OFF-LINE CURSIVE HANDWRITTEN WORD RECOGNITION BASED ON TREE EXTRACTION AND AN OPTIMIZED CLASSIFICATION DISTANCE

J. R. Rico

Departamento de Lenguajes y Sistemas Informáticos.

Universidad de Alicante.

E-03071 Alicante. SPAIN.

e-mail: juanra@dlsi.ua.es

Abstract

This paper describes a geometric approach to the difficult off-line handwritten word recognition problem. The method classifies feature trees from isolated handwritten words, measuring the distance between two trees. The nearest-neighbour method has been used to classify the prototypes and the leaving-one-out criterion has been applied in order to test the classifier.

Keywords: Off-line cursive handwriting recognition, edit-distance optimisation, distance between trees.

1.Introduction

The issue of pattern recognition is central to many applications of computer science and technology. The off-line recognition of cursive handwritten words is an interesting problem because it is easy to scan a handwritten document, to train the system with a particular handwriting style and afterwards to let the system classify the rest of the text. In this paper the objective is to classify isolated words. It is assumed that algorithms for isolating individual words such as the “Cursive word reference line detection”[0] have been used.

The main idea is to extract word features from handwritten words in the corpus and to build a database directly from images. The features of new words will be extracted and will be compared with those in the database. An additional dictionary or the splitting of words in letters or other subword units is not needed. The algorithm extracts the features directly from a bitmap. The recognition process consists of the following stages (Figure 1) described below:

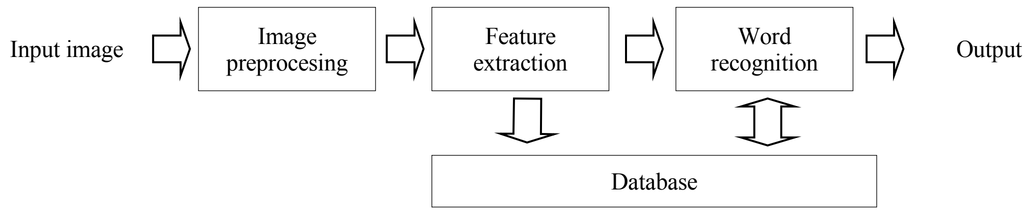


Figure 1. Stages in the word recognition process

This paper is structured as follows. Section 2 explains the way in which words are processed and transformed into trees. Section 3 explains the classification method. Section 4 shows the experimental results. Finally, section 5 gives the conclusion.

2.Data preprocesing and feature tree extraction

2.1Image preprocesing

Input images are taken from cursive handwritten words. Images are filtered to obtain the best representation for each word. The way the filters are applied to the image is important.

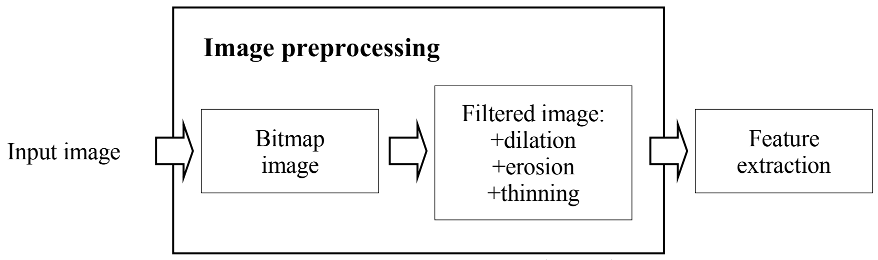
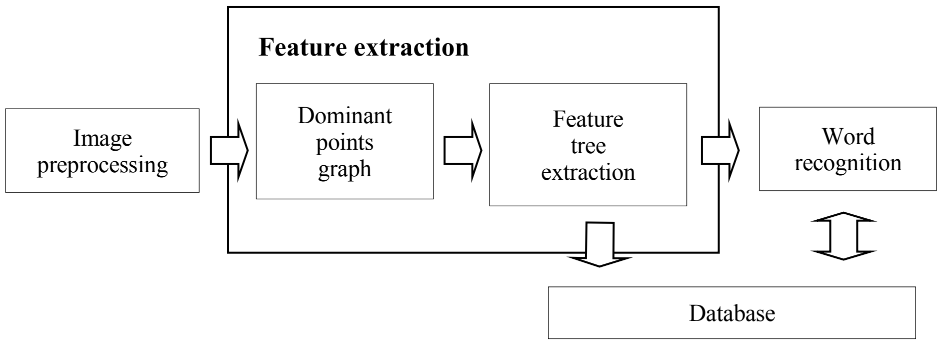


Figure 2. Data preprocessing sub-stages.

