Published in the Proceedings of Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR), 29 November-2 December 2016, Mérida, Mexico, which should be cited to refer to this work.

DOI: https://doi.org/10.1007/978-3-319-49055-7_50

Graph-based Keyword Spotting in Historical Handwritten Documents

Michael Stauffer[1,3], Andreas Fischer[2], 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 and HES-SO, 1700 Fribourg, Switzerland
andreas.fischer@unifr.ch
[3] University of Pretoria, Department of Informatics, Pretoria, South Africa

Abstract. The amount of handwritten documents that is digitally available is rapidly increasing. However, we observe a certain lack of accessibility to these documents especially with respect to searching and browsing. This paper aims at closing this gap by means of a novel method for keyword spotting in ancient handwritten documents. The proposed system relies on a keypoint-based graph representation for individual words. Keypoints are characteristic points in a word image that are represented by nodes, while edges are employed to represent strokes between two keypoints. The basic task of keyword spotting is then conducted by a recent approximation algorithm for graph edit distance. The novel framework for graph-based keyword spotting is tested on the George Washington dataset on which a state-of-the-art reference system is clearly outperformed.

Keywords: Handwritten Keyword Spotting, Bipartite Graph Matching, Graph Representation for Words

1 Introduction

Keyword Spotting (KWS) is the task of retrieving any instance of a given query word in speech recordings or text images [1–3]. Textual KWS can be roughly divided into *online* and *offline* KWS. For online KWS temporal information of the handwriting is available recorded by an electronic input device such as, for instance, a digital pen or a tablet computer. On the other hand side, offline KWS is based on scanned image only, and thus, offline KWS is regarded as the more difficult task than its online counterpart. The focus of this paper is on KWS in historical handwritten documents. Therefore, offline KWS, referred to as KWS from now on, can be applied only.

Most of the KWS methodologies available are either based on *template-based* or *learning-based* matching algorithms. Early approaches of template-based KWS are based on a pixel-by-pixel matching of word images [1]. More elaborated approaches to template-based KWS are based on the matching of feature vectors by means of *Dynamic Time Warping (DTW)* [4]. A recent

and promising approach to template-based KWS is given by the matching of *Local Binary Pattern (LBP)* histograms [5]. One of the main advantages of template-based KWS is its independence from the actual representation formalism as well as the underlying language (and alphabet) of the document. However, template-based KWS does not generalise well to different writing styles. Learning-based KWS on the other side is based on statistical models like *Hidden Markov Models (HMM)* [6,7], *Neural Networks (NN)* [3] or *Support Vector Machines (SVM)* [8]. These models have to be trained a priori on a (relatively large) set of training words. An advantage of the learning-based approach, when compared with the template-based approach, is its higher generalisability. Yet, this advantage is accompanied by a loss of flexibility, which is due to the need for learning the parameters of the actual model on a specific training set.

The vast majority of KWS algorithms available are based on statistical representations of word images by certain numerical features. To the best of our knowledge only few graph-based KWS approaches have been proposed so far [9–12]. However, a graph-based representation is particularly interesting for KWS as graphs, in contrast with feature vectors, offer a more natural and comprehensive formalism for the representation of word images.

A first approach for graph-based KWS has been proposed in [10]. The nodes of the employed graphs represent keypoints that are extracted on connected components of the skeletonised word images, while the edges are used to represent the strokes between the keypoints. The majority of the words consists of more than only one connected component, and thus, a word is in general represented by more than one graph. The matching of words is thus based on two separate procedures. First, the individual costs of assignments of all pairs of connected components (represented by graphs) are computed via bipartite graph matching [13]. Second, an optimal assignment between the connected components has to be found. To this end, a DTW algorithm is employed that operates on the costs produced in the first step. This matching procedure is further improved by a so-called *coarse-to-fine approach* in [11].

Another idea for graph-based KWS has been introduced in [12]. In this paper a graph represents a set of prototype strokes (called invariants). First, a word image is segmented into strokes. Eventually, the most similar invariant is defined for every stroke in the word. The nodes of the graph are used to represent these invariants, while edges are inserted between all pairs of nodes. Edges are labelled with the information whether or not strokes of the corresponding nodes are stemming from the same connected component. Finally, for KWS the graph edit distance is computed by means of the bipartite graph matching algorithm [13].

