the underlying handwriting. Moreover, graphs are capable to represent binary relationships in the handwriting (e.g. strokes between two keypoints). Overall, graphs offer a more natural and comprehensive way to represent handwritten characters or words when compared to feature vectors. Additionally, various procedure for efficiently evaluating the dissimilarity of graphs, commonly known as *graph matching*, have been proposed in the last decade [9].

Yet, in the case of searching $n$ keywords in a certain document (represented by a set of graphs $G$), we need to match $n \times |G|$ pairs of graphs. Even when a fast graph matching procedure is employed, this large amount of matchings can substantially slow down the complete KWS process. To speed up the KWS procedure the number of graph matchings actually carried out, can be reduced by efficiently filtering graphs from $G$ with a low similarity to the current query graph $q$. This approach is known as *fast rejection* [3, 5, 7] and the focus of the present paper. That is, we introduce a novel heuristic for fast and accurate filtering of irrelevant document graphs given a certain query graph.

The remainder of this paper is organised as follows. In Section 2, the proposed fast rejection approach to speed up graph-based KWS is introduced. The datasets employed as well as the different graph representations are reviewed in Section 3. An experimental evaluation and comparison with the original framework is given in Section 4. Finally, Section 5 concludes the paper and outlines possible future research activities.

## 2 Fast Rejection of Document Graphs

Given a set of document graphs $G = \{g_1, \ldots, g_N\}$ as well as query graph $q$ (used to represent a certain keyword), the process of KWS performs a matching of $q$ with all graphs from $G$. We employ the *Bipartite Graph Edit Distance (BP)* [15], and thus observe cubic time complexity for these pairwise dissimilarity computations. The present paper introduces a fast rejection approach in order to substantially

reduce the number of document graphs needed to be matched with $q$. The motivation is to filter document graphs without relevance to the given keyword and thus speeding up the KWS procedure without negatively affecting the retrieval accuracy.

The basic idea of our approach is as follows. Before actually carrying out the graph matching, we first measure the dissimilarity between histograms based on a specific segmentation of the graphs by means of a polar coordinate system. We denote this fast graph dissimilarity computation by *Polar Graph Dissimilarity (PGD)* from now on. If the PGD is below a certain threshold $D$ for a pair of graphs $(q, g_i)$, we carry out the computationally more expensive BP matching procedure [13]. Otherwise, we define the distance between $q$ and $g_i$ to be $\infty$. Formally,

$$d(q, g_i) \begin{cases} \infty, & \text{if } PGD(q, g_i) > D \\ BP(q, g_i), & \text{otherwise} \end{cases} , \tag{1}$$

where $q$ and $g_i$ denotes the query and document graph, respectively. Increasing the threshold $D$ generally reduces the number of filtered document graphs. Likewise, the number of filtered graphs is increased when $D$ is decreased. The overall aim is to find a good tradeoff between low matching time (due to many filterings) and high KWS accuracy.

Our novel dissimilarity model PGD has been inspired by the scale-invariant shape descriptor *Contour Points Distribution Histogram (CPDH)* for 2D-shape matching [16]. The basic idea behind this shape descriptor is to segment equidistant contour points by a specific polar coordinate system. A given shape image is formally described by a histogram CPDH $= \{h_1, \ldots, h_i, \ldots, h_n\}$ where $h_i$ basically consists of the number of contour points $n_i$ in the corresponding segment.

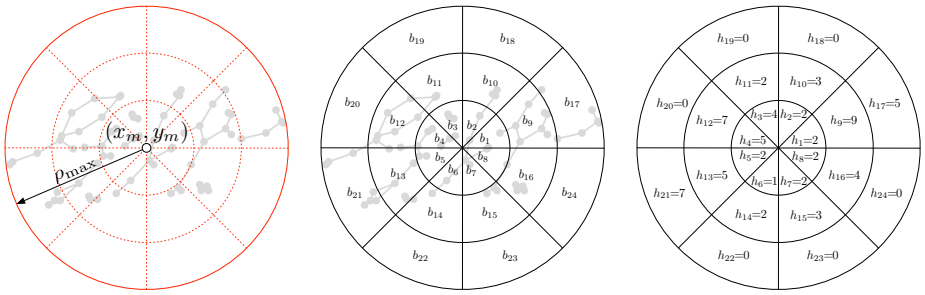
We adopt this procedure in order to measure the dissimilarity between graphs in linear time. Rather than contour points, however, we make use of nodes as shown in Fig. 1. For all of our graphs that represent segmented words, nodes are labelled with two-dimensional numerical labels, while edges remain unlabelled (see Section 3 for details).

