A Novel Graph Database for Handwritten Word Images

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. For several decades graphs act as a powerful and flexible representation formalism in pattern recognition and related fields. For instance, graphs have been employed for specific tasks in image and video analysis, bioinformatics, or network analysis. Yet, graphs are only rarely used when it comes to handwriting recognition. One possible reason for this observation might be the increased complexity of many algorithmic procedures that take graphs, rather than feature vectors, as their input. However, with the rise of efficient graph kernels and fast approximative graph matching algorithms, graph-based handwriting representation could become a versatile alternative to traditional methods. This paper aims at making a seminal step towards promoting graphs in the field of handwriting recognition. In particular, we introduce a set of six different graph formalisms that can be employed to represent handwritten word images. The different graph representations for words, are analysed in a classification experiment (using a distance based classifier). The results of this word classifier provide a benchmark for further investigations.

Keywords: Graph Benchmarking Dataset, Graph Repository, Graph Representation for Handwritten Words

1 Introduction

Structural pattern recognition is based on sophisticated data structures for pattern representation such as strings, trees, or graphs[4]. Graphs are, in contrast with feature vectors, flexible enough to adapt their size to the complexity of individual patterns. Furthermore, graphs are capable to represent structural relationships that might exist between subparts of the underlying pattern (by means of edges). These two benefits turn graphs into a powerful and flexible representation formalism, which is actually used in diverse fields [1, 2].

The computation of a dissimilarity between pairs of graphs, termed graph matching, is a basic requirement for pattern recognition. In the last four decades

[4] Strings and trees can be seen as special cases of graphs.

quite an arsenal of algorithms has been proposed for the task of graph matching [1, 2]. Moreover, also different benchmarking datasets for graph-based pattern recognition have been made available such as ARG [3], IAM [4], or ILPIso [5]. These dataset repositories consist of synthetically generated graphs as well as graphs that represent real world objects.

Recently, graphs have gained some attention in the field of handwritten document analysis [4] like for instance handwriting recognition [6], keyword spotting [7–9], or signature verification [10, 11]. However, we still observe a lack of publicly available graph datasets that are based on handwritten word images. The present paper tries to close this gap and presents a twofold contribution. First, we introduce six novel graph extraction algorithms applicable to handwritten word images. Second, we provide a benchmark database for word classification that is based on the George Washington letters [12, 13].

The remainder of this paper is organised as follows. In Sect. 2, the proposed graph representation formalisms are introduced. In Sect. 3, an experimental evaluation of the novel graph representation formalisms is given on the George Washington dataset. Section 4 concludes the paper and outlines possible further research activities.

2 Graph-based Representation of Word Images

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 either be *undirected* or *directed*, depending on whether pairs of nodes are connected by undirected or directed edges. Additionally, graphs are often divided into *unlabelled* and *labelled* graphs. In the former case we assume empty label alphabets (i.e. $L_v = L_e = \{\}$), and in the latter case, nodes and/or edges can be labelled with an arbitrary numerical, vectorial, or symbolic label.

Different processing steps are necessary for the extraction of graphs from word images. In the framework presented in this paper the document images are first preprocessed by means of *Difference of Gaussian (DoG)*-filtering and binarisation to reduce the influence of noise [14]. On the basis of these preprocessed document images, single word images are automatically segmented from the document and labelled with a ground truth[5]. Next, word images are skeletonised by a 3×3 thinning operator [15]. We denote segmented word images that are binarised and filtered by B. If the image is additionally skeletonised we use the term S.

Graph-based word representations aim at extracting the inherent characteristic of these preprocessed word images. Figure 1 presents an overview of the six different graph representations, which are thoroughly described in the next three subsections. All of the proposed extraction methods result in graphs where the nodes are labelled with two-dimensional attributes, i.e. $L_v = \mathbb{R}^2$, while edges remain unlabelled, i.e. $L_e = \{\}$.

[5] The automatic segmentation is visually inspected and - if necessary - manually corrected. Hence, in our application we assume perfectly segmented words.

