Handwritten Word Spotting by Inexact Matching of Grapheme Graphs

Pau Riba, Josep Lladós, Alicia Fornés
Computer Vision Center - Computer Science Department
Universitat Autònoma de Barcelona
Bellaterra, Catalonia, Spain
Email: {priba,josep,afornes}@cvc.uab.cat

***Abstract*—This paper presents a graph-based word spotting for handwritten documents. Contrary to most word spotting techniques, which use statistical representations, we propose a structural representation suitable to be robust to the inherent deformations of handwriting. Attributed graphs are constructed using a part-based approach. Graphemes extracted from shape convexities are used as stable units of handwriting, and are associated to graph nodes. Then, spatial relations between them determine graph edges. Spotting is defined in terms of an error-tolerant graph matching using bipartite-graph matching algorithm. To make the method usable in large datasets, a graph indexing approach that makes use of binary embeddings is used as preprocessing. Historical documents are used as experimental framework. The approach is comparable to statistical ones in terms of time and memory requirements, especially when dealing with large document collections.**

I. Introduction

Word spotting has emerged in the last years as a highly effective technique for large scale document image retrieval in manuscript databases. In some cases, in particular in historical documents, a strategy based on full transcription using handwriting recognition approaches, and a subsequent search is nowadays far from being feasible, hence word spotting is a pratical alternative. Most of the existing word spotting techniques use statistical representations (e.g. SIFT, HOG) of the word images [1], [2]. However, some structural representations have been recently proposed. The main motivation is that the nature of handwriting suggests than the structure is more stable than the pure appearance of the strokes. This is especially important when dealing with the elastic deformations of different handwriting styles. As stated in the comparison of statistical versus structural representations for handwritten word spotting reported in [3], the main disadvantages of structural approaches are the time complexity and scalability to large document collections.

Graphs are robust representations able to describe shapes in terms of the relationships between constituent parts or primitives. Several graph matching algorithms have been proposed in the literature [4]. In the particular case of handwritten word images represented by graphs, error-tolerant or inexact graph matching is required to cope with the deformation of the strokes. Roughly speaking, the problem consists in finding the minimum transformation cost of one of the graphs such that an isomorphism exists between the transformed graph and the second one. One of the most popular error-tolerant graph matching methods is based on graph edit distance [5], but inexact (sub)graph isomorphism computation is a known NP-Complete problem. Approximate or suboptimal variations of graph edit distance have been proposed to overcome this difficulty [6]. One of the first attemps to use graph matching for word spotting was proposed by Fischer et al. [7] defining a HMM model on a graph structure, but since only the graph nodes are used the structure is not accurately matched. Wang et al. [8] proposed a coarse-to-fine graph matching. First a bag-of-small-graphs approach is used to find words likely to be the query one, afterwards an inexact matching algorithm is employed to verify the true positives. Riesen et al. [9] study if graph matching techniques can be beneficially employed for keyword spotting. These approaches are still far away of being able to cope with large databases in an efficient way.

In general, the use of graph matching for word spotting has to deal with two factors. First, a robust representation able to tolerate the deformations of handwriting without losing the expressiveness in terms of the topology. Second, to overcome the computational cost of traditional graph matching algorithms. In this work we propose contributions to tackle the two problems. First, we propose a graph representation in terms of a codebook of graphemes, the stable constituent units of handwriting, and their spatial relationships. Second an error-tolerant graph matching is used to spot query words. Word spotting can be considered segmentation-free, i.e. words do not have to be segmented in document images, since the query word graph is found as a subgraph of the large graph representing the whole document. To speed up the process and be able to cope with large scale graphs, an indexation strategy is used. Thus, local configurations of graph nodes are represented as binary codes so a hashing strategy can locate subgraphs likely to match the query word graph.

The rest of this paper is organized as follows. First, section II describes the graph model used to represent handwritten words. In section III the approach for word spotting based on graph matching is presented. Section IV presents the proposed indexation method. Section V reports on the experimental evaluation. Finally section VI draws the conclusions.

