A Fast Matching Algorithm for Graph-Based Handwriting Recognition

Andreas Fischer[1], Ching Y. Suen[1],
Volkmar Frinken[2], Kaspar Riesen[3], and Horst Bunke[4]

[1] Centre for Pattern Recognition and Machine Intelligence, Concordia University
1455 de Maisonneuve Blvd West, Montreal, Quebec H3G 1M8, Canada
an_fisch@encs.concordia.ca, suen@encs.concordia.ca
[2] Computer Vision Center, Dept. of Computer Science, Universitat Autònoma
de Barcelona, Edifici O, 08193 Bellaterra, Spain
vfrinken@cvc.uab.cat
[3] Institute for Informations Systems, University of Applied Sciences and Arts
Northwestern Switzerland, Riggenbachstrasse 16, 4600 Olten, Switzerland
kaspar.riesen@fhnw.ch
[4] Institute of Computer Science and Applied Mathematics, University of Bern
Neubrückstrasse 10, 3012 Bern, Switzerland
bunke@iam.unibe.ch

Abstract. The recognition of unconstrained handwriting images is usually based on vectorial representation and statistical classification. Despite their high representational power, graphs are rarely used in this field due to a lack of efficient graph-based recognition methods. Recently, graph similarity features have been proposed to bridge the gap between structural representation and statistical classification by means of vector space embedding. This approach has shown a high performance in terms of accuracy but had shortcomings in terms of computational speed. The time complexity of the Hungarian algorithm that is used to approximate the edit distance between two handwriting graphs is demanding for a real-world scenario. In this paper, we propose a faster graph matching algorithm which is derived from the Hausdorff distance. On the historical Parzival database it is demonstrated that the proposed method achieves a speedup factor of 12.9 without significant loss in recognition accuracy.

1 Introduction

In many pattern recognition applications, graphs are the first choice to represent objects. Their ability to model different parts of an object as well as their binary relations can be used to derive powerful representations of molecular compounds [1], computer networks [2], and symbols in digital images [3], to name just a few. In the domain of handwriting recognition, graphs have found widespread application for single character recognition, especially in the case of Chinese characters that are composed of many complex strokes [4].

However, when it comes to the recognition of unconstrained handwriting images that contain complete sentences in natural language, graph-based repre-

sentation is rarely used due to problems arising from the large variety of character shapes, the large number of words in natural language, and the inability to segment connected handwriting into characters before recognition [5]. Available systems are usually based on vectorial pattern representation $\mathbf{x} \in \mathbb{R}^n$ and statistical classifiers, e.g., hidden Markov models [6] and recurrent neural networks [7]. They cannot be applied directly on graphs representing handwriting.

Recently, a general approach to bridging the gap between graph-based handwriting representation and statistical classification has been proposed in [8]. Based on dissimilarity space embedding [9, 10], handwriting graphs are transformed into feature vectors by calculating their similarity to a set of character prototypes. Graph similarity is obtained by means of graph edit distance [11], an error-tolerant matching method that can be applied to any kind of graph. The similarities constitute the real-valued components of the feature vectors which can then be used in combination with any statistical classifier.

When compared with traditional statistical feature sets, the graph similarity features have shown a promising performance in terms of recognition accuracy [8]. However, although the Hungarian algorithm [12] was used to approximate the graph edit distance in cubic time with respect to graph size [13], the computational complexity remained high, resulting in slow execution speed.

In this paper, we derive a faster matching algorithm for graph-based handwriting recognition from the Hausdorff distance. The proposed method runs in quadratic time with respect to graph size and hence significantly reduces the complexity of the recognition process.

Experiments are conducted on the historical Parzival database [14] which includes images from a 13th century manuscript written in Old German. For a single word recognition task with hidden Markov models, it is demonstrated that the proposed matching algorithm achieves a speedup factor of 12.9 without significant loss in recognition accuracy.

