A Contour Code Feature Based Segmentation For Handwriting Recognition

Brijesh Verma
School of Information Technology, Griffith University-Gold Coast Campus, Australia
E-mail: b.verma@griffith.edu.au

Abstract

The purpose of this paper is to present a novel contour code feature in conjunction with a rule based segmentation for cursive handwriting recognition. A heuristic segmentation algorithm is initially used to over segment each word. Then the prospective segmentation points are passed through the rule-based module to discard the incorrect segmentation points and include any missing segmentation points. The proposed rule-based module validates every segmentation points against closed area, average character size, left character and density. During the left char validation, a contour code feature is extracted and checked weather the left of the prospective segmentation point is a character or rubbish (non-char). The neural network used for this validation was trained on character and non-character database. Following the segmentation, the contour between correct segmentation points is passed through the feature extraction module that extracts the contour code, after which another trained neural network is used for classification. The recognized characters are grouped into words and passed to a variable length lexicon that retrieves words that has highest confidence value.

1. Introduction

Handwriting recognition is one of the very challenging problems. An overview of handwriting recognition process is shown in Figure 1.

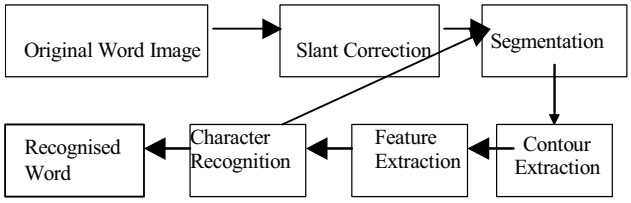

Figure 1 Handwriting recognition process

In the literature, many papers have been published with research detailing new techniques for the classification of handwritten numerals, characters and words. Some researchers have obtained very promising results for isolated/segmented numerals and characters using neural network based techniques [1-5]. However, the results for the segmentation and recognition of touching handwritten words, have not been very good and still there is a need for improvement so that they can be used in real world applications. Some researchers have used heuristic and statistical techniques for character segmentation and recognition respectively [6, 7] while others have used heuristic techniques for segmentation followed by neural network based methods for the character/word recognition process [8, 9]. There have been a number of researchers using intelligent techniques for segmentation of handwriting [10, 11]. As it is mentioned in the literature [12-15], segmentation plays an important role in the overall process of handwriting recognition. It has also been mentioned that the feature extraction is one of the most significant parts of any classification system and it plays an important role in improving the segmentation process and overall recognition.

In this research we propose a novel contour code feature in conjunction with a rule-based segmentation for the improvement of handwriting recognition.

The remainder of the paper is broken down into 3 sections. Section 2 describes the proposed methodology, Section 3 presents experimental results and Section 4 presents a conclusions.

2. Proposed Research Methodology

The proposed research methodology is described in the following sections.

2.1 Segmentation

An overview of segmentation is shown below in Figure 2.

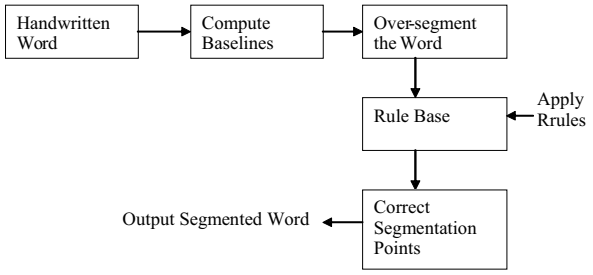

Figure 2 An overview of segmentation

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

The segmentation follows the steps below:
Step 1: Compute baselines
Step 2: Over-segment the word
Step 3: Pass the segmentation points through the rule base
Step 4: Remove the incorrect segmentation points
Step 5: Output the correct segmentation points

2.1.1. Baseline computation. Baseline computation is an important technique in handwriting recognition. Baselines are used for size normalization, correcting rotation, extracting features etc. In this approach we are computing five lines for the segmentation purpose. Those are upper baseline, lower baseline, middle baseline, ascender line and descender line. All the baselines are computed with the respect to the horizontal pixel density.

Upper baseline: Upper baseline is the line that goes through the top of the lower case characters (e.g. a, n etc.). The line is shown in Figure 3.

Lower baseline: Lower baseline is the line that goes through the bottom of the lower case characters (e.g. a, n etc.). The line is shown in Figure 3.

Middle baseline: The middle baseline corresponds to the writing line on which the word is written. The line is shown in Figure 3.

Ascender line: Ascender line corresponds to the line passes through the topmost point of the word. The line is shown in Figure 3.

Descender line: Descender line corresponds to the line passes through the bottom most point of the word. The line is shown in Figure 3.