The document is scanned and a greyscale image (Figure 4) is obtained for system input (Figure 1). Then, the data preprocessing steps are applied (Figure 2). The greyscale image (input) is converted into a bitmap image (Figure 5). This process has an aliasing effect on the shapes. Therefore, some morphology operations as dilation and erosion[0][0] are applied to smooth the image. Besides, to eliminate redundant pixels, a modification of the Nagendraprasad-Wang-Gupta thinning algorithm[0] is used to thin the image (Figure 6).



2.2Feature tree extraction

Figure 3. Feature extraction sub-stages.

Later, the feature extraction sub-stages are applied to the filtered image (Figure 3). Dominant points in shapes[0][0] refer to points in one of the following sets(Figure 7):

- End points of the shape (that is, points simply connected).
- Points corresponding to local extremes of curvature.
- Intersection points.

These points are used to describe sudden changes in the lines of the cursive handwritten word.

The tree features are extracted from the direction primitives between dominant points. The method takes as root of the feature tree the first dominant point from the left and builds the rest of the tree by following the neighbouring dominant points. Each node of the feature tree is labelled with a string which describes how to reach that dominant point from its ancestor in the tree. This string is obtained using nine possible directions (Figure 8). An example is shown in Figure 9.

These feature trees are expressed as a tree code chain (Figure 10) and used as input to the algorithm computing the distance between trees[0][0].

The meaning of the features obtained from word *hello* shown in Figure 10 is: label 63_1 represents a vector of length *63* in direction 1; label 34_3 stands for a vector of length *34* in direction *3*; however, label 70_9 represents a loop of length 70. As an example, a graphic representation of a short string is shown in Figure 11.

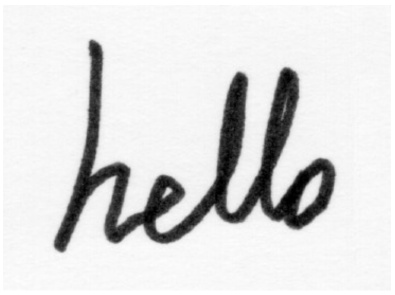


Figure 4. Greyscale image.

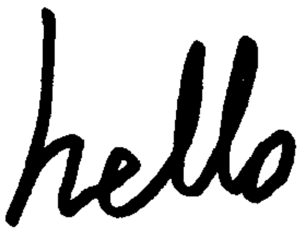


Figure 5. Bitmap image

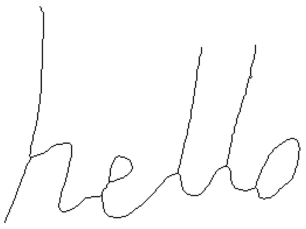


Figure 6. Filtered image (dilated + eroded + thinned)

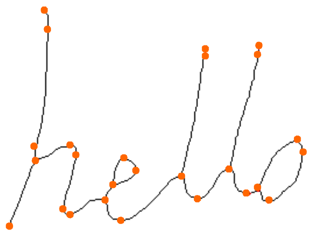


Figure 7 . Dominant points

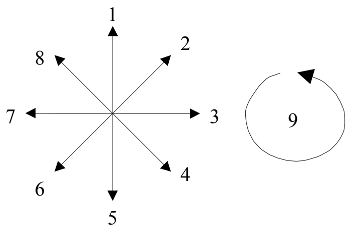


Figure 8. Codes used for straight-line segments and loops

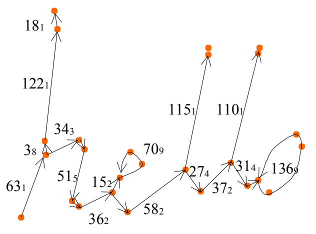


Figure 9. Tree features ($NORM_{direction}$). The norm of the vector is given and a subscript indicates the direction.

hello 23 (63;1(34;3(9;4(51;5(8;4(36;2(15;2(70;9())28;4(58;2(115;1(4;1())27;4(37;2(110;1(5;1())31;4 (8;3(136;9())))))))))))3;8(122;1(18;1()))))

Figure 10. Tree code chain for the word in Figure 9.

$$\begin{tikzcd} & \boxed{()} \arrow[dl] \arrow[dr] & \\ \boxed{34_4} \arrow[d] & & \boxed{45_9} \\ \boxed{22_2} & & \end{tikzcd}$$

Figure 11. Graphic representation of 4 (34;4(22;2())45;9()).