To create a histogram for a given graph $g$, we first calculate the centre of mass $(x_m, y_m)$ of $g$ and then transform the $(x, y)$-coordinates of each node label $\mu(v) = (x, y) \in \mathbb{R}^2$ into polar coordinates (see Fig. 1a)[5]. Formally,

$$\rho = \sqrt{(x - x_m)^2 + (y - y_m)^2} \text{ and } \theta_i = \text{atan2}((y - y_m)/(x - x_m)) \quad ,$$

where $\rho$ denotes the radius from the centre of $g$ to the node position and $-\pi \leq \theta_i < \pi$ refers to the angle from the $x$-axis to the node position (computed via arctangent function with two arguments in order to return the correct quadrant). Next, we define a bounding circle $C$ given by the maximum radius $\rho_{\max}$ that surrounds all nodes of graph $g$. We segment $C$ based on the number of different

[5] Node coordinates are *a priori* denormalised by the standard deviation of all node coordinates, for further details we refer to [13].




(a) Circle $C$ with Centre of Mass $(x_m, y_m)$ and Radius $\rho_{\max}$

(b) Segmentation of $C$ into Bins

(c) Number of Nodes per Bin

Fig. 1: Construction of the Polar Graph Dissimilarity.

radii $u_{\max}$ and angles $v_{\max}$ into $u_{\max} \times v_{\max}$ bins (in Fig. 1b $u_{\max} = 3$ and $v_{\max} = 8$ resulting in 24 bins). Every bin $b_i$ is defined by two radii $\rho_{i_{\min}}$ and $\rho_{i_{\max}}$, and two angles $\theta_{i_{\min}}$ and $\theta_{i_{\max}}$, and thus every node $v \in V$ with coordinates $(\rho, \theta)$ can be assigned to the corresponding bin $b_i$ with $\rho_{i_{\min}} \leq \rho < \rho_{i_{\max}}$ and $\theta_{i_{\min}} \leq \theta < \theta_{i_{\max}}$. Finally, we count the number of nodes of $g$ in each bin and build a corresponding histogram $H = \{h_1, \ldots, h_n\}$ for graph $g$ (see Fig. 1c). To measure the dissimilarity between two histograms $H_1$ and $H_2$, an arsenal of different distance measures have been proposed [17]. In the present paper, we make use of the $\chi^2$ distance.

We further refine the computation of our fast graph dissimilarity computation by implementing a recursive *quadtree* segmentation. The idea is formalised in Algorithm 1. First, the procedure is initialised by an external call with $l = 1$ (i.e. $PGD(1, g_1, g_2)$). On the basis of two graphs $g_1$ and $g_2$, the histograms $H_1$ and $H_2$ are created with respect to $u_{\max}$ and $v_{\max}$ (see line 2 of Algorithm 1)[6]. Next, the $\chi^2$-distance between the two histograms is measured (see line 2). If the current recursion level $l$ is equal to the maximal recursion depth $r$, the distance is returned (see lines 4 and 5). Otherwise, both graphs $g_1$ and $g_2$ are segmented into four independent subgraphs. Each of these subgraphs represent the nodes and edges in one of the four quadrants in circle $C$ (see line 6). Eventually, for each subgraph pair, the PGD is measured by means of a recursive function call (see line 7). This procedure is repeated until the current recursion level $l$ is equal to the user-defined maximum depth $r$.

## 3 Handwriting Graphs

Our novel algorithm for fast rejection is evaluated in the context of KWS on two different manuscripts. First, the *George Washington (GW)* letters that are written in English and consist of twenty pages with a total of 4,894 words stemming from

[6] Note that $u_{\max}$ and $v_{\max}$ can be defined for every recursion level separately.