A third approach for graph-based KWS has been proposed in [9] where complete text lines are represented by a *grapheme graph*. Graphemes are sets of prototype convexity paths, similar to invariants, that are defined a priori in a codebook. The nodes of the particular graphs represent the individual graphemes of the text line, while edges are inserted into the graph whenever two graphemes are directly connected to each other. The matching itself is based on a coarse-to-fine approach. Formally, potential subgraphs of the query graph are determined

first. These subgraphs are subsequently matched against a query graph by means of the bipartite graph matching algorithm [13].

In the present paper we introduce a novel approach for graph representation of individual words that is based on the detection of keypoints. In contrast with [10] our approach results in a single graph per word. Hence, no additional assignment between graphs of different connected components is necessary during the matching process. Furthermore, in our approach the edges are detected by a novel method based on both the skeleton of connected components and their connected subcomponents. Last but not least, also the graph matching procedure actually employed for KWS has been substantially extended when compared to the previous contributions in the field. In particular, we introduce different types of linear and non-linear cost functions for the edit operations used in [13].

The remainder of this paper is organised as follows. In Sect. 2, the basic concept of graph edit distance is briefly reviewed. In Sect. 3, the proposed graph-based KWS approach is introduced. An experimental evaluation of the proposed framework is given in Sect. 4. Section 5 concludes the paper and outlines possible further research activities.

2 Graph Edit Distance

A graph g is formally 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 *unlabelled* and *labelled* graphs. In the latter case, both nodes and edges can be labelled 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. $L_v = L_e = \{\}$.

Graphs can be matched with exact and inexact methods [14,15]. Inexact graph matching, in contrast to exact graph matching, allows matchings between two non-identical graphs by endowing the matching with a certain error-tolerance with respect to labelling and structure. Several approaches for inexact graph matching have been proposed. Yet, *Graph Edit Distance (GED)* is widely accepted as one of the most flexible and powerful paradigms available [16]. The GED between two graphs g_1 and g_2 is defined as the least costly series of edit operations to be applied to g_1 in order to make it *isomorphic* to g_2. Formally,

$$GED(g_1, g_2) = \min_{(e_1,\ldots,e_k)\in\gamma(g_1,g_2)} \sum_{i=1}^{k} c(e_i)$$

where e_i denotes an edit operation, $(e_1, \ldots, e_n)$ an edit path, $\gamma(g_1, g_2)$ the set of all edit paths that transform g_1 into g_2, and $c(e_i)$ the cost for a certain edit operation e_i. Different types of edit operations are allowed such as substitutions, insertions, deletions, splittings, and mergings of both nodes and edges. Commonly, the cost function $c(e_i)$ considers domain-specific knowledge and reflects the strength of edit operation e_i.

The computation of the exact GED is commonly based on an A*-algorithm that explores all possible edit paths $\gamma(g_1, g_2)$ [17]. However, this exhaustive search is exponential with respect to the number of nodes of the involved graphs.

In order to make the concept of GED applicable to large graphs and/or large graph sets, several fast but approximative algorithms have been proposed [13, 18]. In the present paper we make use of the well-known bipartite graph matching algorithm for approximating the GED in cubic time complexity [13]. This algorithm is based on an optimal match between nodes and their local structure (i.e. their adjacent edges). That is, the suboptimal computation of the GED is based on a reduction of the GED problem to a *Linear Sum Assignment Problem (LSAP)*, which can be optimally solved by, for instance, Munkres' algorithm [19]. In case of scalability limitations, one could also make use of the graph matching algorithm for approximating the GED in quadratic, rather than cubic, time complexity [18].

3 Graph-based Keyword Spotting

The proposed graph-based KWS solution is based on four different processing steps as shown in Fig. 1. First, document images are preprocessed and segmented into words (1). Based on the segmented word images, graphs are extracted by means of a novel keypoint-based method (2) and eventually normalised (3). Finally, the graphs of query words are matched against graphs from the document to create a retrieval index (4). In the following four subsections these four steps are described in greater detail.

