A SIMPLE AND EFFECTIVE CURSIVE WORD SEGMENTATION METHOD[1]

G.NICCHIOTTI AND C.SCAGLIOLA

Elsag spa Via Puccini 2 - 16154 Genova, - ITALY
E-mail: gianluca.nicchiotti@elsag.it, carlo.scagliola@elsag.it

S. RIMASSA

Polo Nazionale Bioelettronica Via Roma 28 57030 Marciana (LI) - ITALY
E-mail: simone.rimassa@mailcity.com

A simple procedure for cursive word oversegmentation is presented, which is based on the analysis of the handwritten profiles and on the extraction of "white holes". It follows the policy of using simple rules on complex data and sophisticated rules on simpler data. Experimental results show robustness and performances comparable with the best ones presented in the literature.

1 Introduction

Segmentation is the operation that seeks to decompose a word image in a sequence of subimages containing isolated characters. Segmentation is a critical phase of the single word recognition process, and this is witnessed by the higher performance for the recognition of isolated characters vs. that obtained for cursive words.

There are two main strategies for segmentation [1]. Straight segmentation [2,3] tries to decompose the image in a set of subimages, each one corresponding to a character. In segmentation-recognition strategies [4-7] the image is subdivided in a set of subimages (strokes) whose combinations are used to generate character candidates. The number of subimages is greater than the number of characters and the process is referred to also as oversegmentation. Recognition is then used to select the correct character hypothesis from character candidates. The quality of the oversegmentation process depends on the tradeoff between the number of missed detections of ligatures and the ratio between the strokes produced and the number of characters. The aim of straight segmentation is obviously more ambitious, hence suitable for simpler tasks like segmentation of typewritten or hand printed words.

In the following we will present an effective segmentation method based on a very simple model of character ligature. Experimental results show that the simple strategy adopted is also effective.

[1] This work was funded by the Italian Ministry of University and of Scientific and Technological Research under the grant to Parco Scientifico e Tecnologico dell'Elba, Project No. 62413, for the research line "Neural devices for the recognition of cursive handwriting".

In: L.R.B. Schomaker and L.G. Vuurpijl (Eds.), Proceedings of the Seventh International Workshop on Frontiers in Handwriting Recognition, September 11-13 2000, Amsterdam, ISBN 90-76942-01-3, Nijmegen: International Unipen Foundation, pp 499-504

2 System overview

The segmentation process we will describe is a part of "Corsivo Elba" single word recognizer [8]. The flow chart of the system is sketched in Fig. 1. It consists basically of four main building blocks: preprocessing [9], character recognition [10], word interpretation [8] and the segmentation procedure we describe next.

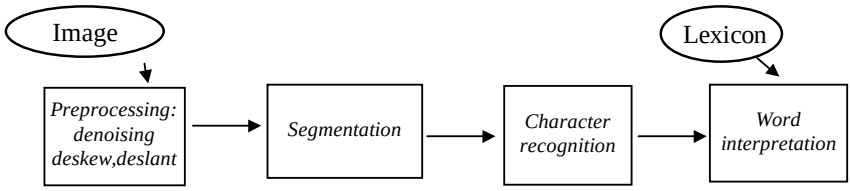

Figure 1: Corsivo Elba Flow Chart

More details of the complete system are reported in [8]. We only recall here that the isolated character recognition performance is 80.3% and that the overall performance of "Corsivo Elba", measured on a test set of 500 word images extracted from the CEDAR database [11], against a 1000 word lexicon, is 83.8 %.

3 The Segmentation Procedure

The intent of our procedure is to "oversegment" the word image, i.e. to cut the image in sufficiently many places that the correct segmentation boundaries are included among the cuts made.

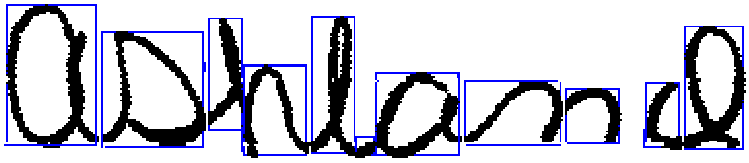

Figure 2: Oversegmentation. Note that some letters (h, n, d) have been dissected into multiple parts. However no pairs of merged character remain, so that a correct segmentation can be produced by the recombination of some of the segments.