In any of the six cases, graphs are normalised in order to reduce the variation in the node labels $(x, y) \in \mathbb{R}^2$ that is due to different word image sizes. Formally, we apply the following transformation to the coordinate pairs (x, y) that occur on all nodes of the current graph.

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

where μ_x, μ_y and σ_x, σ_y denote the mean values and the standard deviations of all node labels in the current graph (in x- and y-direction, respectively).

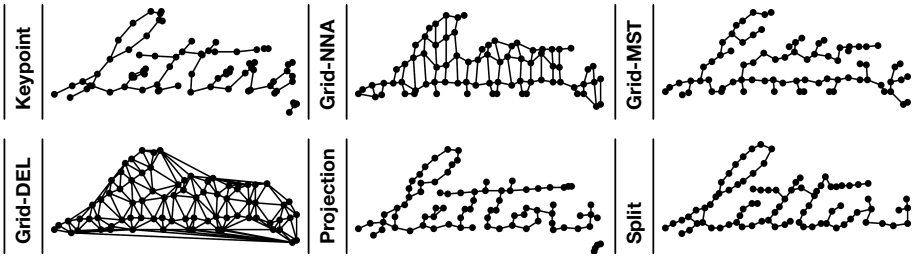

Fig. 1. Different graph representations of the word "Letters"

2.1 Graph Extraction Based on Keypoints

The first graph extraction algorithm is based on the detection of specific keypoints in the word images. Keypoints are characteristic points in a word image, such as for instance end- and intersection-points of strokes. The proposed approach is inspired by [16] and is actually used for keyword spotting in [17]. In the following, Algorithm 1 and its description are taken from [17].