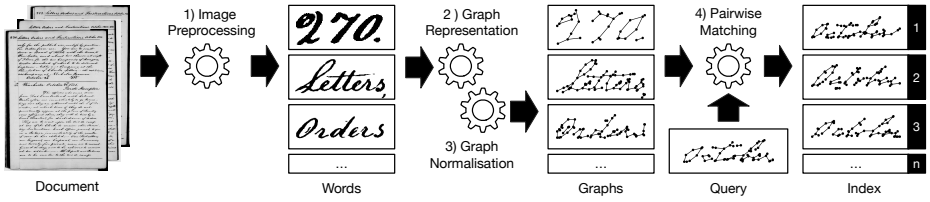

Fig. 1. Process of graph-based keyword spotting of the word "October"

3.1 Image Preprocessing

Image preprocessing aims at reducing variations on document images that are caused, for instance, by noisy background, skewed scanning, or document degradation. In our particular framework, document images are first filtered by a *Difference of Gaussian (DoG)* and binarised by a global threshold [20]. Single word images are then manually segmented. That is, we build our framework on perfectly segmented words in order to focus on the task of KWS. The skew, i.e. the inclination of the document, is removed by a hierarchical rotation of the complete document image such that the horizontal projection profile is step-wise maximised [21]. Note that the skew angle is estimated on complete document images first and then corrected on single word images. Finally, each word image is skeletonised by a 3×3 thinning operator [22].

3.2 Graph Representation

For a graph-based KWS system to succeed, the variations among graphs of the same word have to be minimised, while variations of graphs of different words should remain large. Hence, a graph representation has to represent the inherent characteristic of a word. In the present paper the graph extraction algorithm is based on the detection of keypoints. Keypoints are characteristic points in a word image, such as for instance end- and intersection-points of strokes. The proposed approach is inspired by [7]. However, in contrast with [7] the proposed graph representation makes use of both nodes and edges. Additionally, the keypoint detection is further refined by a local search algorithm.

Graphs are created on the basis of filtered, binarised, and skeletonised word images S (see Algorithm 1). First, end points and junction points are identified for each *Connected Component (CC)* of the skeleton image (see line 2 of Algorithm 1). For circular structures, such as for instance the letter 'O', the upper left point is selected as junction point. Note that the skeletons based on [22] may contain several neighbouring end- or junction points. We apply a local search procedure to select only one point at each ending and junction (this step is not explicitly formalised in Algorithm 1). Both end points and junction points are added to the graph as nodes, labelled with their image coordinates (x, y) (see line 3).

Next, junction points are removed from the skeleton, dividing it into *Connected Subcomponents (CC_{sub})* (see line 4). Afterwards, for each connected subcomponent intermediate points $(x, y) \in CC_{sub}$ are converted to nodes and added to the graph in equidistant intervals of size D (see line 5 and 6).

Finally, an undirected edge (u, v) between $u \in V$ and $v \in V$ is inserted into the graph for each pair of nodes that is directly connected by a chain of foreground pixels in the skeleton image S (see line 7 and 8).

Algorithm 1 Graph Extraction Based on Keypoints

Input: Skeleton image S, Distance threshold D
Output: Graph $g = (V, E)$ with nodes V and edges E

1: **function** KEYPOINT(S,D)
2: **for** Each connected component $CC \in S$ **do**
3: $V = V \cup \{(x, y) \in CC \mid (x, y)$ are end- or junction points$\}$
4: Remove junction points from CC
5: **for** Each connected subcomponent $CC_{sub} \in CC$ **do**
6: $V = V \cup \{(x, y) \in CC_{sub} \mid (x, y)$ are points in equidistant intervals $D\}$
7: **for** Each pair of nodes $(u, v) \in V \times V$ **do**
8: $E = E \cup (u, v)$ if the corresponding points are connected in S
9: **return** $g = (V, E)$

3.3 Graph Normalisation

In order to improve the comparability between graphs of the same word class, the labels $\mu(v)$ of the nodes $v \in V$ are normalised. In our case the node label alphabet is defined by $L_v = \mathbb{R}^2$. A first graph normalisation is based on a centralisation of each node label $\mu(v) = (x, y) \in \mathbb{R}^2$ by

$$\hat{x} = x - \mu_x \text{ and } \hat{y} = y - \mu_y, \tag{1}$$

where $\hat{x}$ and $\hat{y}$ denote the new node coordinates, x and y the original node position, μ_x, and μ_y represent the mean values of all (x, y)-coordinates in the graph under consideration.