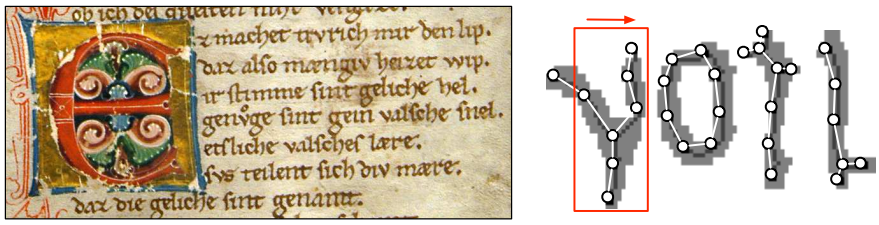
The remainder of this paper is organized as follows. First, the graph similarity features are reviewed in Section 2. Next, the proposed matching algorithm is presented in Section 3. Then, experimental results are presented and discussed in Section 4. Finally, conclusions are drawn in Section 5.

2 Graph Similarity Features

In this section, we briefly review the graph similarity features that are described in detail in [8]. It is a general framework that allows the use of graph-based handwriting representation in combination with statistical classification by means of vector space embedding.

2.1 Handwriting Graphs

The handwriting graphs used in this paper are derived from handwriting skeleton images. An original image is shown in Figure 1a and a skeleton graph in Figure 1b. Image preprocessing includes binarization, correction of the baseline



(a) original image (b) skeleton graph, sliding window

Fig. 1: Handwriting graphs.

inclination, separation of the writing region into an upper, middle, and lower part, and thinning of the strokes to a width of one pixel.

Afterwards, a handwriting graph is constructed by adding endpoints, intersections, and the upper left pixel of circular structures to the set of nodes, labeled with their image position $(x, y) \in \mathbb{R}^2$. Then, further connection points at distance D along the skeleton are added as nodes. The connection point distance D is a parameter to be chosen by the user that determines the node density on the skeleton. Whenever two nodes are connected over the skeleton, they are linked with an undirected, unlabeled edge.

2.2 Vector Space Embedding and Recognition System

An intriguing challenge for connected handwriting recognition is the inability to segment the image into characters before recognition [5]. Instead, a common approach is to perform an oversegmentation with a sliding window moving from left to right over the image and extracting a sequence of feature vectors $x_1, \ldots, x_N$ with $x_i \in \mathbb{R}^n$. The segments are grouped into characters during recognition.

Handwriting graphs are transformed into feature vectors by means of dissimilarity space embedding [10]. First, prototype character graphs are selected, either manually or automatically [15]. Then, a sliding window is moved over the handwriting graph from left to right as illustrated in Figure 1b. At each position, the graph dissimilarity $d(g_1, g_2) \in \mathbb{R}$ between the subgraph g_1 in the window and the prototype graph g_2 is calculated for all prototypes. This results in a sequence of feature vectors $x_1, \ldots, x_N$ with $x_i = (d(g_{1,i}, g_{2,1}), \ldots, d(g_{1,i}, g_{2,n}))$ and a dimensionality $x_i \in \mathbb{R}^n$ equal to the number n of prototypes. The dissimilarity measure used is the graph edit distance which is discussed in detail in Section 3.1. An important property of the edit distance is that it can be applied to any kind of graph.

After embedding, the resulting feature vector sequence can be used for recognition with any statistical classifier. In this paper, we employ hidden Markov models (HMM) [6] for word recognition. For any further details on the recognition system as well as the graph similarity features in general, we refer to [8].

3 Fast Matching Algorithm

A shortcoming of the graph similarity features is their high computational time complexity for matching two handwriting graphs. For a median graph size of 30 nodes, the graph matching process takes about half a minute per word on a 2.66GHz personal computer (see Section 4). Considering a real-world scenario, for instance the daily processing of handwritten letters sent to a company or the processing of large collections of historical manuscripts for digital libraries, this computational speed is demanding in terms of hardware resources.

In this section, we derive a faster graph matching method from the Hausdorff distance. It preserves most properties of the formerly used approximate graph edit distance [13] which is based on a node assignment according to some edit cost. By allowing multiple node assignments for the proposed method, the time complexity is reduced from cubic to quadratic with respect to graph size.

In the following, the approximate graph edit distance is reviewed in Section 3.1, the Hausdorff distance is discussed in Section 3.2, and the proposed modified Hausdorff distance is introduced in Section 3.3.

3.1 Approximate Graph Edit Distance

To calculate the dissimilarity $d(g_1, g_2)$ between two graphs g_1 and g_2, representing the subgraph inside the sliding window and a character prototype (see Section 2.2, Figure 1b), the graph edit distance is used to derive graph similarity features [11]. This distance is given by the minimum cost of edit operations needed to transform g_1 into g_2. Possible edit operations include the substitution, deletion, and insertion of nodes and edges.