II. Graph construction

An *attributed graph* G is defined as a 4-pla $G = (V, E, L_V, L_E)$ where V is the set of nodes; $E \subseteq V \times V$ is the set of edges; L_V and L_E are two labeling functions defined as $L_V : V \rightarrow \Sigma_V \times A_V^k$ and $L_E : E \rightarrow \Sigma_E \times A_E^l$, where Σ_V and Σ_E are two sets of symbolic labels for vertices and edges, respectively, A_V and A_E are two sets of attributes for vertices and edges, respectively, and $k, l \in \mathbb{N}$. We will denote the

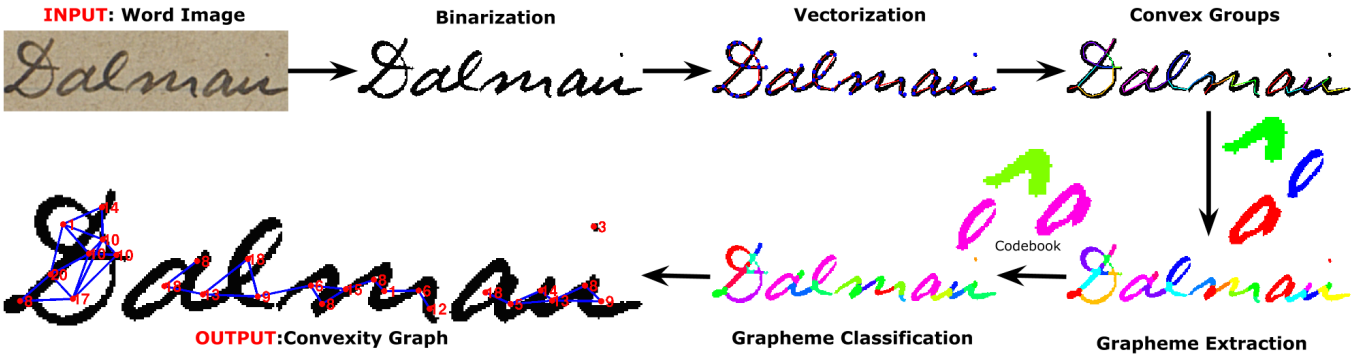


Fig. 1. Outline of the graph construction.

number of vertices in a graph by $|V|$ and the number of edges by $|E|$. We use for the sake of simplicity $G = (V, E)$ instead of $G = (V, E, L_V, L_E)$. To keep the graphs stable against the deformations of handwriting but robust enough to represent the topology of words in terms of their constituent primitives, we proposed a representation based on graphemes graphs. A grapheme is the smallest unit used in describing the writing system of a language. Several works in the literature represent handwriting in terms of graphemes [10] or allographs [11]. Thus, a codebook of graphemes is extracted after text images are oversegmented at subcharacter level. Graph nodes are associated to grapheme codewords, and graph edges represent adjacency and proximity relations.

The graph is extracted as follows (see Fig. 1). First, the codebook of graphemes is extracted. The graphemes have been defined as a convex path in the vectorial approximation of the skeleton graph (see Fig. 2). To extract the convex paths, the mathematical definition is used, hence for every pair of points within the convex group, the straight line segment that joins them is also within the object. Graphemes are defined as the part of the image foreground extracted from the geodesic reconstruction from the convex groups. A codebook of graphemes is defined according to a clustering in terms of the Blurred Shape Model (BSM) descriptor [12]. The clusters have been generated using all the graphemes in the database and the k-means algorithm with 20 different classes.

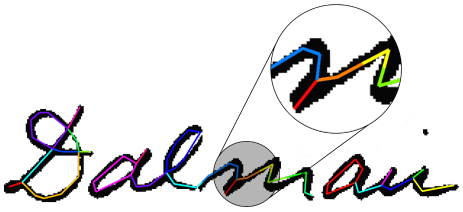


Fig. 2. Graphemes are extracted from convex groups of the skeleton.