The distance used is such that, if two word images show a small difference, for instance, a disconnected letter (Figure 12 compared to Figure 4), the distance remains small due to the way the feature tree is obtained. Recall that the algorithm takes the next dominant point, which is not used, from left to right. For instance, the distance between the feature tree of the word in

Figure 10 and that in Figure 14 is only 78 units, whereas the distance between the feature tree of the word in Figure 10 and that in Figure 15 is 654 units.

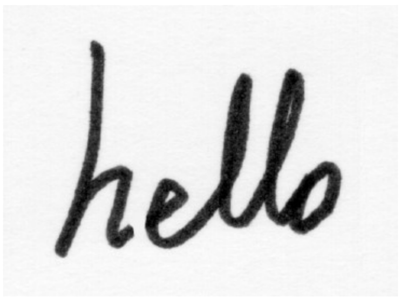


Figure 12. ‘hello’ image with an isolated ‘h’

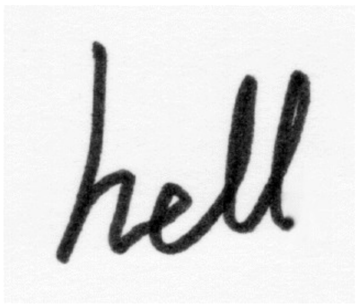


Figure 13. ‘hell’ greyscale image.

hello 24 (63;1(34;3(9;4(51;5(8;4(12;3(26;3(16;2(70;9())27;4(58;2(115;1(4;1())27;4(37;2(109;1(5;1())31;4(8;3 (136;9())))))))))))))3;8(121;1(18;1()))))

Figure 14. Tree code chain associated to ‘hello’ Figure 12.

hell 21 (75;1(40;3(11;4(60;5(10;4(42;2(16;2(83;9())33;4(68;2(136;1(5;1())32;4(44;2(129;1(6;1())32;4()))))))))) 3;8(143;1(22;1()))))

Figure 15. Tree code chain associated to ‘hell’ Figure 13.

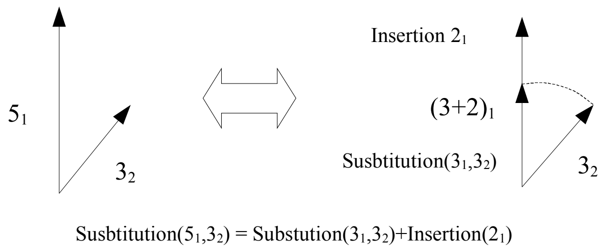
3.Classification

In this paper an edit distance between trees is used[0][0] which uses weights for each of these three operations: insertion, deletion or substitution. In addition, a weight adaptation algorithm is proposed to increase the rate of classification. This algorithm changes the weights applied to the distance between trees to obtain the best classification rate for words in the database.

Words of different cursive handwriting styles from four different writers have been collected. A database with 2.400 samples (600 for each writer) has been built. Each writer has written 50 different words and each word has been written 12 times. All words in the database have four or five letters. This poses an additional difficulty because the number of tree nodes is very similar for all words in the database.

The leaving-one-out[0] technique has been used to estimate the error rate. Each database prototype is compared with the rest and classified. The error rate is obtained by applying this method to all database prototypes. This technique uses the maximum information from test patterns to estimate the error rate.

The algorithm that has been used to compute the distance between trees is the one by Zhang and Shasha[0]. The algorithm is based on insertion, deletion and substitution edition functions. Each operation has an associated cost. Elementary insertions and deletions have the same cost $w_I = w_D$, and substituting a unary vector of type *a* by another of type *b* has cost w_{ab}. Given vectors M_a and N_b, deleting or inserting M_a has cost $M\, w_I$ and substituting M_a by N_b has cost $\min\{M, N\}\, w_{ab} + |M - N|\, w_I$. An example is shown in Figure 16.



5_1
3_2
Insertion 2_1
$(3+2)_1$
Susbtitution$(3_1,3_2)$
3_2
Susbtitution$(5_1,3_2)$ = Substution$(3_1,3_2)$+Insertion(2_1)

Figure 16. Example of substitution operation between vectors 5_1 and 3_2.