The second graph normalisation centralises the node labels and reduces variations of node positions that might occur due to different word image sizes. Formally,

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

where σ_x and σ_y denote the standard deviation of all node coordinates in the current graph.

3.4 Pairwise Matching

The actual KWS is based on a pairwise matching of a query graph g against all graphs of a set of word graphs $G = \{g_1, \ldots, g_n\}$ stemming from the underlying document. We make use of the bipartite graph matching algorithm [13]. In our system the resulting GED between g and $g_i \in G$ is normalised by using the cost of the maximum cost edit path between g and g_i, viz. the edit path that results from deleting all nodes and edges of g and inserting all nodes and edges of g_i. We refer to this maximum cost as *Max-GED* from now on. By means of this procedure a retrieval index $r_i(g) \in [0, 1]$ can be created for every word graph $g_i \in G$ given a certain query graph g. Formally,

$$r_i(g) = \frac{GED(g, g_i)}{Max\text{-}GED(g, g_i)}$$

The effectiveness of edit distance based KWS relies on an adequate definition of cost functions for the basic edit operations. In general, the cost $c(e)$ of a particular edit operation e is defined with respect to the underlying label alphabets L_V and L_E. In our framework the nodes are labelled with two-dimensional numerical labels while edges remain unlabelled, i.e. $L_V = \mathbb{R}^2$ and $L_E = \{\}$. In the present section four cost functions are defined for this particular labelling.

For all of our cost models a constant cost $\tau_v \in \mathbb{R}^+$ for node deletion and insertion is used. Formally, the cost for the node deletions and insertions is defined by $c(u \to \varepsilon) = c(\varepsilon \to v) = \tau_v$. For edges a similar cost with another constant cost $\tau_e \in \mathbb{R}^+$ is defined. The cost models to be used in our framework differ in the definition of the cost for node substitutions. The basic intuition

behind all approaches is that the more dissimilar two labels are, the stronger is the distortion associated with the corresponding substitution.

The first cost model is based on a weighted Euclidean distance of the two corresponding labels. Formally, given two graphs $g_1 = (V_1, E_1, \mu_1, \nu_1)$ and $g_2 = (V_2, E_2, \mu_2, \nu_2)$, where $\mu_1, \mu_2 : V_1, V_2 \rightarrow \mathbb{R}^2$ the cost for a node substitution $(u \rightarrow v)$ with $\mu_1(u) = (x_i, y_i)$ and $\mu_2(v) = (x_j, y_j)$ is defined by

$$c_E(u \rightarrow v) = \sqrt{\alpha(x_i - x_j)^2 + (1-\alpha)(y_i - y_j)^2},$$

where $\alpha \in [0, 1]$ is a weighting parameter to define whether the x- or the y-coordinate is more important for the resulting substitution cost.

For graphs with scaled node labels (see Sect. 3.3) the standard deviation σ of the node labels of a query graph might be additionally included in the cost model by defining

$$c_{E_\sigma}(u \rightarrow v) = \sqrt{\alpha\sigma_x(x_i - x_j)^2 + (1-\alpha)\sigma_y(y_i - y_j)^2},$$

where σ_x and σ_y denote the standard deviation of all node coordinates in the query graph.

The third and fourth cost function are based on the weighted Euclidean distance that is additionally scaled by means of a Sigmoidal function to $[0, 2\tau_v]$. Formally,

$$c_S(u \rightarrow v) = \frac{2\tau_v}{1 + e^{(kc_E(u \rightarrow v) - \gamma)}} \text{ and } c_{S_\sigma}(u \rightarrow v) = \frac{2\tau_v}{1 + e^{(kc_{E_\sigma}(u \rightarrow v) - \gamma)}},$$

where k is the steepness and γ the threshold of the Sigmoidal function. For both cost functions c_S and c_{S_σ} the maximal substitution cost is equal to the sum of cost of a node deletion and node insertion.