As a result, a graph G is generated. Graph nodes correspond to graphemes, attributed with the corresponding label codeword of the codebook of graphemes. Graph edges represent adjacency relations between nodes (the corresponding graphemes are adjacent). The edge attributes are: the number of points in common between the corresponding convex groups (we will refer it as edge weight) the angle and the length.

III. Graph Matching for Word Spotting

Error-tolerant matching aims to establish a (sub)graph isomorphism that may include some distortions. The type of distortions that are considered are application dependent. A typical distortion model is based on string matching. It includes the insertion, deletion and substitution of both vertices and edges. These distortions are called edit operations which have an associated cost. Error-tolerant graph matching is computed by searching the sequence of edit operation with minimum cost that transforms one graph into another. Usually, *branch and bound* techniques are used to compute the minimum cost edit sequence from one graph to another. Bipartite graph matching [6] is one of the most efficient methods for error-tolerant graph matching. It is based on defining a matrix of edit costs between the nodes of both graphs. The best correspondence between nodes is found by a linear assignment method.

The matrix definition for the bipartite graph matching takes into consideration both the local structure of the vertices and their attributes. Let $G_w = (V_w, E_w, L_{V_w}, L_{E_w})$ be the graph of the query word and $G_t = (V_t, E_t, L_{V_t}, L_{E_t})$ be the target graph with $V_w = \{u_1, \ldots, u_n\}$ and $V_t = \{v_1, \ldots, v_m\}$, respectively. The cost matrix C is defined as:

$$C = \left[\begin{array}{cccc|cccc} c_{1,1} & c_{1,2} & \cdots & c_{1,m} & c_{1,\varepsilon} & \infty & \cdots & \infty \\ c_{2,1} & c_{2,2} & \cdots & c_{2,m} & \infty & c_{2,\varepsilon} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \infty \\ c_{n,1} & c_{n,2} & \cdots & c_{n,m} & \infty & \cdots & \infty & c_{n,\varepsilon} \\ \hline c_{\varepsilon,1} & \infty & \cdots & \infty & 0 & 0 & \cdots & 0 \\ \infty & c_{\varepsilon,2} & \ddots & \vdots & 0 & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \infty & \vdots & \ddots & \ddots & 0 \\ \infty & \cdots & \infty & c_{\varepsilon,n} & 0 & \cdots & 0 & 0 \end{array}\right]$$

where $c_{i,j}$ denotes the cost of a node substitution $c(u_i \rightarrow v_j)$, $c_{i,\varepsilon}$ denotes the cost of a node deletion $c(u_i \rightarrow \varepsilon)$, and $c_{\varepsilon,j}$ denotes the costs of a node insertion $c(\varepsilon \rightarrow v_j)$. A suboptimal graph edit distance between G_w and G_t is computed by a linear assignment algorithm [6]. A key decision in the matching algorithm is the definition of the cost functions. In this work, the cost functions are defined as follows:

Insertion and deletion costs. Both, node insertion and deletion costs are computed in the same way (both can be seen as deletions in one graph or the other). Intuitively, the

cost is computed in terms of the local configuration of the node defined by the incident edges. If the node is strongly connected the cost will be higher than for example a simple node that appears disconnected. Thus, the insertion cost has three terms:

$$\begin{aligned} c(\varepsilon \rightarrow v_j) &= c(u_i \rightarrow \varepsilon) \\ &= we_0 C_{weightEdges} + we_1 C_{edges} + we_2 t_{vertices} \end{aligned} \quad (1)$$

where we_i are weighting factors; $C_{weightEdges}$ is the sum of the attributes of the edges incident in the node being deleted, indicating how much the node is sharing part of its grapheme with the neighboring ones; C_{edges} is a measure of the density of the node computed as the ratio between the number of incident edges and the total number of graph nodes; $t_{vertices}$ is a constant value experimentally set as a baseline cost for the edit operation.

