Handwritten Word Recognition based on Structural Characteristics and Lexical Support

E. Kavallieratou, K. Sgarbas, N. Fakotakis and G. Kokkinakis
Wire Communications Lab.
University of Patras, 26500 Patras
ergina@wcl.ee.upatras.gr

Abstract

In this paper a handwritten recognition algorithm based on structural characteristics, histograms and profiles, is presented. The well-known horizontal and vertical histograms are used, in combination with the newly introduced radial histogram, out-in radial and in-out radial profiles for representing 32x32 matrices of characters, as 280-dimension vectors.

The recognition process has been supported by a lexical component based on dynamic acyclic FSAs (Finite-State-Automata).

1.Introduction

In this paper a handwritten character recognition approach is presented. The proposed technique has been developed as the last module of an integrated document analysis system, shown in fig. 1 [1-6].

In general, the character recognition procedure consists of two steps: (i) feature extraction where each character is represented as a feature vector and (ii) classification of the vectors into a number of classes. Govindan [7] distinguishes two categories of features: the structural and the statistical features, while Bunke [8] estimates that the structural approach is closer to the human way of recognition. In this paper, we propose a structural approach for feature extraction. Thus, a 280-dimension vector is extracted for each character, consisting of histograms and profiles. One new histogram and two new profiles are introduced. The k-means algorithm is, then, used for classification.

The proposed technique, described in section 2, is fast and simple, while the experimental results, illustrated in section 4, are quite promising. An accompanying lexical component is described in section 3. Finally, in section 5 our suggestions and plans for further improving are presented.

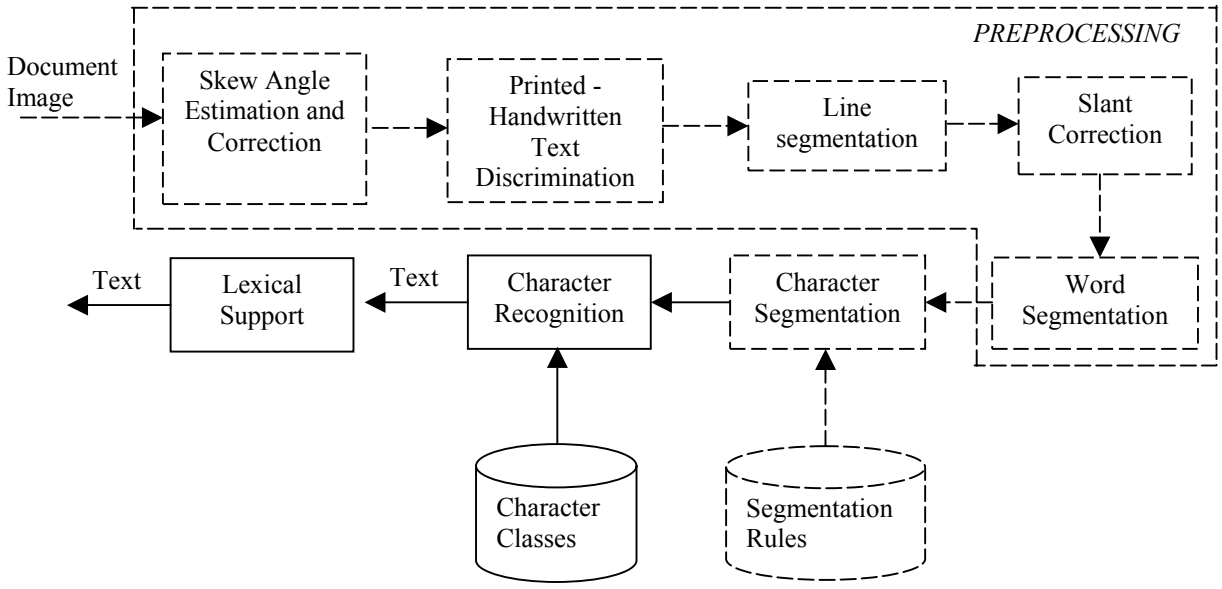
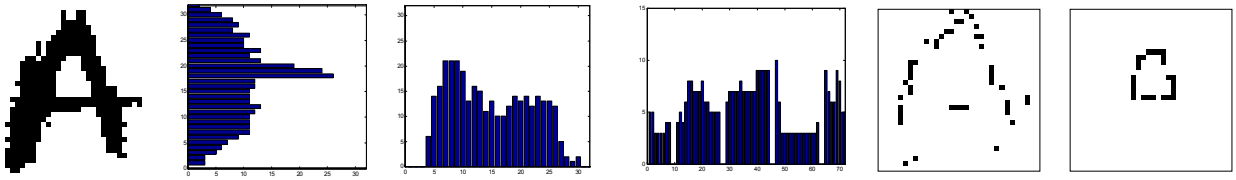