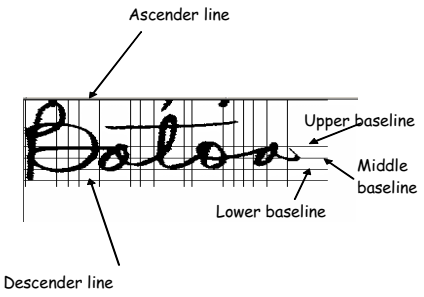

Figure 3 Baselines

2.1.2. Over Segmentation. This module is used to assign a Candidate (prospective) Segmentation Point (CSP) that could be validated through the rule-based module for further processing. A heuristic over segmentation algorithm is used that incorporate the vertical histogram change. A vertical histogram is drawn at each point in column and the change in vertical density is noted. Where the change is drastic, the possible candidate segmentation point is drawn.

2.1.3. Rule based validation. The over-segmented word is passed through the rule-base module where rules are written on the basis of the contour characteristic of a character (such as a loop, a hat shape etc.) described in the following section. According to the rules the incorrect segmentation points are removed form the over-segmented word.

Rule 1: Detect a loop (closed area) and remove the segmentation points within a loop. Add a segmentation point after end of the loop as a Candidate Segmentation Point (CSP). An example is shown below in Figure 4.

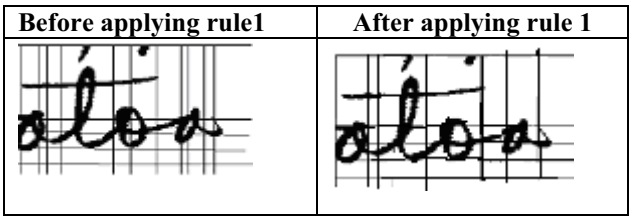

Figure 4 Rule 1

Rule 2: Detect the hat shape and remove the segmentation point within the hat shape contour. Add an extra segmentation point after the end point of the hat shape. The hat shape is described as the '∧' or '∨'. An example is shown below in Figure 5.

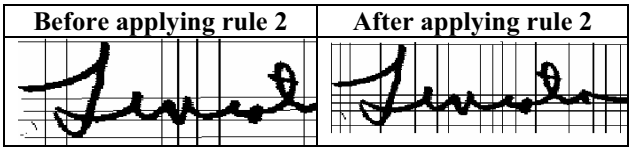

Figure 5 Rule 2

Rule 3: If the average area between two prospective segmentation points is less than the average width of the character, remove the segmentation point through validation of the neural network. An example is shown below in Figure 6.

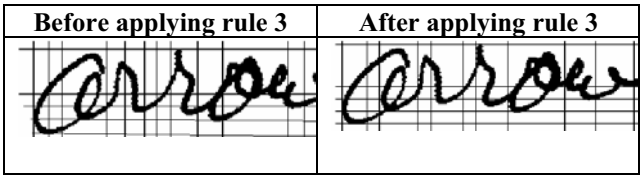

Figure 6 Rule 3

Rule 4: Add missing segmentation point. The missing points are detected by the threshold between two segmentation points. The average distance between two segmentation points is calculated taking the average of all

segmentation points. If the distances cross the threshold value a Candidate Segmentation Point is added as missing one.

2.2 Contour extraction

This section describes the proposed methodology for the extracting the contour between the two segmentation points. The contour between two consecutive segmentation points is extracted using the following few steps. In the first step disconnect the pixels near the first segmentation point; disconnect the pixels near the second segmentation point. Find the nearest distance of the first black pixel from the first segmentation point and the three baselines. Follow the contour path across that baseline having minimum distance is closest. Find the connecting contour. Mark it as visited once it is visited. If the contour is already visited then discard that, take the other part if any.

2.3 Proposed contour code feature extraction

A novel feature extraction technique is proposed to extract the feature between the two contours. The values for feature extracted are structural feature from the contour profile. So the feature is named as contour code feature. The rate of change of slope along with the point where it is changing is extracted. With the contour slop, a few additional values that count number of ascenders, number of descenders, start point, end point, etc. are taken into consideration to capture the structural properties of the contour profile. The contour code feature vector size is taken as 25. The contour code feature is described below.

Slope: The slope of the consecutive points is estimated using linear regression. The rate of change of slope is used as the main feature. The feature is described below in Figure 7.