4 Experimental Evaluation

The experimental evaluation of the proposed KWS system is carried out on the *George Washington (GW)* dataset, which consists of twenty pages of handwritten letters with only minor variations in the writing style[4]. The same dataset has already been used in [6, 23, 24]. For our KWS framework individual graphs are created for each of the 4893 words of the dataset by means of the keypoint-based graph representation algorithm described in Sect. 3.2. We use a threshold of $D = 5$ for all of our evaluations.

The performance of KWS is measured by the *mean Average Precision (mAP)* in two subsequent experiments. First, the meta-parameters and the different image and graph normalisations are optimised for all cost functions. To this end, the mAP is computed on a small validation set, consisting of ten different query

[4] 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`

words with a frequency of at least 10 as well as a reduced training set based on 1000 different words including all instances of the query word.

In Table 1 the results of this validation phase are shown. We distinguish between graphs that are based on word images where the skew is corrected or not. For both variants we use graphs where the node labels remain unnormalised (denoted by U in Table 1), and graphs where the labels are normalised by using (1) and (2) (denoted by N_1 and N_2, respectively). Note that the cost models c_E and c_S can be applied to graph normalisation with N_2 only.

Table 1. mAP of Euclidean and Sigmoidal cost functions for different preprocessing

Preprocessing	Skew not corrected			Skew corrected		
Cost Function	U	N_1	N_2	U	N_1	N_2
c_E	50.17	72.87	-	47.08	72.24	-
c_{E_σ}	-	-	**76.53**	-	-	75.59
c_S	49.71	72.72	-	50.60	73.53	-
c_{S_σ}	-	-	**76.24**	-	-	75.24

We observe that graphs based on not skew corrected word images in combination with scaled and centralised node labels (N_2) is optimal for both the Euclidean and the Sigmoidal cost functions. These two models are further optimised by means of the node label weighting factor α. By using this weighting parameter, the mAP can be increased from 76.53 to 80.94 and from 76.24 to 79.32 with c_{E_σ} and c_{S_σ}, respectively.

Using this optimal parameter settings, the proposed KWS system is compared with a reference system based on DTW [23, 24] with optimised Sakoe-Chiba band. This evaluation is conducted in a four-fold cross-validation, where each fold consists of a test set (avg. 2447 words) that is tested with a training set (avg. 1223.5 words).

In Table 2 the results of our novel graph-based KWS system (using both c_{E_σ} and c_{S_σ}) and the reference DTW system are given. Our graph-based system outperforms the DTW-based KWS system in both cases. The Euclidean and Sigmoidal cost models improve the mAP of the reference system by 2.31% and 5.62%, respectively.

Table 2. Graph-based vs. DTW-based KWS

System	mAP	Improvement
DTW	54.08	
Proposed c_{E_σ}	55.33	+ 2.31%
Proposed c_{S_σ}	57.12	+ 5.62%

5 Conclusion and Outlook

The novel KWS system proposed in this paper is based on a keypoint-based graph representation of individual words. Keypoints are characteristic points in a word image that are represented by nodes, while edges are represented by strokes between two keypoints. The actual KWS is based on a bipartite matching of pairs of graphs. Four different cost functions have been introduced to quantify the substitution cost of nodes that are matched. These cost functions in combination with different image and graph normalisations are optimised on the George Washington dataset. The optimal system clearly outperforms the reference DTW algorithm.

In future work, we aim at extending our word-based approach to a line-based approach. The actual KWS would therefore be based on finding a subgraph isomorphism of a query graph in the larger line graph. Moreover, other graph representation formalisms as well as more powerful labelling functions could be a rewarding avenue to be pursued. Thus, we will be able to conduct a more thorough comparison against other state-of-the-art systems using further graph representations and documents.