Substitution costs. Computed in terms of the position of the nodes, their label according to the codebook and the similarity of the local structure. The different terms are weighted by factors denoted with wn_i. Formally:

$$c(u_i \rightarrow v_j) = wn_0 D_{i,j} + wn_1 C_{BSM} + wn_2 C_{local\_structure} \quad (2)$$

where wn_i are different weights. $D_{i,j}$ is the euclidean distance between the spatial position of the nodes u_i and v_j position. This distance is normalized by the maximum node position of the both graphs. C_{BSM} is the L_1 distance between the corresponding BSM shape descriptor of the node graphemes. The value C_{BSM} is computed with the distance between the centroids of the k-means clustering when the codebook is extracted. Finally, $C_{local\_structure}$ is the edit operation cost on the incident edges.

Matching of the incident edges. In order to compute the edit cost on the adjacent edges ($C_{local\_structure}$), the Bipartite Graph Matching algorithm has been used. Firstly a matrix of edit costs between the adjacent edges of both nodes is defined Ce with the same structure as C. In this case the cost of edge insertion and deletion is a constant t_{edges}. The substitution costs are computed in terms of the edge attributes. i.e. weight, angle and length for the both edges to substitute.

$$c(e_i \rightarrow f_j) = we_0 C_{weight} + we_1 C_{angle} + we_2 C_{length} \quad (3)$$

where we_i are weighting factors, C_{weight} is the difference between the weight of the two edges, C_{angle} is the angle between them and C_{length} is equivalent to $1 - e_{short}/e_{long}$ where e_{short} denotes the length of the shorter edge and e_{long} the length of the longer one.

The values of the weights and the constant values have been empirically set as follows: $t_{vertices} = t_{edges} = 0.5$ and $wn_0 = 2/5$, $wn_1 = 1/5$, $wn_2 = 2/5$, $we_0 = 1/5$, $we_1 = 2/5$ and $we_2 = 2/5$. All the cost computations are scaled into the range $(0, 1)$, and the sum of the partial costs involved in the computation of any cost is 1.

With the above considerations, given a query word represented by a graph G_w and a set of target images of handwritten documents represented by graphs G_t^i, word spotting is computed as an inexact subgraph matching where several subgraphs of G_t^i G_w are found as being similar to G_w. The edit cost is computed using the bipartite graph matching algorithm with the costs defined before. In terms of size, graphs corresponding to query words have an average of 25 nodes, and a graph representing a page of a document has an average of 4,500 nodes. It means that since graph matching is a computationally costly process, it might result in unrealistic elapsed time responses from the application point of view. To tackle this problem, we propose a graph indexing process previous to the graph matching.

IV. Graph indexing

As it has been mentioned before, error-tolerant graph matching is a computationally costly process. Spotting words represented by graphs in large scale document collections would require an individual matching between the query graph and all the ones of the database. To speed up this process we propose a graph hashing approach. It can be seen as a preprocess that given a query word graph, retrieves subgraphs of the targets graphs (documents) likely to be similar. This indexation strategy is designed with the aim to achieve a maximum recall. Afterwards, the matching algorithm described in section III filters out the retrieved subgraphs discarding the false positives. This coarse-to-fine strategy has two advantages. First, it allows to compute the spotting in a fast way. Second, word spotting can be considered as segmentation free, i.e. it is not necessary to segment the words in the document prior to the matching.

The indexation approach is constructed in terms of a hashing architecture defined as graph node binary embeddings [13]. Thus, graph node attributes are complemented by vectors of attributes characterizing their context. We refer to the context of a node as the local topology of it. In graph theory, the *Morgan index* is a well-known feature to compute the local context of a node. The Morgan index of order k associated to a given node v counts the number of paths of length k incident in node v and starting somewhere in the graph. Let us denote as $M_l(v, k)$ the Morgan index of node v, order k and label l which counts the number of paths of length k incident at node v and starting at nodes labeled as l. The label l of a node is the label codeword as it has been defined in section II. Hence, the *context* of a node v is defined as:

$$\begin{aligned} \nu(v) = [&M_{l_1}(v,1), \ldots, M_{l_1}(v,K), M_{l_2}(v,1), \ldots, \\ &M_{l_2}(v,K), \ldots, M_{l_{|\Sigma_V|}}(v,K)], \end{aligned}$$

where K is the maximum length of the paths incident in v that is considered. The value of K depends on each experimental setup. In this work we have set $K = 3$. Each context attribute vector is converted to a binary code applying a thresholding function. Thus, graph indexing is formulated in terms of finding target graphs in the database whose nodes have a small Hamming distance from the query nodes. This process is implemented with an inverted file indexing structure with binary-valued hash functions. In other words, the binary vectors attributing the nodes of the query word graph G_w are used as hashing keys into the inverted file indexing structure, retrieving those nodes of the target graphs G_t^i with a small Hamming distance.

The final step consists in finding the retrieved subgraphs of the target graphs in terms of the retrieved nodes. We define a *partition* P of a graph G as a decomposition of it in n small subgraphs. With this idea, the node indexing in

terms of binary codes is reformulated as a voting process into the partitions of the target graphs, seen as voting bins, previously defined. The final subgraphs that are retrieved are those induced by the partitions that receive a high number of votes. These subgraphs can be seen as candidates to contain the query word that is spotted. Thus, the definition of partitions is important so they represent graphs associated to potential words. In the experiments described in section V we have defined as partitions the subgraphs induced by a rough word segmentation using the method described in [14]. As it has been mentioned, the indexing step is seen as a preprocess to the matching described in section III with the aim of speeding up the process and selecting the regions of interest of the images (subgraphs) candidate to contain the query word.

V. Results

The proposed approach has been evaluated in two different scenarios: segmentation-based and segmentation-free word spotting. The first experiment consists in a segmentation-based word spotting scenario. We have used a set of pre-segmented words with the aim of comparing our approach with other methods in the literature [3], [15]. In particular, we show that a graph-based word spotting achieves good performance in comparison to other well-known approaches. The second experiment is addressed to evaluate the complete pipeline, using a graph indexation based on binary embedding to boost graph-based word spotting method in large datasets. In both experiments, we have used a subset of the Barcelona Historical Handwritten Marriages Database (BH2M) [16], which was written in the 17th century. The performance has been measured by the mean Average Precision (mAP), which is defined as follows:

$$\text{mAP} = \frac{\sum_{n=1}^{|\text{ret}|} P@n \times r(n)}{|\text{rel}|}$$

where $P@n$ is the precision at n, $r(n)$ is a binary function on the relevance of the *n*-th item in the returned ranked list, *rel* is the set of the relevant objects with regard to the query, and *ret* is the set of retrieved elements from the dataset.

A. Graph-based word spotting performance

In this experiment we have evaluated the performance of our method using the 27 pages used in [3]. This set contains 6,544 segmented words from 1,751 different transcriptions. All the words having at least three characters and appearing at least ten times have been selected as queries. Thus, there are 514 queries corresponding to 32 different words.

Table I shows the quantitative results and compares them to some other methods in the literature. The proposed approach outperforms most of the methods representing classical families of approaches in the literature (statistical, structural, pseudo-structural). We must notice that the aim of this work is to propose graph matching as a valid alternative for word spotting, in front of the more widespread techniques usually inspired by statistical pattern recognition.

Some qualitative results are shown in Fig. 3. It is interesting to notice that most words have been correctly retrieved. This example takes the name *Farrer* as a query. The system correctly retrieves the first 11th words, whereas the 12th retrieved word corresponds to the name *Barrer*. This word indeed corresponds to a very similar word: it has the same character length, and only one different letter.

Method	mAP
DTW [3]	19.20
Graph-based [15]	24.60
BoVW [3]	30.00
Loci-based [17]	40.06
nrHOG [18]	56.06
Proposed	51.62

TABLE I. Word Spotting results.

Fig. 3. Qualitative results for the query Farrer.