Graphs are created on the basis of filtered, binarised, and skeletonised word images S (see Algorithm 1 denoted by `Keypoint` from now on). 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 [15] 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 ∈ S do
3:         V = V ∪ {(x, y) ∈ CC | (x, y) are end- or junction points}
4:         Remove junction points from CC
5:         for Each connected subcomponent CC_sub ∈ CC do
6:             V = V ∪ {(x, y) ∈ CC_sub | (x, y) are points in equidistant intervals D}
7:     for Each pair of nodes (u, v) ∈ V × V do
8:         E = E ∪ (u, v) if the corresponding points are connected in S
9:     return g = (V, E)
```

2.2 Graph Extraction Based on a Segmentation Grid

The second graph extraction algorithm is based on a grid-wise segmentation of word images. Grids have been used to describe features of word images like *Local Gradient Histogram (LGH)* [18] or *Histogram of Oriented Gradients (HOG)* [19]. However, to the best of our knowledge grids have not been used to represent word images by graphs.

Graphs are created on the basis of binarised and filtered, yet not skeletonised, word images B (see Algorithm 2). First, the dimension of the segmentation grid, basically defined by the number of columns C and rows R, is derived (see line 2 and 3 of Algorithm 2). Formally, we compute

$$C = \frac{\text{Width of } B}{w} \text{ and } R = \frac{\text{Height of } B}{h},$$

where w and h denote the user defined width and height of the resulting segments.

Next, a word image B is divided into $C \times R$ segments of equal size. For each segment s_{ij} $(i = 1, \ldots, C; j = 1, \ldots, R)$ a node is inserted into the resulting graph and labelled by the (x, y)-coordinates of the centre of mass (x_m, y_m) (see line 4). Formally, we compute

$$x_m = \frac{1}{n} \sum_{w=1}^{n} x_w \text{ and } y_m = \frac{1}{n} \sum_{w=1}^{n} y_w, \tag{1}$$

where n denotes the number of foreground pixel in segment s_{ij}, while x_w and y_w denote the x- and y-coordinates of the foreground pixels in s_{ij}. If a segment does not contain any foreground pixel, no centre of mass can be determined and thus no node is created for this segment.

Finally, undirected edges (u, v) are inserted into the graph according to one out of three edge insertion algorithms, viz. *Node Neighbourhood Analysis (NNA)*, *Minimal Spanning Tree (MST)*, or *Delaunay Triangulation (DEL)*. The first algorithm analyses the four neighbouring segments on top, left, right, and bottom of a node $u \in V$. In case a neighbouring segment of u is also represented by a

node $v \in V$, an undirected edge (u, v) between u and v is inserted into the graph. The second algorithm reduces the edges inserted by the *Node Neighbourhood Analysis* by means of a *Minimal Spanning Tree* algorithm. Hence, in this case the graphs are actually transformed into trees. Finally, the third algorithm is based on a *Delaunay Triangulation* of all nodes $u \in V$. We denote this algorithmic procedure by `Grid-NNA`, `Grid-MST`, and `Grid-DEL` (depending on which edge insertion algorithm is employed).

Algorithm 2 Graph Extraction Based on a Segmentation Grid

Input: Binary image B, Grid width w, Grid height h
Output: Graph $g = (V, E)$ with nodes V and edges E

1: **function** GRID(B,w,h)
2: **for** $i \leftarrow 1$ to number of columns $C = \frac{\text{Width of } B}{w}$ **do**
3: **for** $j \leftarrow 1$ to number of rows $R = \frac{\text{Height of } B}{h}$ **do**
4: $V = V \cup \{(x_m, y_m) \mid (x_m, y_m)$ is the centre of mass of segment $s_{ij}\}$
5: **for** Each pair of nodes $(u, v) \in V \times V$ **do**
6: $E = E \cup (u, v)$ if associated segments are connected by *NNA*, *MST*, or *DEL*
7: **return** g

2.3 Graph Extraction Based on Projection Profiles

The third graph extraction algorithm is based on an adaptive rather than a fixed segmentation of word images. That is, the individual word segment sizes are adapted to respect to projection profiles. Projection profiles have been used for skew correction [20] and feature vectors of word images [21], to name just two examples. However, to the best of our knowledge projection profiles have not been used to represent word images by graphs.

Graphs are created on the basis of binarised and filtered word images B (see Algorithm 3, denoted by `Projection` from now on). First, a histogram of the vertical projection profile $P_v = \{p_1, \ldots, p_{max}\}$ is computed, where p_i represents the frequency of foreground pixels in column i of B and max is the width of B (see line 2 of Algorithm 3). Next, we split B vertically by searching so called white spaces, i.e. subsequences $\{p_i, \ldots, p_{i+k}\}$ with $p_i = \ldots = p_{i+k} = 0$. To this end, we split B in the middle of white spaces, i.e. position $p = \lfloor (p_i + p_{i+k})/2 \rfloor$, into n segments $\{s_1, \ldots, s_n\}$ (see line 3). In the best possible case a segment encloses word parts that semantically belong together (e.g. characters). Next, further segments are created in equidistant intervals D_v when the width of a segment $s \in B$ is greater than D_v (see line 4 and 5).

The same procedure as described above is then applied to each (vertical) segment $s \in B$ (rather than whole word image B) based on the projection profile of rows (rather than columns) (see lines 6 to 10). Thus, each segment s is individually divided into horizontal segments $\{s_1, \ldots, s_n\}$ (Note that a user defined parameter D_h controls the number of additional segmentation points (similar to D_v)). Subsequently, for each segment $s \in B$ a node is inserted

into the resulting graph and labelled by the (x, y)-coordinates of the centre of mass (x_m, y_m) (see (1) as well as line 11 and 12). If a segment consists of background pixels only, no centre of mass can be determined and thus no node is created for this segment.

Finally, an undirected edge (u, v) between $u \in V$ and $v \in V$ is inserted into the graph for each pair of nodes, if the corresponding pair of segments is directly connected by a chain of foreground pixels in the skeletonised word image S (see line 13 and 14).

Algorithm 3 Graph Extraction Based on Projection Profiles

Input: Binary image B, Skeleton image S, Ver. threshold D_v, Hor. threshold D_h
Output: Graph $g = (V, E)$ with nodes V and edges E