Figure 1. An integrated OCR system.

Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003)
0-7695-1960-1/03 $17.00 © 2003 IEEE



(a) (b) (c) (d) (e) (f)

Figure 2. a) Character matrix, b) Horizontal histogram, c) Vertical histogram, d) Radial histogram, e) Radial out-in profile and f) Radial in-out profile.

2. Algorithm description

As can been seen in fig. 1, after the preprocessing stage, isolated character images are produced which are used as input to the character recognition module. Each character is, then, represented as a 280-dimension vector, consisting of histograms and profiles. In more detail, each character is normalized in a 32x32 matrix. The horizontal histogram, the vertical histogram, the *radial* histogram, the *out-in* profile and the *in-out radial* profile are, then, calculated. The first two histograms are well- known and have been used extensively in optical character recognition research while the remaining features are used for first time. The formal definition of these features is follows.

Consider that the value of the element in the *m*-th row and *n*-th column of the character matrix is given by a function *f*:

$$f(m,n) = a_{mn}$$

where a_{mn} takes binary values (i.e., 0 for white pixels and 1 for black pixels). The horizontal histogram H_h of the character matrix is the sum of black pixels in each row (i.e., 32 features):

$$H_h(m) = \sum_n f(m,n)$$

Similarly, the vertical histogram H_v of the character matrix is the sum of black pixels in each column (i.e., 32 features):

$$H_v(n) = \sum_m f(m,n)$$

We define the value of the radial histogram H_r at an angle φ as the sum of black pixels on a rad that starts from the center of the character matrix (i.e., the element in the 16^{th} row and the 16^{th} column) and ends up at the border of the matrix, forming an angle φ with the horizontal axis. The radial histogram values are calculated with a step of 5 degrees (i.e., 72 features):

$$H_r(\phi) = \sum_{i=1}^{16} f(\|16 - i\sin\phi\|, \|16 + i\cos\phi\|),$$

$$\phi = 5 * k, k \in [0,72)$$

Additionally, we define the value of the *out-in radial profile* P_{oi} at an angle φ as the position of the first black pixel found on the rad that starts from the periphery and goes to the center of the character matrix forming an angle φ with the horizontal axis. The out-in radial profile values are calculated with a step of 5 degrees (i.e., 72 features):

$$P_{oi}(\phi) = \left\{ \begin{array}{l} I : \sum_{i=16}^{I-1} f(\|16 - i\sin\phi\|, \|16 + i\cos\phi\|) \equiv 0 \\ \& \; f(\|16 - I\sin\phi\|, \|16 + I\cos\phi\|) \equiv 1 \end{array} \right\},$$

$$\phi = 5 * k, k \in [0,72)$$

Similarly, we define the value of the *in-out radial profile* P_{io} at an angle φ as the position of the first black pixel on the rad that starts from the center and goes to the periphery of the character forming an angle φ with the horizontal axis. As above, the in-out radial profile values are calculated with a step of 5 degrees (i.e., 72 features):

$$P_{io}(\phi) = \left\{ \begin{array}{l} J : \sum_{i=0}^{J-1} f(\|16 - i\sin\phi\|, \|16 + i\cos\phi\|) \equiv 0 \\ \& \; f(\|16 - J\sin\phi\|, \|16 + J\cos\phi\|) \equiv 1 \end{array} \right\},$$

$$\phi = 5 * k, k \in [0,72)$$

An illustrated example is given in fig. 2, where the previously defined histograms and profiles (i.e., fig. 2b, 2c, 2d, and 2e) for a handwritten character (i.e., fig 2a) are shown.