**Algorithm 1** Polar Graph Dissimilarity (PGD)

**Input:** Graphs $g_1$ and $g_2$, number of radii and segments $u_{\max}$ and $v_{\max}$, recursion depth $r$
**Output:** Polar graph dissimilarity between graph $g_1$ and $g_2$

**function** PGD($l$, $g_1$, $g_2$)
  Create $H_1$ based on $g_1$, $u_{\max}$, $v_{\max}$, and $H_2$ based on $g_2$, $u_{\max}$, $v_{\max}$
  Calculate $\chi^2$-distance $d(H_1, H_2)$
  **if** $l$ equal $r$ **then**
    **return** $d$
  Segment $g_1$ and $g_2$ based on quadtree to $g_{1_1}$, $g_{1_2}$, $g_{1_3}$, $g_{1_4}$ and $g_{2_1}$, $g_{2_2}$, $g_{2_3}$, $g_{2_4}$
  **return** $(\sum_{i=1}^{4} \text{PGD}(l+1, g_{1_i}, g_{2_i})) + d$

handwritten letters with only minor writing variations and signs of degradation[7]. Second, the *Parzival (PAR)* manuscript that is written in Middle High German and consists of 45 pages with a total of 23,478 words stemming from handwritten letters with low writing variations but markable signs of degradation[8].

We extract graphs from segmented words of both documents by means of the following four graph extraction algorithms (originally presented in [14]).

- `Keypoint`: The first graph extraction algorithm makes use of keypoints in the word images such as start, end, and junction points. These keypoints are represented as nodes that are labelled with the corresponding $(x, y)$-coordinates. Between pairs of keypoints further intermediate points are converted to nodes and added to the graph in 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 the word images. For every segment, a node is inserted into the graph and labelled by the $(x, y)$-coordinates of the centre of mass of this segment. Undirected edges are inserted between two neighbouring segments that are actually represented by a node. Eventually, the inserted edges are reduced by means of a *Minimal Spanning Tree* algorithm.
- `Projection`: The next graph extraction algorithm works very similar to `Grid`. However, this methods is based on an adaptive segmentation of word images by means of projection profiles (using horizontal and vertical projection profiles).
- `Split`: The last graph extraction algorithm is based on an iterative segmentation of word images. Segments are iteratively split into smaller subsegments until the width and height of all segments is below a certain threshold.

The dynamic range of the $(x, y)$-coordinates of each node label $\mu(v)$ is normalised with a z-score. Formally,

[7] 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`

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

$$\hat{x} = \frac{x - \mu_x}{\sigma_x} \text{ and } \hat{y} = \frac{y - \mu_y}{\sigma_y} \quad , \tag{2}$$

where $(\mu_x, \mu_y)$ and $(\sigma_x, \sigma_y)$ represent the mean and standard deviation of all $(x, y)$-coordinates in the graph under consideration.

On the resulting sets of word graphs, ten different keywords are manually selected on both datasets to optimise several system parameters (see Section 4.2). For validation these keywords are matched against a validation set that consists of 1,000 different random words including at least 10 instances of all 10 keywords. The optimised systems are eventually evaluated on the same training and test sets as used in [4]. All templates of a keyword present in the training set are used for KWS. In Table 1 a summary of the datasets is given.

Table 1: The number of keywords as well as the size of the training and test sets for both documents.

| Dataset | Keywords | Train | Test |
|---|---|---|---|
| GW | 105 | 2,447 | 1,224 |
| PAR | 1,217 | 11,468 | 6,869 |

## 4 Experimental Evaluation

### 4.1 Basic KWS Systems

For evaluating our proposed fast rejection heuristic, we consider the graph-based KWS system introduced in [13] and the four types of handwriting graphs described in Section 3. The original KWS system [13] is termed BP from now on, while our extended model with fast rejection is termed BP-FR.

To evaluate the KWS performance, two different metrics are used for global and local thresholds. In the case of global thresholds, the *Average Precision (AP)* is measured, which is the area under the Recall-Precision curve for all keywords given a single (global) threshold. In the case of local thresholds, we compute the *Mean Average Precision (MAP)*, that is the mean of all APs for each individual keyword query. To measure the effects of our fast rejection filter, we compute the relative amount of pairwise matchings that is filtered by BP-FR (termed *Filter Rate (FR)* from now on).