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: Comp. Vision and Pattern Rec. (1996) 631–637
2. Rath, T., Manmatha, R.: Word image matching using dynamic time warping. In: Comp. Vision and Pattern Rec. Volume 2. (2003) II–521–II–527
3. Frinken, V., Fischer, A., Manmatha, R., Bunke, H.: A novel word spotting method based on recurrent neural networks. IEEE Trans. PAMI **34**(2) (2012) 211–224
4. Kolcz, A., Alspector, J., Augusteijn, M., Carlson, R., Viorel Popescu, G.: A Line-Oriented Approach to Word Spotting in Handwritten Documents. Pattern Anal. and Appl. **3**(2) (2000) 153–168
5. Dey, S., Nicolaou, A., Llados, J., Pal, U.: Local Binary Pattern for Word Spotting in Handwritten Historical Document. Computing Research Repository (2016)
6. Lavrenko, V., Rath, T., Manmatha, R.: Holistic word recognition for handwritten historical documents. In: Int. W. on Doc. Image Anal. for Libraries. (2004) 278–287
7. Fischer, A., Riesen, K., Bunke, H.: Graph similarity features for HMM-based handwriting recognition in historical documents. In: Int. Conf. on Frontiers in Handwriting Rec. (2010) 253–258
8. Huang, L., Yin, F., Chen, Q.H., Liu, C.L.: Keyword Spotting in Offline Chinese Handwritten Documents Using a Statistical Model. In: Int. Conf. on Doc. Anal. and Rec. (2011) 78–82
9. Riba, P., Llados, J., Fornes, A.: Handwritten word spotting by inexact matching of grapheme graphs. In: Int. Conf. on Doc. Anal. and Rec. (2015) 781–785
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: Int. W. on Doc. Anal. Systems. (2014) 207–211

11. Wang, P., Eglin, V., Garcia, C., Largeron, C., Llados, J., Fornes, A.: A Coarse-to-Fine Word Spotting Approach for Historical Handwritten Documents Based on Graph Embedding and Graph Edit Distance. In: Int. Conf. on Pattern Rec. (2014) 3074–3079
12. Bui, Q.A., Visani, M., Mullot, R.: Unsupervised word spotting using a graph representation based on invariants. In: Int. Conf. on Doc. Anal. and Rec. (2015) 616–620
13. Riesen, K., Bunke, H.: Approximate graph edit distance computation by means of bipartite graph matching. Image and Vision Computing **27**(7) (2009) 950–959
14. Conte, D., Foggia, P., Sansone, C., Vento, M.: Thirty Years Of Graph Matching In Pattern Recognition. Int. J. Pattern Rec. Artif. Intell. **18**(03) (2004) 265–298
15. Foggia, P., Percannella, G., Vento, M.: Graph Matching and Learning in Pattern Recognition in the last 10 Years. Int. J. Pattern Rec. Artif. Intell. **28**(01) (2014)
16. Bunke, H., Allermann, G.: Inexact graph matching for structural pattern recognition. Pattern Rec. Letters **1**(4) (1983) 245–253
17. Hart, P., Nilsson, N., Raphael, B.: A Formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Trans. on Systems Science and Cybernetics **4**(2) (1968) 100–107
18. Fischer, A., Suen, C.Y., Frinken, V., Riesen, K., Bunke, H.: Approximation of graph edit distance based on Hausdorff matching. Pattern Rec. **48**(2) (2015) 331–343
19. Munkres, J.: Algorithms for the Assignment and Transportation Problems. J. of the Society for Industrial and Applied Mathematics **5**(1) (1957) 32–38
20. Fischer, A., Indermühle, E., Bunke, H., Viehhauser, G., Stolz, M.: Ground truth creation for handwriting recognition in historical documents. In: Int. W. on Doc. Anal. Systems, New York, New York, USA (2010) 3–10
21. Hull, J.: Document image skew detection: Survey and annotated bibliography. In: Series in Machine Perception and Artif. Intell. Volume 29. (1998) 40–64
22. Guo, Z., Hall, R.W.: Parallel thinning with two-subiteration algorithms. Communications of the ACM **32**(3) (1989) 359–373
23. Rath, T.M., Manmatha, R.: Word spotting for historical documents. Int. J. of Doc. Anal. and Rec. **9**(2-4) (2007) 139–152
24. Fischer, A., Keller, A., Frinken, V., Bunke, H.: Lexicon-free handwritten word spotting using character HMMs. Pattern Rec. Letters **33**(7) (2012) 934–942