For the handwriting graphs under consideration (see Section 2.1), the Euclidean cost function is used with

- $c(n_1, n_2) = ||(x_1, y_1) - (x_2, y_2)||$ for node label substitution
- $c(n_1, \epsilon) = c(\epsilon, n_2) = C_n \geq 0$ for node deletion and insertion
- $c(e_1, \epsilon) = c(\epsilon, e_2) = C_e \geq 0$ for edge deletion and insertion

where (x_i, y_i) is the attribute vector associated with node n_i, representing the location of n_i in the two-dimensional plane. This definition ensures the edit distance to be metric [11]. The non-negative parameters C_n and C_e for deletion and insertion are optimized on a validation set to adapt the generic cost function to the graph data. As there are no edge labels, no edge label substitution cost need to be defined.

Usually, the edit distance is calculated with the A^* algorithm which performs a best-first tree search, possibly using a lower bound heuristic for the estimated future cost [11]. The A^* algorithm always finds the optimal solution but has an exponential time complexity with respect to the graph size. In order to match large handwriting graphs, an approximation is used to obtain a suboptimal edit distance in polynomial time [13]. The approximation reduces the edit distance to a node assignment problem which can then be solved in cubic time by Munkres'

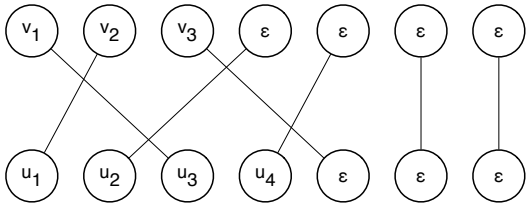


Fig. 2: Assignment problem.

algorithm [12], also known as the Hungarian algorithm. Although the algorithm does not always find an optimal solution for the edit distance, it is reasonably accurate, especially for small distances among similar graphs which is important for the task of classification [13].

The edit distance can be formulated as a node assignment problem as illustrated in Figure 2. In this example, we consider a graph g_1 with three nodes v_1, v_2, v_3 (top row) that is matched with a graph g_2 having four nodes u_1, u_2, u_3, u_4 (bottom row). For each node in g_1, an ϵ-node is inserted in the bottom row, and for each node in g_2 an ϵ-node is inserted in the top row. Assignments between the top and the bottom row correspond with node edit operations, e.g., substitution (v_1, u_3), deletion (v_3, ϵ), and insertion (ϵ, u_2).

Finding a complete assignment with minimum cost[5] corresponds to finding an optimal solution for the edit distance if the edges of the graphs are ignored. By taking the implied cost of adjacent edge operations into account for each node assignment, the true edit distance can be approximated. The method is suboptimal because only local adjacent edge structures are matched instead of the global edge structure. Munkres' algorithm solves the assignment problem in $O(N^3)$ where N is the sum of the number of nodes in g_1 and g_2.

For graph similarity features, a normalization of the approximate graph edit distance with respect to its maximum value has proven beneficial. It is accomplished by $d_{max}(g_1, g_2) = N \cdot C_n + E \cdot C_e$ where N is the sum of the number of nodes in g_1 and g_2 and E is the sum of the number of edges. The maximum corresponds to the deletion of all nodes and edges in g_1 and the insertion of all nodes and edges in g_2. A similarity value is obtained by

$$\hat{s}(g_1, g_2) = 1 - \frac{d(g_1, g_2)}{d_{max}(g_1, g_2)} \tag{1}$$

Also, a normalization over all prototype characters $p \in P$ is performed at each sliding window position yielding the final graph similarity measure

$$s(g_1, g_2) = \frac{\hat{s}(g_1, g_2)^2}{\sum_P \hat{s}(g_1, p)} \tag{2}$$

[5] The assignment cost (ϵ, ϵ) is zero.

3.2 Hausdorff Distance

The Hausdorff distance is a distance measure between two subsets of a metric space. In case of finite subsets A and B the Hausdorff distance $H(A, B)$ is

$$H(A,B) = \max(\max_A \min_B d(a,b), \max_B \min_A d(a,b)) \tag{3}$$

where $a \in A$, $b \in B$, and $d(a, b)$ is the underlying metric [16]. In Equation 3, $\min_B d(a, b)$ is the nearest neighbor distance of a in B, $\min_A d(a, b)$ is the nearest neighbor distance of b in A, and the Hausdorff distance corresponds to the maximum over all nearest neighbor distances.