### 4.2 Optimisation of the Parameters

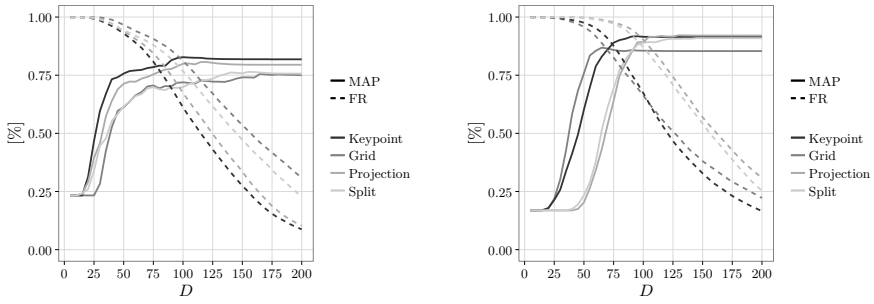
For the basic KWS system BP and the four graph representations, we adopt parameters from previous work [13, 14]. For our extension BP-FR the following parameters are additionally optimised on the validation set.

First, the parameters of PGD are optimised with respect to MAP. That is, we employ PGD (rather than BP) as basic matching procedure in our KWS framework. On the validation set different polar segmentations (defined via $u_{\max}$ and $v_{\max}$) are tested for two recursion levels (i.e. we define the maximal recursion depth to $r = 2$). For $l = 1$, the parameter combinations $u_{\max} = \{1, 2, 3, 4, 5, 6\} \times v_{\max} = \{4, 8, 12, 16, 20, 24, 28, 32, 36, 40\}$ are evaluated, while for $l = 2$ the parameter combinations $u_{\max} = \{1, 2, 3, 4\} \times v_{\max} = \{2, 4, 6, 8, 10\}$ are tested. Hence, we evaluate $6 \times 10 \times 4 \times 5 = 1{,}200$ parameter combinations for every graph extraction method. In Table 2 the best performing parameters are presented for every graph extraction method and both datasets.

Table 2: Optimal $u_{\max}$ and $v_{\max}$ for PGD on both recursion levels $l$.

| | GW | | | | PAR | | | |
|---|---|---|---|---|---|---|---|---|
| | $l = 1$ | | $l = 2$ | | $l = 1$ | | $l = 2$ | |
| Method | $u_{\max}$ | $v_{\max}$ | $u_{\max}$ | $v_{\max}$ | $u_{\max}$ | $v_{\max}$ | $u_{\max}$ | $v_{\max}$ |
| `Keypoint` | 4 | 12 | 1 | 6 | 3 | 20 | 2 | 6 |
| `Grid` | 5 | 24 | 1 | 4 | 4 | 20 | 1 | 6 |
| `Projection` | 5 | 16 | 1 | 4 | 3 | 36 | 3 | 4 |
| `Split` | 4 | 20 | 1 | 4 | 3 | 40 | 2 | 6 |

For fast rejection in our extension BP-FR we evaluate different thresholds $D = \{5, 10, \ldots, 195, 200\}$. In Fig. 2, the MAP and FR are shown for every tested threshold $D$. By increasing $D$ we observe that the KWS performance is improved in general. Simultaneously, the number of filtered graphs is decreasing (making the KWS process slower in general). Threshold $D$ is finally determined such that the MAP is maximal (or not further improved, when $D$ is increased). In Table 3 the selected threshold $D$ is given for each graph extraction method and both datasets.




(a) GW (b) PAR

Fig. 2: Mean average precision (MAP) and filter rate (FR) as function of the threshold $D$.

Table 3: Optimal $D$ for BP-FR and corresponding filter rate (FR).

| | GW | | | PAR | | |
|---|---|---|---|---|---|---|
| Method | $D$ | MAP | FR | $D$ | MAP | FR |
| `Keypoint` | 100 | 82.8 | 61.1 | 95 | 91.7 | 71.5 |
| `Grid` | 165 | 75.6 | 46.0 | 70 | 86.5 | 85.6 |
| `Projection` | 115 | 80.7 | 56.9 | 130 | 92.2 | 70.9 |
| `Split` | 155 | 76.4 | 44.6 | 145 | 90.9 | 57.5 |

### 4.3 Results and Discussion

We compare the optimised system BP-FR on the independent test sets with the original KWS framework BP [13] (without fast rejection). In Table 4 the mean average precision (MAP) for local thresholds, the average precision (AP) for global thresholds, as well as the filter rate (FR) is given for both BP and BP-FR. On the GW dataset we observe a filter rate between 50% and 70% (i.e. only 50% to 30% of all comparisons have to be carried out by the bipartite matching algorithm). Due to this filtering, we decrease the computation time of the complete KWS experiment by about 80 to 150 hours on the different graph representations. Similar (or even better) filter rates can be observed on the second dataset[9].