Thus, a 280-dimension vector is extracted for each character image. A classification model is, then, produced by applying the k-means algorithm to the training data.

The preprocessing of the document images (see fig. 1) is likely to cause the undesirable segmentation of a handwritten character image into two character images. Taking this into account, during the classification of unseen cases, a feature vector is extracted for each

character image as well as for each pair of successive character images. These vectors are classified into the character class from which the Euclidean distance is minimized.

3. Lexical support

The recognition process has been supported by a lexical component based on dynamic acyclic FSAs (Finite-State-Automata).

Acyclic FSAs provide a very efficient data structure for lexicon representation and fast string matching, with a great variety of applications [9-11]. They constitute very compact representations of lexicons, since common prefixes and suffixes are represented by the same transitions (see Fig.3).

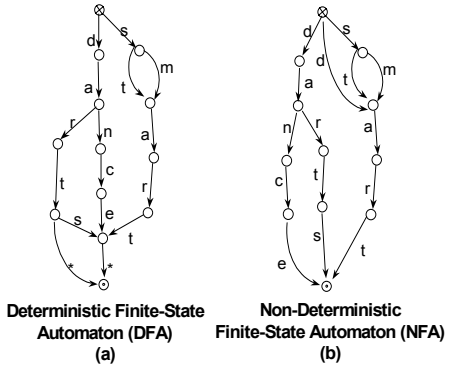


Figure 3. The same lexicon in DFA and NFA.

The FSAs consist of *states* and *transitions* between states. Each transition has a *label*. The words are stored as directed paths on the graph. They can be retrieved by traversing the graph from an initial state (*source*) to a terminal state (*sink*), collecting the labels of the transitions encountered. In this way, traversing the graphs of Fig.1 from the source (⊗) to the sink (⊙) we retrieve the strings *dance*, *darts*, *dart*, *start* and *smart*. Storing lexicons in FSAs is efficient because common transitions are used to represent identical suffixes and prefixes, and because the stρing or substring search can be performed in various flexible ways (including content-addressable and approximate matching).

There are two types of FSAs. The graph of Fig.1a is called *deterministic* (DFA) because no transitions exist that have the same labels and leave the same state. This property results to a very efficient search function (linear to the size of the string). Graphs that do not have this property, like the one of Fig.1b, are called *non-deterministic* automata (NFA). NFAs require less space than DFAs to store the same lexicon, but they are a little slower to search.

Recent research [12-21] has produced a class of algorithms that are able to efficiently update acyclic FSAs, even maintaining their minimality. These algorithms can dynamically add and remove strings from acyclic FSAs, without performing a minimization procedure in the whole graph. They produce minimal DFAs and near-minimal NFAs (called compact [21]) since the minimization of NFAs is still an open problem. The constructed lexicon used in the current system consists of 230,000 Greek words (average word length: 9.5 characters; alphabet size: 36). Figure 4 shows the space and time performance for the process of building the lexicon in both forms (DFA and NFA) using a pair of dynamic algorithms of the aforementioned class (described in detail in [19-21]). The number of states, transitions and the construction time were measured. The thick lines refer to the NFA; the thin lines refer to the corresponding minimal DFA. The test was performed on a 200 MHz PC.

Figures 4a, 4b and 4c display respectively the number of states, the number of transitions and the construction time of the automaton, in respect to the size of the lexicon (number of words). The same results are also shown in logarithmic scales in Figs.4d, 4e and 4f, respectively. The results agree with the theoretic calculations [19,20,22], indicating linear growth of transitions, less than linear growth of states and an O(n) time performance for the insertion process.