```
 1: function PROJECTION(B,S,D_v,D_h)
 2:     Compute vertical projection profile P_v of B
 3:     Split B vertically at middle of white spaces of P_v into {s_1, ..., s_n}
 4:     for Each segment s ∈ B with width larger D_v do
 5:         Split s vertically in equidistant intervals D_v into {s_1, ..., s_n}
 6:     for Each segment s ∈ B do
 7:         Compute horizontal projection profile P_h of s
 8:         Split s horizontally at middle of white spaces of P_h into {s_1, ..., s_n}
 9:         for Each segment s ∈ {s_1, ..., s_n} with height larger D_h do
10:             Split s horizontally in equidistant intervals D_h into {s_1, ..., s_n}
11:     for Each segment s ∈ B do
12:         V = V ∪ {(x_m, y_m) | (x_m, y_m) is the centre of mass of segment s}
13:     for Each pair of nodes (u, v) ∈ V × V do
14:         E = E ∪ (u, v) if the corresponding segments are connected in S
15:     return g
```

2.4 Graph Extraction Based on Splittings

The fourth graph extraction algorithm is based on an adaptive and iterative segmentation of word images by means of horizontal and vertical splittings. Similar to `Projection`, the segmentation is based on projection profiles of word images. Yet, their algorithmic procedures clearly distinguishes from each other. To the best of our knowledge such a split-based segmentation has not been used to represent word images by graphs.

Graphs are created on the basis of binarised and filtered word images B (see Algorithm 4, denoted by `Split` from now on). Thus, each segment $s \in B$ (initially B is regarded as one segment) is iteratively split into smaller subsegments until the width and height of each segment in $s \in B$ is below a certain threshold D_w and D_h, respectively (see lines 2 to 12). Formally, each segment $s \in S$ (with width greater than threshold D_w) is vertically subdivided into subsegments $\{s_1, \ldots, s_n\}$ by means of the projection profile P_v of s (for further details we refer to Sect. 2.3). If the histogram P_v contains no white spaces, i.e. $\forall h_i \in P \neq 0$, the segment s is

split in its vertical centre into $\{s_1, s_2\}$ (see lines 3 to 7). Next, the same procedure as described above is applied to each segment $s \in B$ (with height greater than threshold D_h) in the horizontal, rather than vertical, direction (see lines 8 to 12).

Once no segment from $s \in B$ can further be split, the centre of mass (x_m, y_m) (see (1)) is computed for each segment $s \in B$ and a node is inserted into the graph labelled by the (x, y)-coordinates of the closest point on the skeletonised word image S to (x_m, y_m) (see lines 13 and 14). If a segment consists of background pixels only, no centre of mass can be determined and thus no node is created for this segment.

Finally, an undirected edge (u, v) between $u \in V$ and $v \in V$ is inserted into the graph for each pair of nodes, if the corresponding pair of segments is directly connected by a chain of foreground pixels in the skeletonised word image S (see line 15 and 16).

Algorithm 4 Graph Extraction Based on Splittings

Input: Binary image B, Skeleton image S, Width threshold D_w, Height threshold D_h
Output: Graph $g = (V, E)$ with nodes V and edges E