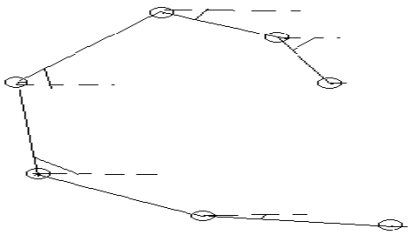

Figure 7 Contour code feature

The input to the contour code feature extraction module is the set of coordinate (x, y) of the contour extracted from the contour extraction phase. With the coordinate the slope of two consecutive points are estimated. The slope estimation is done using linear probing. Linear regression attempts to explain this relationship with a straight line fit to the data. The linear regression model postulates that

$$Y = a + bX$$

The coefficients *a* and *b* are determined by the following equations.

$$b = \frac{2\left(x_1 y_1 + x_2 y_2\right) - \left(x_1 + x_2\right)\left(y_1 + y_2\right)}{2\left(x_1^2 + x_2^2\right) - \left(x_1 + x_2\right)^2}$$

$$a = \frac{\left(y_1 + y_2\right) - b\left(x_1 + x_2\right)}{2}$$

The slope is between the two points (x1, y1) and (x2, y2) is represented by the parameter b. The following values are calculated and stored as a contour code in a single dimension vector.

Point of Change: The point with respect to the main body of the contour where the slope is changing is taken.

Direction Change (Up/Down): The point with respect to the main body of the contour where the direction is changing is taken. The change of direction is denoted by the contour, which is changing the direction upwards to downward or vice versa.

Number of Ascenders: The number of point above the upper baseline is counted and stored.

Number of Descenders: The number of point below the lower baseline is counted and stored.

Start Point: Start point of a character (position with respect to baselines) is detected and stored.

End Point: End point of a character (position with respect to baselines) is detected and stored.

2.4 Recognition of characters

A feed forward ANN trained with the tradition back propagation algorithm is used to recognize the segmented characters. Two separate neural networks are trained for upper and lower case characters. Another neural network was trained for the validation of the segmentation point.

2.5 Recognition of words using lexicon

A variable sized lexicon words was implemented to recognize all words used for testing. The lexicon used is a simple string comparison algorithm, which first matched

each character of each lexicon word to the characters in the test word being examined.

3. Implementation and experimental results

This section describes the database, the implementation platform and the experimental results.

3.1 Database

A number of experiments were conducted. Samples of handwritten words from CEDAR benchmark dataset [16] were used to test the segmentation module. The character dataset of the CEDAR was also used to train the neural network classifiers.

3.2 Implementation

All the algorithms were implemented in C++ on a UNIX platform.

3.3 Experimental results

The experiments were conducted in three sets. The first set of experiment was conducted to test the segmentation approach. A single hidden layer neural network was used with 25 inputs, 40 hidden units and 53 outputs to detect the rubbish contour. In the second set of experiment the contour code feature was tested. Two separate feed forward neural networks with 25 inputs, 30 hidden units and 26 outputs were used for lower and upper case characters. Third set of experiment was conducted to test the word recognition rates.

3.3.1. Segmentation results

To test the accuracy of the rule-based segmentation approach in conjunction with novel feature extraction, three criteria [17, 18, 19] were used. Those are 1) number of over segmentations, 2) missed segmentations and 3) bad segmentations. Over segmentation is denoted when the character is segmented by more than two segmentation points. Missed segmentation is denoted when a correct segmentation point is not noted by the segmentation approach. Bad segmentation denotes the segmentation point that does not separate two characters properly. The error rates are shown in Table 1.

Table 1 Segmentation errors

Over segmentation (%)	Missed (%)	Bad (%)
10.02	0.2	8.7

As shown in Table 1, the segmentation algorithm performed reasonably well. The missed error rate was almost zero (0.2%). The over segmentation error was prominent but not excess (10.02%). The bad segmentation error obtained was also medium (8.7%).

3.3.2. Segmented character recognition results. The character recognition experiments were conducted using a back propagation neural network. The number of characters used for training and testing were 1500 and 1200 respectively. The number of outputs was 26 representing uppercase characters (A-Z) and 26 representing lowercase characters (a-z). The segmentation and recognition approach was tested with the proposed contour code feature and the transition feature. The results obtained for that character recognition are shown in Tables 2 – 5.

Table 2 Character recognition results for Lower case (Training Dataset)

Hidden Units	Classification rate [%] with transition feature	Classification rate [%] with contour code feature
10	96.87	97.45
20	98.23	98.56
30	99.67	100

Table 3 Character recognition results for Lower case (Testing Dataset)

Hidden Units	Classification rate [%] with transition feature	Classification rate [%] with contour code feature
10	75.56	76.54
20	82.23	84.12
30	83.46	86.84

Table 4 Average (Upper and Lower) character recognition results (Training Dataset)

Hidden Units	Classification rate [%] with transition feature	Classification rate [%] with contour code feature
10	95.23	92.42
20	98.32	94.21
30	99.14	96.87

Table 5 Average (Upper and Lower) character recognition results (Testing Dataset)

Hidden Units	Classification rate [%] with transition feature	Classification rate [%] with contour code feature
10	82.25	80.08
20	84.93	82.00

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

30	85.83	83.84

3.3.2. Word recognition results. The word recognition results are shown in Table 6. The words are passed through the lexicon analyser. The word recognition rate after passing through the lexicon was 93%. The overall results on a small lexicon are very good.