Given a word pattern, the lexicon is searched using an improved A* search algorithm, which is able to retrieve a list of N-best words that match the input pattern. Character insertions, deletions and substitutions are modelled as distance penalties according to a confusion matrix that represents the possible alterations. Thus all possible errors are expressed as scores in the process of the A* search and each entry in the output list of N-best words is accredited an overall matching score as well.

4. Experimental results

In order to evaluate our technique, we performed experiments using the NIST [23] database for English characters as well as the GRUHD [24] database for Greek characters. In both cases, the recognition system was trained using 2000 samples and 128 classes for each symbol and was tested on 500 samples for each symbol. The training and the test set were completely disjoint. Thus, the writers used in testing were completely different from the ones used in training.

In more detail, the recognition system was first trained based on the NIST database, for each one of the following categories separately: digits, uppercase characters and lowercase characters. Then, it was tested on unseen cases of the corresponding categories, taken from the same database. The accuracy rate in each case is shown in table

1. Since the output of the proposed character recogniser could be further improved by using lexicons, the recognition accuracy when the second and the third choices are taken into account are also given.

Next, the algorithm was trained and tested using the Modern Greek database of unconstrained writing (GRUHD). Since, the forms of GRUHD are very similar to those of the NIST database, it was possible to train our system separately for digits, uppercase and lowercase characters as well. The results on unseen cases of the corresponding categories are shown on table 2. In contrast to the NIST database, the GRUHD database contains unconstrained handwriting, so the accuracy results are somewhat lower in the latter case. In figures 5 and 6 the relation of the recognition accuracy with the training set size and the number of classes per symbol, respectively, is illustrated. In both cases the given results concern the characters of the NIST database.

The system trained as before, using NIST database, was applied to about 280 forms of the IAM-DB, database of unconstrained writing. The accuracy here, depending on the form varied from 71%-75,8%, an accuracy rate much lower than the one succeeded on NIST database. Two are the reasons that contributed to that decrease. The NIST is a database of hand-printed characters that are well segmented and much easier in recognition. On the other hand, in IAM-DB both lowercase and uppercase characters had to be recognized at the same time. By providing the system with lexical information, the accuracy results are 86-87%.

Table 1: The accuracy rates for the NIST.

	1st Choice	2nd Choice	3rd Choice
Digits	98.8%	99.91	100%
Uppercase characters	93.85%	96.54	98.86%
Lowercase characters	91.4%	94.50%	98.85%
Mixed characters	82.79	89.27%	96.85%

Table 2: Experimental results for the GRUHD.

	1st Choice	2nd Choice	3rd Choice
Digits	94%	97.42%	99,54%
Uppercase characters	86.03%	96.40%	98.96%
Lowercase characters	81%	90.36%	96.60%
Mixed characters	72.8%	80.04%	88.83%