```
 1: function SPLIT(B,S,D_w,D_h)
 2:     while Any segment s ∈ B has a width larger D_w or height larger D_h do
 3:         for Each segment s ∈ B with width larger D_w do
 4:             if s contains white spaces in vertical projection profile P_v then
 5:                 Split s vertically at middle of white spaces of P_v into {s_1, ..., s_n}
 6:             else
 7:                 Split s vertically at vertical centre of s into {s_1, s_2}
 8:         for Each segment s ∈ B with height larger D_h do
 9:             if s contains white spaces in horizontal projection profile P_h then
10:                 Split s horizontally at middle of white spaces of P_h into {s_1, ..., s_n}
11:             else
12:                 Split s horizontally at horizontal centre of s into {s_1, s_2}
13:     for Each segment s ∈ B do
14:         V = V ∪ {(x_m, y_m) | (x_m, y_m) is the centre of mass of segment s}
15:     for Each pair of nodes (u, v) ∈ V × V do
16:         E = E ∪ (u, v) if the corresponding segments are connected in S
17:     return g
```

3 Experimental Evaluation

The proposed graph extraction algorithms are evaluated on preprocessed word images of the George Washington (GW) dataset, which consists of twenty different multi-writer letters with only minor variations in the writing style[6]. The same documents have been used in [12, 13].

[6] 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`

For our benchmark dataset a number of perfectly segmented word images is divided into three independent subsets, viz. a training set (90 words), a validation set (60 words), and a test set (143 words)[7]. Each set contains instances of thirty different words. The validation and training set contain two and three instances per word, respectively, while the test contains at most five and at least three instances per word. For each word image, one graph is created by means of the six different graph extraction algorithms (using different parameterisations).

In Table 1 an overview of the validated meta-parameters for each graph extraction method is given. Roughly speaking, small meta-parameter values result in graphs with a higher number of nodes and edges, while large meta-parameter values result in graph with a smaller number of nodes and edges.

Table 1. Validated meta-parameters of each graph extraction algorithm

Graph Extraction Algorithm	Validated Meta-Parameter Values
`Keypoint`	$D = \{2, 3, 4, 5, 6\}$
`Grid-NNA`	$w = \{7, 9, 11, 13, 15\} \times \quad h = \{7, 9, 11, 13, 15\}$
`Grid-MST`	$w = \{7, 9, 11, 13, 15\} \times \quad h = \{7, 9, 11, 13, 15\}$
`Grid-DEL`	$w = \{7, 9, 11, 13, 15\} \times \quad h = \{7, 9, 11, 13, 15\}$
`Projection`	$D_v = \{5, 7, 9, 11\} \quad \times D_h = \{4, 6, 8, 10\}$
`Split`	$D_w = \{3, 5, 7, 9\} \quad \times D_h = \{7, 9, 11, 13\}$

The quality of the different graph representation formalisms is evaluated by means of the accuracy of a kNN-classifier[8] that operates on approximated *Graph Edit Distances (GED)* [22]. The meta-parameters are optimised with respect to the accuracy of the kNN on the validation set[9]. Then, the accuracy of the kNN-classifier is measured on the test set using the optimal meta-parameters for each graph extraction method.

In Table 2, the optimal meta-parameters, the median number of nodes $|\bar{V}|$ and edges $|\bar{E}|$ (defined over training, validation and test set) as well as the accuracy of the kNN on the test set are shown for each extraction method. We observe that the `Projection` and `Split` extraction methods clearly perform the best among all algorithms with a classification accuracy of about 82% and 80% on the test set, respectively. `Keypoint` achieves the third best classification result with about 77%. However, the average number of nodes and edges of both `Projection` and `Split` are substantially lower than those of `Keypoint`. `Grid-MST` achieves an accuracy which is virtually the same as with `Keypoint`. Yet, with substantially less nodes and edges than `Keypoint`. The worst classification accuracies are obtained with `Grid-NNA` and `Grid-DEL` (about 65% and 63%, respectively). Note especially the large number of edges which are produced with the Delaunay triangulation.

[7] Available at `http://www.histograph.ch`

[8] We define $k = 5$ for all of our evaluations.

[9] If different meta-parameter settings lead to the same accuracy, the setting with the lower average number of nodes and edges is used.

Table 2. Classification accuracy of each graph representation formalism

Graph Extraction Algorithm	Optimal Meta-Parameter	$\lvert\bar{V}\rvert$	$\lvert\bar{E}\rvert$	Acc.