The Hausdorff distance is widely used in the domain of image matching [16], for example to locate templates within target images. In its original definition, only the maximum over all nearest neighbor distances is taken into account. Hence, Hausdorff distance is sensitive to outliers in the data. A straight-forward modification that integrates all nearest neighbor distances can be achieved with

$$H'(A,B) = \sum_A \min_B d(a,b) + \sum_B \min_A d(a,b) \tag{4}$$

Considering A as the nodes of graph g_1, B as the nodes of graph g_2, and $d(n_1, n_2) = c(n_1, n_2) = ||(x_1, y_1) - (x_2, y_2)||$ as the underlying metric, i.e., the node substitution cost (see Section 3.1), the Hausdorff distance can be directly applied to the handwriting graphs. It ignores the edges of the graphs and can trivially be calculated in $O(NM)$ where N and M denote the number of nodes in g_1 and g_2, respectively.

Therefore the Hausdorff distance or its modification in Equation 4 can be used as a fast alternative for the approximate graph edit distance. In our experiments (see Section 4) a normalization with respect to all prototypes $p \in P$ has proven beneficial yielding the final graph dissimilarity measures

$$h(g_1, g_2) = \frac{H(g_1, g_2)}{\sum_P H(g_1, p)} \tag{5}$$

$$h'(g_1, g_2) = \frac{H'(g_1, g_2)}{\sum_P H'(g_1, p)} \tag{6}$$

3.3 Modified Hausdorff Distance

In this paper, we propose a novel modification of the Hausdorff distance that takes into regard not only substitution, but also deletion and insertion cost. It is defined as

$$H''(A,B) = \sum_A \min_B \bar{c_1}(a,b) + \sum_B \min_A \bar{c_2}(a,b) \tag{7}$$

by replacing the metric d in Equation 4 with the cost functions $\bar{c_1}$ and $\bar{c_2}$. Again, A corresponds with the nodes of graph g_1 and B with the nodes of g_2. The cost

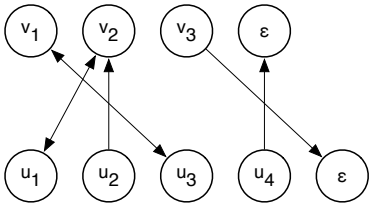


Fig. 3: Multiple assignments.

functions $\bar{c_1}(n_1, n_2)$ and $\bar{c_2}(n_1, n_2)$ for matching node n_1 with node n_2 are

$$\bar{c_1}(n_1, n_2) = \begin{cases} \frac{c(n_1,n_2)}{2}, & \text{if } c(n_1, n_2) < c(n_1, \epsilon) \\ c(n_1, \epsilon), & \text{otherwise} \end{cases} \tag{8}$$

$$\bar{c_2}(n_1, n_2) = \begin{cases} \frac{c(n_1,n_2)}{2}, & \text{if } c(n_1, n_2) < c(\epsilon, n_2) \\ c(\epsilon, n_2), & \text{otherwise} \end{cases} \tag{9}$$

with respect to the cost function c of the edit distance (see Section 3.1). That is, $\min_B \bar{c_1}(a, b)$ returns half of the node substitution cost $\frac{c(n_1,n_2)}{2}$ if the substitution (n_1, n_2) is preferred over deletion (n_1, ϵ) of n_1. Among all possible substitutions the one with the smallest cost is chosen. Otherwise, the deletion cost $c(n_1, \epsilon)$ is returned. Similarly, $\min_A \bar{c_2}(a, b)$ returns the best $\frac{c(n_1,n_2)}{2}$ if the substitution (n_1, n_2) is preferred over insertion (ϵ, n_2) of n_2.

The correspondence between the modified Hausdorff distance H'' and the approximate edit distance is illustrated in Figure 3 in analogy to Figure 2. Based on H'', each node of graph g_1 (top row) is either assigned to a node of graph g_2 (bottom row) if a substitution is preferred in $\bar{c_1}$ or to the ϵ node for deletion. Vice versa, each node of graph g_2 is either assigned to a node of graph g_1 if a substitution is preferred in $\bar{c_2}$ or to the ϵ node for insertion.