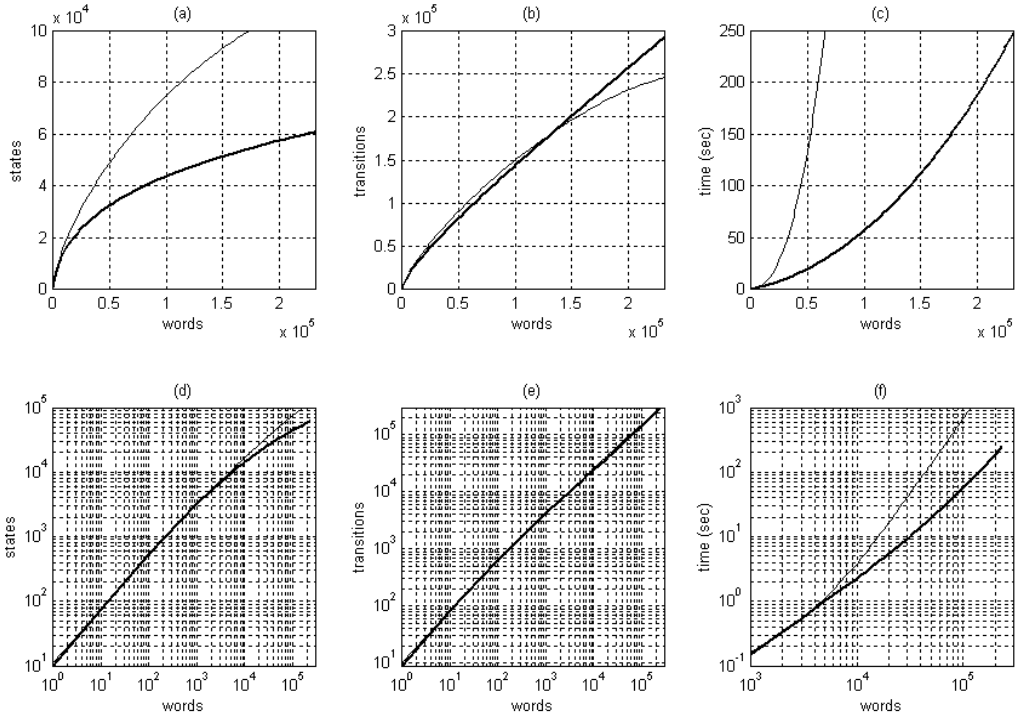


Figure 4. Space-time performance for dynamic DFA (thin lines) and NFA (thick lines).

4. Conclusion

In this paper a technique for handwritten character recognition is presented. The proposed technique is focuses on the extraction of the features that best describe a handwritten character introducing one new histogram (i.e., radial) and two new profiles (i.e., in-out and out-in). These features together with the well-known horizontal and vertical histograms form a reliable representation of a handwritten character.

The described approach has been tested on two different databases with recognition accuracy varying from 72.8% to 98.8% depending on the difficulty of the database and the character category. On word recognition level, the system has been supported by a lexical component based on dynamic acyclic FSAs (Finite-State-Automata).

5. References

[1] E.Kavallieratou, N.Fakotakis and G.Kokkinakis, ``Skew Estimation using Cohen's class distributions", Pattern Recognition Letters 20, 1999, pp. 1305-1311.

[2] E.Kavallieratou, N.Fakotakis, and G.Kokkinakis, ``Slant Estimation Algorithm for OCR Systems", Pattern Recognition, v.34, n.12, 2001, pp.2515-2522.

[3] M.Maragoudakis, E.Kavallieratou, N.Fakotakis and G.Kokkinakis, ``How Conditional Independence Assumption Affects Handwritten Character Segmentaion", In Proc. ICDAR, 2001.

[4] E.Kavallieratou, D.C.Balcan, M.F.Popa, and N.Fakotakis, ``Handwritten Text Localization in Skewed Documents", In Proc. ICIP, 2001, pp.1102-1105.

[5] E. Kavallieratou, E. Stamatatos, N. Fakotakis, and G. Kokkinakis, ``Handwritten Character Segmentation Using Transformation-Based Learning", In Proc. ICPR, 2000, pp.634-637.

[6] E.Kavallieratou, N.Fakotakis, G.Kokkinakis, ``New algorithms for skewing correction and slant removal on word-level", In Proc. ICECS, 1999, pp.1159-1162.

[7] V.K.Govindan, A.P.Shivaprasad, ``Character recognition -- A review", Pattern Recognition, v. 23, No 7, 1990, pp. 671-683.

[8] H.Bunke, A.Sanfeliu, Syntactic and structural Pattern Recognition, Theory and Applications, World Scientific, Singapore.

[9] M. Crochemore and R. Verin. 1997. Direct Construction of Compact Directed Acyclic Word Graphs. *8th Annual Symposium, CPM 97,* Aarhus, Denmark, 116-129.

[10] R. Lacouture and R. De Mori. 1991. Lexical Tree Compression. *EuroSpeech '91, 2nd European Conference on Speech Communications and Techniques,* Genova, Italy, 581-584.