B. Word Spotting indexed for large documents

In this second experiment graph indexation based on binary embedding has been applied to boost the word spotting method, with the aim of dealing with large datasets, reducing the complexity inherent to graph matching. For this experiment, we have selected 11 pages, containing 3,609 words. From them, we have randomly chosen 5 instances for each one of the 8 different words, obtaining a total of 40 query words.

For each one of the query words, the fast matching is used to select the candidate regions in the pages. In this way, only the areas that receive a minimum amount of matches are considered as candidate regions, which is then matched using the bipartite graph-matching. In our experiments, thanks to the indexation and detection of the candidate regions, for each query, we can reduce from 3,609 to an average of 1,254 the number of regions to compare. From these 1,254 candidate regions, an average of 40 regions contain the desired word, whereas 10 words have been missed. In summary, the indexation allows to reduce by 3 the number the volume of target graphs to be matched to the query word graph, showing that the graph-based method can be used in large datasets.

Table II presents the Precision, Recall and the mean Average Precision using the indexation step. Note that in this case, we only compute the matching distance between the query word and the candidate regions. Looking only to the *mAP* column, we notice that two queries are generating a lower value than the others. The special queries are *Eularia* and *defunct*. Studying these particular cases, some typical problems appear:

Binarization Problem: The binarization step can provoke degradations in the generation of the graph. This problem appears in one of the queries *Eularia* (Fig. 4(a)).

Sharing Parts: Two different words that share most of their letters may have a smaller cost than the correct word with a

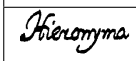
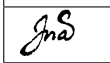
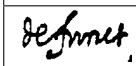
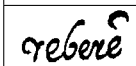
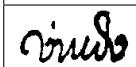
Query	Transcription	Precision	Recall	mAP
	Eularia	0.0080	0.8462	0.7959
	Hieronyma	0.0118	0.7875	0.9329
	Jua$	0.0149	0.5389	0.8490
	defunct	0.0271	0.7886	0.6372
	donsella	0.0420	0.8215	0.9454
	pages	0.0590	0.9352	0.9463
	rebere$	0.0645	0.7676	0.9815
	viudo	0.0133	0.6455	0.9231
	Total	**0.0301**	**0.7664**	**0.8764**

TABLE II. WORD SPOTTING RESULTS BASED ON GRAPH INDEXING.

(a) (b) (c)

Fig. 4. Problems of the word spotting: (a) Binarization, (b) Shared letters, (c) Lexical variations.

different writing style. For example, *Maria* is retrieved when searching the query *Eularia* (Fig. 4(b)).

Lexical Variations: This problem is similar to the last one. In this case, the word appears with different lexical variations, for example for plural or feminine. The query *defunct* has a lower performance than the others due to this fact. (Fig. 4(c)).

In terms of computational cost, although the implementation is not optimized, the elapsed time for indexing a graph corresponding to a page is around 0,02 seconds (target graph of 4,500 nodes). For a query that is searched inside 11 pages, the elapsed time is 0.021 seconds for the indexation and 4.85 seconds for the standard implementation of a bipartite graph-matching. Hence, the time is drastically reduced.

VI. CONCLUSION

In this paper we have proposed a word spotting method based on graph-representations. First, we have shown that graphemes based on convexities can be stable under the deformations of handwriting. Second, we have shown how the graph indexing approach using binary embeddings can deal with large collections and avoid the segmentation of words at the same time. The experimental results demonstrate that our structural approach is comparable to statistical approaches in terms of performance and time requirements.

Future work will focus on the evaluation of the stability of graph-based representations in large multi-writer document collections.

ACKNOWLEDGMENT

This work has been partially supported by the Spanish project TIN2012-37475-C02-02 and the European project ERC-2010-AdG-20100407-269796.

REFERENCES

[1] J. Almazán, A. Gordo, A. Fornés, and E. Valveny, "Word spotting and recognition with embedded attributes," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 36, no. 12, pp. 2552–2566, Dec 2014.

[2] M. Rusiñol, D. Aldavert, R. Toledo, and J. Lladós, "Efficient segmentation-free keyword spotting in historical document collections," *Pattern Recognition*, vol. 48, no. 2, pp. 545–555, 2015.