Table 4: Mean average precision (MAP) using local thresholds, average precision (AP) using a global threshold, and filter rate (FR) for KWS using the original bipartite graph matching without rejection (BP) and with the proposed fast rejection (BP-FR). With ± we indicate the relative percental gain or loss in the accuracy of BP-FR when compared with BP.

| | | GW | | | | | PAR | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Method | MAP | ± | AP | ± | FR | MAP | ± | AP | ± | FR |
| BP | `Keypoint` | 66.08 | | 54.99 | | 0.00 | 62.04 | | 60.74 | | 0.00 |
| BP | `Grid` | 60.02 | | 46.44 | | 0.00 | 56.50 | | 44.08 | | 0.00 |
| BP | `Projection` | 61.43 | | 48.69 | | 0.00 | 66.23 | | 60.61 | | 0.00 |
| BP | `Split` | 60.23 | | 47.96 | | 0.00 | 59.44 | | 55.46 | | 0.00 |
| BP-FR | `Keypoint` | 68.81 | +4.12 | 55.68 | +1.25 | 69.04 | 67.70 | +9.12 | 58.03 | −4.46 | 58.72 |
| BP-FR | `Grid` | 62.59 | +4.27 | 47.48 | +2.23 | 54.65 | 63.41 | +12.23 | 38.59 | −12.45 | 78.71 |
| BP-FR | `Projection` | 64.65 | +5.25 | 50.41 | +3.53 | 61.04 | 72.02 | +8.74 | 55.83 | −7.89 | 58.10 |
| BP-FR | `Split` | 63.49 | +5.41 | 46.95 | −2.11 | 47.70 | 65.65 | +10.45 | 56.97 | +2.72 | 39.24 |

Regarding the effects of our fast filtering on the KWS performance, we observe that the MAP is not negatively affected on both datasets. On the contrary, the

[9] Actually, we carry out our experiment on a high performance computing cluster with dozens of CPU nodes. Hence, these readings are approximated by means of the average matching time per keyword measured on the validation set in a sequential scenario.

filtering of irrelevant documents via PGD actually improves the MAP by about 5% and 10% on the GW and PAR dataset, respectively.

Regarding the AP (employed for global rather than local thresholds), we observe both deteriorations and improvements of BP-FR when compared with the original framework. Yet, most of the deviations are negligible. In particular on the GW dataset only small differences are observed on the resulting APs. On PAR we observe two substantial deteriorations of the AP. Yet, in these two cases we observe very high filter rates of about 60% and 70%.

Regarding the results in Table 4 the question arises whether the novel graph dissimilarity PGD would be able to achieve a competitive KWS accuracy. In order to answer this question, we employ the optimised PGD (rather than the bipartite matching) in the original KWS framework. In Table 5, the MAP and AP of this particular KWS system is shown on the `Keypoint` graphs (for the other graphs similar results are obtained). We observe that this system achieves worse results than BP on both datasets (regarding both MAP and AP). Hence, we conclude that PGD itself is not powerful enough to serve as basic dissimilarity model for graph-based KWS. Yet, as seen in the previous evaluation in Table 4, the PGD as fast rejection criterion in conjunction with BP is clearly beneficial.

Table 5: Mean average precision (MAP) using local thresholds, average precision (AP) using a global threshold for KWS using the original bipartite graph matching (BP), and the polar graph dissimilarity (PGD) on the `Keypoint` graphs.

| | GW | | PAR | |
|---|---|---|---|---|
| | MAP | AP | MAP | AP |
| BP | 66.08 | 54.99 | 62.04 | 60.74 |
| PGD | 58.54 | 44.77 | 42.65 | 31.63 |

## 5 Conclusion and Outlook

In the present paper a fast rejection approach for graph-based KWS is introduced. The rejection is based on a novel graph dissimilarity model, which compares the histograms of the node distributions in a polar coordinate system.