H'' equals the cost of all assignments. The cost of double assignments, e.g., (v_1, u_3) in Figure 3, is the full substitution cost $c(n_1, n_2) = \bar{c_1}(n_1, n_2) + \bar{c_2}(n_1, n_2)$. Deletions and insertions contribute their respective cost. The only difference to the assignment problem solved by the approximate edit distance is the possibility of multiple assignments, e.g., with v_2 in Figure 3. In such a case H'' is smaller than the approximate edit distance which is an upper bound of H''.

Finally, implied adjacent edge costs can be taken into account for the assignments in $\bar{c_1}$ and $\bar{c_2}$ in the same way as for the approximate edit distance. For small edit distances between two graphs, H'' is, indeed, expected to provide a good approximation of the edit distance in quadratic time.

For the graph similarity features, the usual normalization is applied (see Section 3.1). With $\hat{h''}(g_1, g_2) = 1 - \frac{H''(g_1,g_2)}{d_{max}(g_1,g_2)}$ the final graph similarity measure is obtained as

$$h''(g_1, g_2) = \frac{\hat{h''}(g_1, g_2)^2}{\sum_P \hat{h''}(g_1, p)} \tag{10}$$

4 Experimental Evaluation

Experiments are conducted on the historical Parzival database[6] [14] which includes images from a 13th century manuscript written in Old German. 11,743 word images are considered that contain 3,177 word classes and 87 characters.

For a single word recognition task with graph similarity features and HMM-based recognition (see Section 2), the similarity function h'' obtained from the modified Hausdorff distance is compared with the Hausdorff distance h, its variant h', and the approximate edit distance s (see Section 3).

4.1 Setup

First, the word images are divided into three distinct sets for training, validation, and testing. Half of the words are used for training and a quarter of the words for validation and testing, respectively. For vector space embedding, 79 character prototypes are used as in [8].

Parameters that are optimized with respect to the validation accuracy include the connection point distance $D \in \{3, 5, 7, 9\}$ for graph-based representation, the deletion and insertion cost $C_n, C_e \in \{0, 0.4D, 0.6D, \ldots, 1.4D\}$ of the graph edit distance, and the number of Gaussian mixtures $G \in \{1, 2, \ldots, 30\}$ of the HMM. Optimal parameter values are adopted from previous studies [8] conducted with the approximate edit distance s. The same values are used for h, h', and h''.

Table 1: Word recognition accuracy on the test set in percentage.

h	h'	h''	s
49.78	83.95	93.66	94.00

4.2 Results

The achieved word recognition accuracy on the test set is listed in Table 1. As stated in [8], the best accuracy of 94.00% achieved with graph similarity features and approximate edit distance s significantly outperforms traditional statistical feature sets which achieve a maximum accuracy of 90.49% [8]. This result demonstrates the general effectiveness of the proposed graph-based approach to handwriting recognition.

With an accuracy of 93.66% the proposed modified Hausdorff distance h'' achieves nearly the same performance as s. There is no statistically significant difference between the results (t-test, $\alpha = 0.05$). That is, the improvement from cubic to quadratic time complexity can be achieved without significant loss in accuracy.

[6] http://www.iam.unibe.ch/fki/databases/iam-historical-document-database

Table 2: Runtime statistics. The median graph size in terms of number of nodes, the median number of graph matchings per word, the mean runtimes on a 2.66GHz processor for s and h'' per word in seconds, and the speedup factor.

Graph Size	Matchings	Runtime s	Runtime h''	Speedup
30	6162	33.24	2.57	12.9

When comparing the results of h and h', a very remarkable difference in performance is observed. This is possibly due to the fact that the Hausdorff variant h' is less sensitive to outliers. Still, both distance measures perform significantly worse than h'' and s.

The runtime statistics of h'' and s are listed in Table 2. Both methods are implemented in Java. For an optimal value $D_{opt} = 3$, the median number of graph nodes is 30 and the proposed algorithm achieves a speedup factor of 12.9.

An anomaly is observed for the cost function parameters $C_{n,opt} = 3$ and $C_{e,opt} = 0$. The edge cost parameter $C_{e,opt} = 0$ indicates that the use of edges in the chosen graph representation actually leads to worse results. We explain this anomaly by the fact that the handwriting images of historical manuscripts contain many broken characters due to binarization problems. Imposing an edge cost leads to stronger deviations from the clean prototype characters in this case.