[3] J. Llados, M. Rusinol, A. Fornes, D. Fernandez, and A. Dutta, "On the influence of word representations for handwritten word spotting in historical documents," *Int. Journal of Pattern Recognition and Artificial Intelligence*, vol. 26, no. 05, 2012.

[4] D. Conte, P. Foggia, C. Sansone, and M. Vento, "Thirty years of graph matching in pattern recognition," *Int. Journal of Pattern Recognition and Artificial Intelligence*, vol. 18, no. 03, pp. 265–298, 2004.

[5] X. Gao, B. Xiao, D. Tao, and X. Li, "A survey of graph edit distance," *Pattern Analysis and applications*, vol. 13, no. 1, pp. 113–129, 2010.

[6] K. Riesen and H. Bunke, "Approximate graph edit distance computation by means of bipartite graph matching," *Image and Vision Computing*, vol. 27, no. 7, pp. 950 – 959, 2009, 7th Workshop on Graph-based Representations.

[7] A. Fischer, C. Suen, V. Frinken, K. Riesen, and H. Bunke, "A fast matching algorithm for graph-based handwriting recognition," in *Graph-Based Representations in Pattern Recognition*. Springer, 2013, pp. 194–203.

[8] P. Wang, V. Eglin, C. Garcia, C. Largeron, J. Lladós, and A. Fornés, "A coarse-to-fine word spotting approach for historical handwritten documents based on graph embedding and graph edit distance," in *Int. Conf. on Pattern Recognition*, 2014, pp. 3074–3079.

[9] K. Riesen, D. Brodi, Z. Milivojevi, and C. Maluckov, "Graph based keyword spotting in medieval slavic documents: A project outline," in *Digital Heritage. Progress in Cultural Heritage: Documentation, Preservation, and Protection*. Springer, 2014, vol. 8740, pp. 724–731.

[10] H. Daher, D. Gaceb, V. Eglin, S. Bres, and N. Vincent, "Ancient handwritings decomposition into graphemes and codebook generation based on graph coloring," in *Int. Conf. on Frontiers in Handwriting Recognition*, Nov 2010, pp. 119–124.

[11] L. Schomaker and M. Bulacu, "Automatic writer identification using connected-component contours and edge-based features of uppercase western script," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 787–798, June 2004.

[12] S. Escalera, A. Fornés, O. Pujol, P. Radeva, G. Sánchez, and J. Lladós, "Blurred shape model for binary and grey-level symbol recognition," *Pattern Recognition Letters*, vol. 30, no. 15, pp. 1424–1433, 2009.

[13] P. Riba, J. Lladós, A. Fornés, and A. Dutta, "Large-scale graph indexing using binary embeddings of node contexts," in *Graph-Based Representations in Pattern Recognition*, 2015.

[14] D. Fernández, J. Lladós, and A. Fornés, "A graph-based approach for segmenting touching lines in historical handwritten documents," *Int. Journal on Document Analysis and Recognition*, vol. 17, no. 3, pp. 293–312, 2014.

[15] P. Wang, V. Eglin, C. Garcia, C. Largeron, J. Lladós, and A. Fornés, "A novel learning-free word spotting approach based on graph representation," *11th Int. Workshop on Document Analysis Systems*, pp. 207–211, 2014.

[16] D. Fernández, J. Almazán, N. Cirera, A. Fornés, and J. Lladós, "Bh2m: The barcelona historical, handwritten marriages database," in *Int. Conf. on Pattern Recognition*, 2014, pp. 256–261.

[17] D. Fernandez, P. Riba, A. Fornes, and J. Llados, "On the infuence of key point encoding for handwritten word spotting," in *14th Int. Conf. on Frontiers in Handwriting Recognition*, 2014.

[18] J. Almazán, A. Fornés, and E. Valveny, "Deformable hog-based shape descriptor," in *12th Int. Conf. on Document Analysis and Recognition*. IEEE, 2013, pp. 1022–1026.