A failure to detect a legature could result in a fatal error during word recognition, but, for the success of the entire recognition process, it is also important that the number of segmentation points is kept as small as possible to reduce the possibility of confusion among words. The oversegmentation process consists of 3 main steps:

1. Detection of Possible Segmentation Points,
2. Stroke generation,
3. Stroke analysis.

The oversegmentation algorithm processes the input image after denoising, deslanting and deskewing are applied as described in [9]. The estimated pen stroke thickness and the positions of baseline and upperline are used as auxiliary data during the process.

3.1 Possible Segmentation Points Detection

A simple ligature model is used; the features used to detect ligatures are very simple to detect too. Such features are the handwriting contour minima and the holes, which are white connected regions.

Ligatures are sought where the word contours present minima. Three contours are extracted for each image: the upper contour, the median contour and the lower contour. Median contour is defined only for the columns with 3 vertical black runs; it follows the profile of the writing below the ascenders and above the descenders and it allows us to detect ligatures hidden by either the upper contour or lower contour. Local minima are searched along each contour. They are considered valid PSPs only if the contour can be computed in every point of the neighborhood of the minimum.

Other possible segmentation points are located outside the holes belonging to the region between baseline and upperline (core region). In neat cursive handwriting, holes are present in the following letters: *a b d e f g h l o p q*. Each hole of this subset plays a double role in segmentation points detection: it forbids the presence of segmentation points in its interior and it forces the presence of two segmentation points on either side.

Holes' detection and local minima contour provide a set of PSPs. Some of these could be very close to one another and hence, if the distance between two PSPs is lower than a given percentage of the average stroke's thickness, the two PSPs are merged into a single PSP.

3.2 Cut's direction determination

Segments, or strokes, are found by a growing algorithm on black pixels of the image. Therefore, to cut a word image, we chose to switch to white the black pixels below the segmentation points.

Each segmentation point gives birth to a cut. In most cases the cuts are led along vertical directions, but sometimes this is not the best way to cut the image. A cut in the vertical direction may sometimes be too long, thus making it quite difficult to properly reconstruct the characters. As shown in Fig. 3, if the two letters are separated along the vertical direction, it will be quite hard to recognize correctly the letter *a*. In this case the slanted cut's direction is the correct one. So, if the length of the vertical cut is over a certain length, parameterized by the width

of the core region, other directions are tested in the range ±45° around the vertical one. The direction that involves the least number of pixels is chosen for the cut.

Figure 3: Comparison between vertical cut and slanted cut.

3.3 Stroke generation: aggregation, rejection.

Cutting divides the image into strokes, that are the black connected regions of the cut image. Each stroke is a potential hypothesis of character, but some of them can be evaluated, immediately, as unlikely hypotheses of character when they are taken alone. More in detail, strokes are considered inconsistent when they comply with one of the following requirements:

- Area (number of pixels) lower than a threshold parameterized by the width of the core region and the estimate stroke thickness.
- Bounding box all below the baseline or all over the upperline.

Inconsistent strokes are discarded or merged with adjacent consistent ones . The merging is carried out by means of a recursive procedure. This enables multiple adjacent inconsistent strokes to be merged with the same consistent one. Fig. 4 shows the results of segmentation and aggregation/rejection of strokes for the word “Vegas”.

4 Experimental results and conclusions

A total of 850 word images (6009 characters) extracted from the TRAIN directory of CEDAR database [11] were used to test the segmentation algorithm.

In order to measure the performance of the segmentation process, missing detections, hypersegmented words and the average number of strokes per character are computed. There is a missing detection when the algorithm misses to cut two adjacent characters, and there is an hyper-segmentation when the algorithm cuts a character in a number of strokes greater then the maximum number of strokes

allowed. In our system such maximum number is set to 4 for lower case and to 6 for upper case letters.