An iterative method has been used to adjust the substitution weights w_{ij}:

$$w_{ij}^{t} = w_{ij}^{t-1} + \alpha \frac{n'_{ij} - n_{ij}}{Max\{n'_{ij}, n_{ij}\}} \quad \begin{array}{l} 0 \le w_{ij} \le 4 \\ 0 \le \alpha \le 1 \end{array} \qquad w_{ij}^{0} = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 3 & 2 & 1 & 4 \\ 1 & 0 & 1 & 2 & 3 & 4 & 3 & 2 & 4 \\ 2 & 1 & 0 & 1 & 2 & 3 & 4 & 3 & 4 \\ 3 & 2 & 1 & 0 & 1 & 2 & 3 & 4 & 4 \\ 4 & 3 & 2 & 1 & 0 & 1 & 2 & 3 & 4 \\ 3 & 4 & 3 & 2 & 1 & 0 & 1 & 2 & 4 \\ 2 & 3 & 4 & 3 & 2 & 1 & 0 & 1 & 4 \\ 1 & 2 & 3 & 4 & 3 & 2 & 1 & 0 & 4 \\ 4 & 4 & 4 & 4 & 4 & 4 & 4 & 4 & 0 \end{bmatrix}$$

The weights are initialised to the difference between *i* and *j* orientation (Figure 8), α is a rate to accelerate the algorithm and n_{ij} is the number of due *ij-substitutions* in correctly classified prototypes and n'_{ij} is the number of substitutions of this kind in prototypes incorrectly classified. In this way, incorrectly classified prototypes increase their distance to the sample, while those correctly classified decrease their distances.

4.Results

The proposed method has been applied to a chosen set of data and the results are shown in Table 1 and in Figure 17. All samples are captured using a scanner with resolution of 300 ppi and 256 grey levels from pages wrote by four writers who used a black pen.

Different tests have been made and the highest classification rate obtained for the same writer was 87,5% while the maximum average classification rate was 85% (Table 1).

The simple algorithm (without adaptation weights) has been applied and the results ranged from 54% to 87,5% depending of the number of prototypes and the writer. For the weights adaptation algorithm the results ranged from 56,65% to 87,5% depending of the number of prototypes and the writer.

	Without adaptation algorithm		With adaptation algorithm	
Number of Prototypes	Correct classification Average (%)	Standard Deviation	Correct classification Average (%)	Standard Deviation
150	60	6.3	63	5.2
300	76	3.8	77	3.3
450	82	2.4	83	2
600	85	2.4	85	2

Table 1. Word classification rate without and with adaptation weights algorithm.

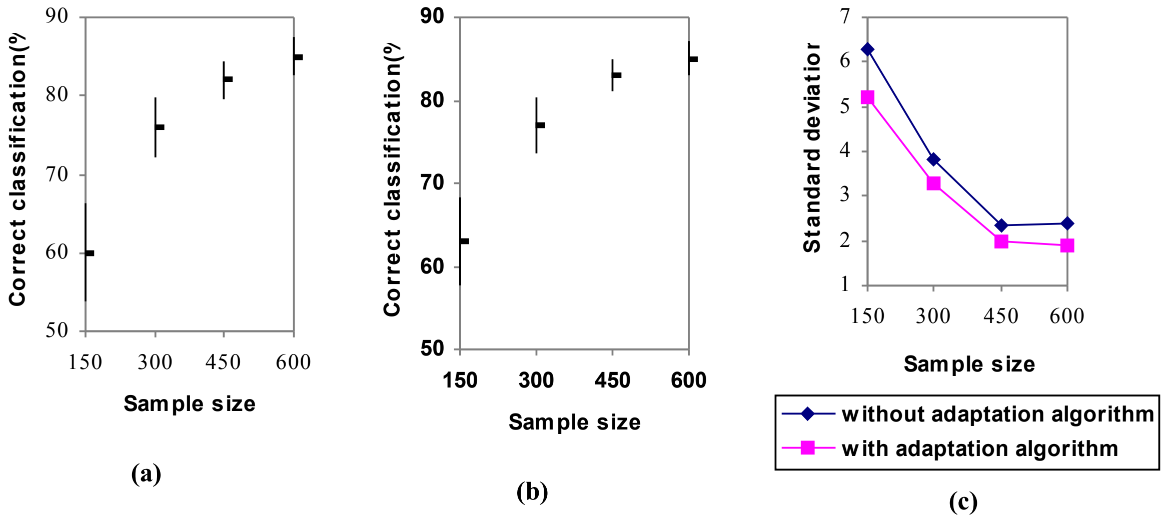


Figure 17. Average word classification rate (a) without adaptation algorithm and (b) with adaptation algorithm and (c) Standard deviation of the results with and without adaptation method.