Table 6 Word recognition results

Lexicon size	Word recognition [%]
10	93
20	91

4. Conclusions

A novel contour code feature based segmentation approach has been presented in this paper that produces promising results. It was used to segment difficult cursive words from CEDAR benchmark database. The segmentation results are much better than published in the literature. The novel contour code feature extraction technique was compared with a transition feature and it was found that the proposed contour code feature produced average much higher recognition rates. The word recognition rate on small database is very promising.

Acknowledgement

This work was funded by Australian Research Council (ARC) small grant. We would like to thank the ARC for their support and also thanks to research assistant Moumita Ghosh for conducting a number of useful experiments included in this paper.

References

[1] C. Y. Suen, R. Legault, C. Nadal, M. Cheriet, and L. Lam, "Building a New Generation of Handwriting Recognition Systems", Pattern Recognition Letters, Vol. 14, 1993, pp. 305-315.

[2] S-W. Lee, "Multilayer Cluster Neural Network for Totally Unconstrained Handwritten Numeral Recognition", Neural Networks, Vol. 8, 1995, pp. 783-792.

[3] H. I. Avi-Itzhak, T. A. Diep, and H. Garland, "High Accuracy Optical Character Recognition using Neural Networks with Centroid Dithering", IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 17, 1995, pp. 218-224.

[4] S-W. Lee, "Off-Line Recognition of Totally Unconstrained Handwritten Numerals Using Multilayer Cluster Neural Network", IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 18, 1996, pp. 648-652.

[5] S-B. Cho, "Neural-Network Classifiers for Recognizing Totally Unconstrained Handwritten Numerals", IEEE Trans. on Neural Networks, Vol. 8, 1997, pp. 43-53.

[6] R. M. Bozinovic, and S. N. Srihari, "Off-Line Cursive Script Word Recognition", IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 11, 1989, pp. 68-83.

[7] N.W. Strathy, C.Y. Suen, and A. Krzyzak, "Segmentation of Handwritten Digits using Contour Features", ICDAR '93, 1993, pp. 577-580.

[8] B. A. Yanikoglu, and P. A. Sandon, "Off-line cursive handwriting recognition using style parameters", Tech. Report PCS-TR93-192, Dartmouth College, NH., 1993.

[9] J-H. Chiang, "A Hybrid Neural Model in Handwritten Word Recognition", Neural Networks, Vol. 11, 1998, pp. 337-346.

[10] G. L. Martin, M. Rashid, and J. A. Pittman, "Integrated Segmentation and Recognition through Exhaustive Scans or Learned Saccadic Jumps", Int'l J. Pattern Recognition and Artificial Intelligence, Vol. 7, 1993, pp. 831-847.

[11] B. Eastwood, A. Jennings, and A. Harvey, "A Feature Based Neural Network Segmenter for Handwritten Words", ICCIMA'97, Gold Coast, Australia, 1997, pp. 286-290.

[12] S. N. Srihari, "Recognition of Handwritten and Machine-printed Text for Postal Address Interpretation", Pattern Recognition Letters, Vol. 14, 1993, pp. 291-302.

[13] M. Gilloux, "Research into the New Generation of Character and Mailing Address Recognition Systems at the French Post Office Research Center", Pattern Recognition Letters, Vol. 14, 1993, pp. 267-276.

[14] R. G. Casey and E. Lecolinet, "A Survey of Methods and Strategies in Character Segmentation", IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 18, 1996, pp. 690-706.

[15] Y. Lu, M. Shridhar, "Character Segmentation in Handwritten Words – An Overview", Pattern Recognition, Vol. 29, 1996, pp. 77-96.

[16] J. J. Hull, "A Database for Handwritten Text Recognition", IEEE Transactions of Pattern Analysis and Machine Intelligence, Vol. 16, 1994, pp. 550-554.

[17] B. Yanikoglu and P. A. Sandon, "Segmentation of Off-Line Cursive Handwriting using Linear Programming", Pattern Recognition, Vol. 31, 1998, pp. 1825-1833.

[18] X. Xiao and G. Leedham, "Knowledge-based Cursive Script Segmentation, Pattern Recognition Letters", Vol. 21, 2000, pp. 945-954.

[19] M. Blumenstein, and B. K. Verma, "Analysis of Segmentation Performance on the CEDAR Benchmark Database", ICDAR'01, 2001, pp. 1142-46.