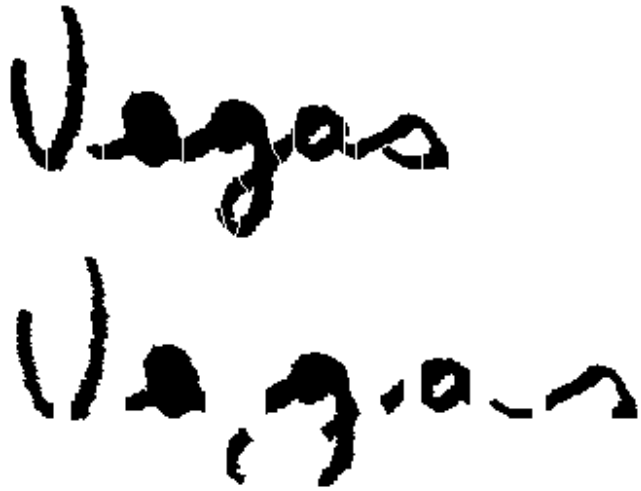

Figure 4: Segmentation result for image of the word Vegas: some little strokes are discarded, this helps to keep the number of strokes as small as possible.

The segmentation results are shown in table 1

Table 1: Segmentation results

Correctly segmented word	86.9%
Intercharacter ligatures detected	97.9%
Stroke character ratio	1.8
Hypersegmented images	1.7%

The algorithm segmented 86.9% of the test set images with no missing detection, only 1.7% of images contain hypersegmented characters and 97.9% of the ligatures were correctly detected. The stroke character ratio is 1.8. These results outperform the ones reported by [12], [13] and [6] and are comparable with the performance reported by [4].

Most of missing PSPs come from “odd” ligatures, which cannot be detected by holes and minima contours criteria. The existence of missing PSPs would justify implementation of ad hoc procedures to segment hardly detectable ligatures, like the single-run stretch splitter described in [6], but experiments have proved that the recognition algorithm performs best without special procedures. These procedures decrease the number of missing PSPs but the price to pay is a relevant increase of false segmentation points.

In conclusion, we have presented a simple and effective segmentation algorithm which proved to produce excellent results. The algorithm demonstrates to be also stable, i.e. the number of strokes in which the same letter is splitted has small variations. This property allowed us to use the information of duration probability, i.e. the frequency with which each letter is splitted into different number of strokes [14], to improve word recognition accuracy [8].

References

1. Casey R.G. and Lecolinet E., A survey of methods and strategies in character segmentation. *IEEE Trans. PAMI* **18** (7) 1996 pp. 690-706.
2. Cesar M. and Shingal R., Algorithm for segmenting handwritten postal codes. *Int'l J. Man Machine Studies* **33** (1) 1990 pp. 63-80.
3. Baird H.S., Kahan S. and Pavlidis T., Component of an Omnifont Page Reader. *Proc 8th ICPR* Paris 1986 pp. 344-348.
4. Yanikoglu B. and Sandon P.A., Segmentation of off-line cursive handwriting using linear programming. *Patt. Recog.* **31** (12) 1998 pp. 1825-1833.
5. Bozinovic R.M. and Shrihari S.N., Off-line cursive script recognition. *IEEE Trans. PAMI* **11** (1) 1989 pp. 68-83.
6. Kimura F. et al., Improvements of a Lexicon Directed Algorithm for Recognition of Unconstrained Handwritten Words. *Proc 2nd ICDAR* Tsukuba October 20-22 1993 pp. 18-22.
7. Romeo-Pakker K. et al. A New Approach for Latin Arabic Character Segmentation *3rd ICDAR*, Montreal, August 14-16, 1995, pp 874-877.
8. Scagliola C., Nicchiotti G. and Camastra F., Enhancing Cursive Word Recognition Performance by the Integration of all the Available Information. *7th IWFHR*, Amsterdam, September 11-13, 2000.
9. Nicchiotti G. and Scagliola C., Generalised Projections: a Tool for Cursive Handwriting Normalisation. *Proc. 5th ICDAR* Bangalore 1999, pp 729-733.
10. Camastra F. and Vinciarelli A., Isolated Cursive Character Recognition by Neural Nets. *Kuenstliche Intelligenz* **2** (1999) pp. 17-19.
11. Hull J.J., A Database for Handwritten Text Recognition Research. *IEEE Trans. PAMI* **16** (5) 1994 pp 550-554.
12. Leedham C.G. and Friday P.D., Isolating Individual Handwritten Characters *Proc. IEE Colloq.on Character recognition and applications* 1989.
13. Han K. and Sethi I K, Off-line cursive handwriting segmentation. *Proc 3rd ICDAR* Montreal 1995 pp 894-897.
14. Kim G. and Govindaraju V., A Lexicon Driven Approach to Handwritten Word Recognition for Real-Time Applications. *IEEE Trans. PAMI* **19** (4) 1997 pp366-379.