`Keypoint`	$D = 4$,	73	67	0.7762
`Grid-NNA`	$w = 13$, $h = 9$	39	55	0.6503
`Grid-MST`	$w = 9$, $h = 11$	46	44	0.7413
`Grid-DEL`	$w = 9$, $h = 9$	52	138	0.6294
`Projection`	$D_v = 9$ $D_h = 6$	44	41	0.8182
`Split`	$D_w = 7$ $D_h = 9$	51	48	0.8042

4 Conclusion and Outlook

The novel graph database presented in this paper is based on graph extraction methods. These methods aim at extracting the inherent characteristics of handwritten word images and represent these characteristics by means of graphs. The `Keypoint` extraction method is based on the representation of nodes by characteristic points on the handwritten stroke, while edges represent strokes between these keypoints. Three of our extraction methods, viz. `Grid-NNA`, `Grid-MST` and `Grid-DEL`, are based on a grid-wise segmentation of a word image. Each segment of this grid is represented by a node which is then labelled by the centre of mass of the segment. Finally, the `Projection` and `Split` extraction methods are based on vertical and horizontal segmentations by means of projection profiles. An empirical evaluation of the six extraction algorithm is carried out on the George Washington letters. The achieved accuracy can be seen as a first benchmark to be used for future experiments. Moreover, the experimental results clearly indicate that both `Projection` and `Split` are well suited for extracting meaningful graphs from handwritten words.

In future work we plan to extend our graph database to further documents using the presented extraction methods and make them all publicly available. Thus, we will be able to provide a more comparative and thorough study against other state-of-the-art representation formalisms at a later stage.

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

References

1. 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
2. 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)
3. De Santo, M., Foggia, P., Sansone, C., Vento, M.: A large database of graphs and its use for benchmarking graph isomorphism algorithms. Pattern Rec. Letters **24**(8) (2003) 1067–1079
4. Bunke, H., Riesen, K.: Recent advances in graph-based pattern recognition with applications in document analysis. Pattern Rec. **44**(5) (2011) 1057–1067

5. Le Bodic, P., Héroux, P., Adam, S., Lecourtier, Y.: An integer linear program for substitution-tolerant subgraph isomorphism and its use for symbol spotting in technical drawings. Pattern Rec. **45**(12) (2012) 4214–4224
6. Fischer, A., Suen, C.Y., Frinken, V., Riesen, K., Bunke, H.: A fast matching algorithm for graph-based handwriting recognition. In: Graph-Based Rep. in Pattern Rec. Volume 7877. (2013) 194–203
7. 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
8. 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
9. 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
10. Wang, K., Wang, Y., Zhang, Z.: On-line Signature Verification Using Segment-to-Segment Graph Matching. In: Int. Conf. on Doc. Anal. and Rec. (2011) 804–808
11. Fotak, T., Bača, M., Koruga, P.: Handwritten signature identification using basic concepts of graph theory. Trans. on Signal Processing **7**(4) (2011) 117–129
12. 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
13. Fischer, A., Keller, A., Frinken, V., Bunke, H.: Lexicon-free handwritten word spotting using character HMMs. Pattern Rec. Letters **33**(7) (2012) 934–942
14. 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
15. Guo, Z., Hall, R.W.: Parallel thinning with two-subiteration algorithms. Communications of the ACM **32**(3) (1989) 359–373
16. 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
17. Stauffer, M., Fischer, A., Riesen, K.: Graph-based Keyword Spotting in Historical Handwritten Documents. In: Int. W. on Structural and Syntactic Pattern Rec. (2016)
18. Rodriguez, J.A., Perronnin, F.: Local gradient histogram features for word spotting in unconstrained handwritten documents. In: Int. Conf. on Frontiers in Handwriting Rec. (2008) 7–12
19. Almazán, J., Gordo, A., Fornés, A., Valveny, E.: Segmentation-free word spotting with exemplar SVMs. Pattern Rec. **47**(12) (2014) 3967–3978
20. Hull, J.: Document image skew detection: Survey and annotated bibliography. In: Series in Machine Perception and Artif. Intell. Volume 29. (1998) 40–64
21. Rath, T., Manmatha, R.: Word image matching using dynamic time warping. In: Comp. Vision and Pattern Rec. Volume 2. (2003) II–521–II–527
22. Riesen, K., Bunke, H.: Approximate graph edit distance computation by means of bipartite graph matching. Image and Vision Computing **27**(7) (2009) 950–959