5 Conclusions

Graph similarity features provide a general framework to combine graph-based representation and statistical classification for the recognition of handwritten text images. The framework proposed in this paper is based on vector space embedding of handwriting graphs with respect to a set of character prototypes. It showed a high recognition accuracy when compared with traditional statistical feature sets, but had shortcomings in computational speed when matching two handwriting graphs with an approximate graph edit distance.

In this paper, we propose a fast matching algorithm derived from the Hausdorff distance that reduces the complexity of the graph matching process from cubic to quadratic time with respect to graph size. The method retains most properties of the approximate edit distance but allows multiple node assignments. On the historical Parzival database it was demonstrated for an HMM-based word recognition task that a speedup factor of 12.9 could be achieved without significant loss in accuracy.

In the domain of handwriting recognition, future work includes the investigation of different graph-based representations of handwriting within the proposed framework. In the field of image matching, the proposed distance measure could be used as a variant of the Hausdorff distance in various applications, such as template location. Finally, for graph-based recognition in general, the algorithm offers a promising possibility to approximate the graph edit distance in quadratic time with respect to graph size. This issue needs to be verified on diverse graph data sets.

Acknowledgment

This work has been supported by the Swiss National Science Foundation fellowship project PBBEP2_141453, the Spanish project TIN2009-14633-C03-03, and the Spanish MICINN under the MIPRCV "Consolider Ingenio 2010" CSD2007-00018 project.

References

1. Mahé, P., Ueda, N., Akutsu, T., Perret, J., Vert, J.: Graph kernels for molecular structure-activity relationship analysis with support vector machines. Journal of Chemical Information and Modeling **45**(4) (2005) 939–951
2. Bunke, H., Dickinson, P.J., Kraetzl, M., Wallis, W.D.: A Graph-Theoretic Approach to Enterprise Network Dynamics. Volume 24 of Progress in Computer Science and Applied Logic. Birkhäuser (2006)
3. Llados, J., Marti, E., Villanueva, J.: Symbol recognition by error-tolerant subgraph matching between region adjacency graphs. IEEE Trans. PAMI **23**(10) (2001) 1137–1143
4. Lu, S., Ren, Y., Suen, C.Y.: Hierarchical attributed graph representation and recognition of handwritten chinese characters. Pattern Recognition **24**(7) (1991) 617–632
5. Bunke, H., Varga, T.: Off-line Roman cursive handwriting recognition. In Chaudhuri, B., ed.: Digital Document Processing. Springer (2007) 165–173
6. Ploetz, T., Fink, G.A.: Markov models for offline handwriting recognition: A survey. Int. Journal on Document Analysis and Recognition **12**(4) (2009) 269–298
7. Graves, A., Liwicki, M., Fernandez, S., Bertolami, R., Bunke, H., Schmidhuber, J.: A novel connectionist system for improved unconstrained handwriting recognition. IEEE Trans. PAMI **31**(5) (2009) 855–868
8. Fischer, A., Riesen, K., Bunke, H.: Graph similarity features for HMM-based handwriting recognition in historical documents. In: Proc. 12th Int. Conf. on Frontiers in Handwriting Recognition. (2010) 253–258
9. Pekalska, E., Duin, R.: The Dissimilarity Representations for Pattern Recognition: Foundations and Applications. World Scientific (2005)
10. Riesen, K., Bunke, H.: Graph Classification and Clustering Based on Vector Space Embedding. World Scientific (2010)
11. Bunke, H., Allermann, G.: Inexact graph matching for structural pattern recognition. Pattern Recognition Letters **1**(4) (1983) 245–253
12. Munkres, J.: Algorithms for the assignment and transportation problems. Journal of the Society for Industrial and Applied Mathematics **5**(1) (1957) 32–38
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. Fischer, A., Wüthrich, M., Liwicki, M., Frinken, V., Bunke, H., Viehhauser, G., Stolz, M.: Automatic transcription of handwritten medieval documents. In: Proc. 15th Int. Conf. on Virtual Systems and Multimedia. (2009) 137–142
15. Fischer, A., Bunke, H.: Character prototype selection for handwriting recognition in historical documents with graph similarity features. In: Proc. 19th European Signal Processing Conference. (2011) 1435–1439
16. Huttenlocher, D.P., Klanderman, G.A., Kl, G.A., Rucklidge, W.J.: Comparing images using the Hausdorff distance. IEEE Trans. PAMI **15** (1993) 850–863