[11] K. Sgarbas, N. Fakotakis and G. Kokkinakis. 2000. A Straightforward Approach to Morphological Analysis and Synthesis. *Proc. COMLEX 2000, Workshop on Computational Lexicography and Multimedia Dictionaries,* Kato Achaia, Greece, 31-34.

[12] J. Aoe, K. Morimoto and M. Hase. 1993. An Algorithm for Compressing Common Suffixes Used in Trie Structures. *Systems and Computers in Japan,* 24(12):31-42.

[13] M. Ciura and S. Deorowicz. 1999. Experimental Study of Finite Automata Storing Static Lexicons. Report BW-453/RAu-2/99 (Also at http://www-zo.iinf.polsl.gliwice.pl/~sdeor/pub.htm).

[14] J. Daciuk, S. Mihov, B. Watson and R. Watson. 2000. Incremental Construction of Minimal Acyclic Finite State Automata. *Computational Linguistics,* 26(1):3-16.

[15] J. Daciuk, R.E. Watson and B.W. Watson. 1998. Incremental Construction of Acyclic Finite-State Automata and Transducers. *Proceedings of Finite State Methods in Natural Language Processing,* Bilkent University, Ankara, Turkey.

[16] S. Mihov. 1998. Direct Construction of Minimal Acyclic Finite States Automata. *Annuaire de l' Universite de Sofia 'St. Kl. Ohridski',* Faculte de Mathematique et Informatique, Sofia, Bulgaria, 92(2).

[17] K. Park, J. Aoe, K. Morimoto and M. Shishibori. 1994. An Algorithm for Dynamic Processing of DAWG's. *International Journal of Computer Mathematics,* Gordon and Breach Publishers SA, OPA Amsterdam BV, 54:155-173.

[18] D. Revuz. 2000. Dynamic Acyclic Minimal Automaton. CIAA 2000, 5th International Conference on Implementation and Application of Automata, London, Canada pp.226-232.

[19] K. Sgarbas, N. Fakotakis and G. Kokkinakis. 1995. Two Algorithms for Incremental Construction of Directed Acyclic Word Graphs. *International Journal on Artificial Intelligence Tools*, World Scientific, 4(3):369-381.

[20] K. Sgarbas, N. Fakotakis and G. Kokkinakis. 2000. Optimal Insertion in Deterministic DAWGs. Technical Report WCL/SLT#000524, Wire Communications Lab., Dept. of Electrical and Computer Engineering, University of Patras, Greece (Also at http://slt.wcl.ee.upatras.gr/sgarbas/PublAbsEN.htm).

[21] K. Sgarbas, N. Fakotakis and G. Kokkinakis. 2001. Incremental Construction of Compact Acyclic NFAs. *Proc. ACL-2001, 39th Annual Meeting of the ACL,* Toulouse, France, 474-481.

[22] A. Blumer, D. Haussler, and A. Ehrenfeucht. 1989. Average Sizes of Suffix Trees and DAWGs. *Discrete Applied Mathematics,* 24:37-45.

[23] R. Wilkinson, J. Geist, S. Janet, P. Grother, C. Burges, R. Creecy, B. Hammond, J. Hull, N. Larsen, T. Vogl, and C. Wilson, The first census optical character recognition systems conf. #NISTIR 4912. The U.S Bureau of Census and the National Institute of Standards and Technology. Gaithersburg, MD, 1992.

[24] E.Kavallieratou, N.Liolios, E.Koutsogeorgos, N.Fakotakis, G.Kokkinakis, ``The GRUHD database of Modern Greek Unconstrained Handwriting", In Proc. ICDAR ,2001.

[25] G.Frosini, B.Lazzerini, A.Maggiore, F.Marcelloni, ``Fuzzy classification based system for handwritten character recognition", In Proc. 2 nd Intern. Conf. On Knowledge-based Intelligent Electronic Systems (KES'98), pp. 61-65, 1998.