We compare our extended model with the original KWS framework without rejection ability on two benchmark datasets. We observe that our novel rejection approach reduces the amount of graph matchings by 50% or more on both datasets (in fact, filter rates of up to 80% are observed). Our rejection criterion is computed in linear time, while the actual graph matching needs cubic time. Hence, a dramatic speed up of the complete KWS process is achieved. Moreover, we can conclude that our novel extension for speeding up the existing KWS framework does not negatively influence the spotting accuracy.

In future work we aim at extending our novel graph dissimilarity model. For instance, we could consider not only nodes but also edges in the histograms.

**Acknowledgments.** This work has been supported by the Hasler Foundation Switzerland.

## References

1. Manmatha, R., Chengfeng Han, Riseman, E.: Word spotting: a new approach to indexing handwriting. In: Computer Vision and Pattern Recognition. (1996) 631–637
2. Rath, T., Manmatha, R.: Word image matching using dynamic time warping. In: Computer Vision and Pattern Recognition. Volume 2. (2003) II–521–II–527
3. Rodríguez-Serrano, J.A., Perronnin, F.: Handwritten word-spotting using hidden Markov models and universal vocabularies. Pattern Recognition **42**(9) (2009) 2106–2116
4. Fischer, A., Keller, A., Frinken, V., Bunke, H.: Lexicon-free handwritten word spotting using character HMMs. Pattern Recognition Letters **33**(7) (2012) 934–942
5. Rodriguez, J.A., Perronnin, F.: Local gradient histogram features for word spotting in unconstrained handwritten documents. In: International Conference on Frontiers in Handwriting Recognition. (2008) 7–12
6. Rodríguez-Serrano, J.A., Perronnin, F.: A model-based sequence similarity with application to handwritten word spotting. IEEE Transactions on Pattern Analysis and Machine Intelligence **34**(11) (2012) 2108–20
7. Perronnin, F., Rodriguez-Serrano, J.A.: Fisher Kernels for Handwritten Word-spotting. In: International Conference on Document Analysis and Recognition. (2009) 106–110
8. Conte, D., Foggia, P., Sansone, C., Vento, M.: Thirty Years Of Graph Matching In Pattern Recognition. International Journal of Pattern Recognition and Artificial Intelligence **18**(03) (2004) 265–298
9. Riesen, K.: Structural Pattern Recognition with Graph Edit Distance. Advances in Computer Vision and Pattern Recognition, Cham (2015)
10. Wang, P., Eglin, V., Garcia, C., Largeron, C., Llados, J., Fornes, A.: A Novel Learning-Free Word Spotting Approach Based on Graph Representation. In: International Workshop on Document Analysis Systems. (2014) 207–211
11. Bui, Q.A., Visani, M., Mullot, R.: Unsupervised word spotting using a graph representation based on invariants. In: International Conference on Document Analysis and Recognition. (2015) 616–620
12. Riba, P., Llados, J., Fornes, A.: Handwritten word spotting by inexact matching of grapheme graphs. In: International Conference on Document Analysis and Recognition. (2015) 781–785
13. Stauffer, M., Fischer, A., Riesen, K.: Graph-based Keyword Spotting in Historical Handwritten Documents. In: International Workshop on Structural, Syntactic, and Statistical Pattern Recognition. (2016)
14. Stauffer, M., Fischer, A., Riesen, K.: A Novel Graph Database for Handwritten Word Images. In: International Workshop on Structural, Syntactic, and Statistical Pattern Recognition. (2016)
15. Riesen, K., Bunke, H.: Approximate graph edit distance computation by means of bipartite graph matching. Image and Vision Computing **27**(7) (2009) 950–959
16. Shu, X., Wu, X.J.: A novel contour descriptor for 2D shape matching and its application to image retrieval. Image and Vision Computing **29**(4) (2011) 286–294
17. Serratosa, F., Sanfeliu, A.: Signatures versus histograms: Definitions, distances and algorithms. Pattern Recognition **39**(5) (2006) 921–934