The adaptation method proves more useful when there are few prototypes. Even if increasing the number of prototypes decreases the effect. The standard deviation decreases when adaptation method is applied (Figure 17c).

5.Conclusion

A method for off-line handwritten word recognition is explored. The method is based on feature tree extraction and an edit distance between trees. The algorithm is easy to apply and the results are promising. It uses examples from cursive handwritten words as a bitmap image. It is clear that if we increase the number of database examples we increase the time to get an answer from the classifier algorithm linearly. This linear dependency may be avoided algorithms as AESA (Approximating Eliminating Search Algorithm) or related[0][0]. If the database does not change, these algorithms compute the distance between some prototypes in pre-process time, and uses this precomputed information to speed up the search for the nearest neighbour.

Other algorithms[0] need additional information as well as bitmap words. For example, they have a restricted dictionary, it is difficult to increase the database with a new words, and they need a splitting algorithm[0][0][0] to separate words in to letters. The algorithm proposed in this paper is a new point of view about cursive off-line handwriting recognition problem.

For future work, the efforts can focus on making this algorithm more flexible and quicker. An interesting idea would be applying it to a signature recognition system.

6. References

[0] R. C. Carrasco and M. L. Forcada: "A note on the Nagendraprasad-Wang-Gupta thinning algorithm". Pattern Recognition Letters, Vol. 16, pp 539-541 (1995).

[0] K. Huang and H. Yan: "Off-line signature verification based on geometric feature extraction and neural classification". Pattern Recognition, Vol. 30, No 1, pp 9-17 (1997).

[0] X. Li and D. Yeung: "On-line handwritten alphanumeric character recognition using dominant points in strokes". Pattern Recognition, Vol. 30, No 1, pp 31-34 (1997).

[0] R. K. Powalka, N. Sherkat and R. J. Whitrow: "Word shape analysis for a hybrid recognition system". Pattern Recognition, Vol. 30, No 3, pp 421-445 (1997).

[0] J. Wang, Maylor, K. H. Leung and S. Cheung Hui: "Cursive word reference line detection". Pattern Recognition, Vol. 30, No 3, pp 503-511 (1997).

[0] B. J. Oommen, K. Zhang and W. Lee: "Numerical similarity and dissimilarity measures between two trees". IEEE Transactions on Computers, Vol. 45, (1996).

[0] K. Zhang and D. Shasha: "Simple fast algorithms for the editing distance between trees and related problems". SIAM Journal of Computing, Vol. 18, pp. 1245-1262 (1989).

[0] L. Micó, J. Oncina and R. C. Carrasco: "A fast branch & bound nearest neighbour classifier in metric spaces". Pattern Recognition Letters, Vol. 17, pp 731-739 (1996)

[0] L. Micó and J. Oncina: "Comparison of fast nearest neighbour classifiers for handwritten character recognition ". Pattern Recognition Letters, (to be published), (1998)

[0] R. O. Duda & P. E. Hart: *Pattern Classification and Scene Analysis*. John Wiley and Sons (1973)

[0] S. Serra: *Image analysis and mathematical morphology*. Academic Press (1989-1992)

[0] C. Y. Suen, M. Berthod and S. Mori: "Automatic recognition of handprinted characters – the state of art". Proc. IEEE, 68(4):469-487, April 80. Contains 244 references.

[0] S. Impedevo, G Dimauro and G. Pirlo: "A new decision tree algorithm for handwritten numerals recognition using topological features". Proc. SPIE, 1384:280-284, November 1990.

[0] D. G. Elliman and R. N. Banks: "A comparison of two neural networks for hand-printed character recognition". In IEE 2nd Neural Networks, number 349 in IEE, pages 224-228, November 1991.

[0] A.W. Senior and A.J.Robinson. "An Off-line Cursive Handwriting Recognition System". IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20. No. 3, pp 309-321 (1998).

Acknowledgements: The author thanks the Dirección General de Investigación Científica y Técnica (CICYT) of the Government of Spain for support through project TIC97-0941, as well as R. C. Carrasco and M. Forcada for their careful reading of this paper.