Proceedings of the 9th International Conference on Neural Information Processing (ICONIP'02), Vol. 5
Lipo Wang, Jagath C. Rajapakse, Kunihiko Fukushima, Soo-Young Lee, and Xin Yao (Editors)

A CONTOUR CHARACTER EXTRACTION APPROACH IN CONJUNCTION WITH A NEURAL CONFIDENCE FUSION TECHNIQUE FOR THE SEGMENTATION OF HANDWRITING RECOGNITION

Brijesh Verma
School of Information Technology, Griffith University-Gold Coast Campus
PMB 50, Gold Coast Mail Centre, QLD 9726, Australia
E-mail: b.verma@mailbox.gu.edu.au

ABSTRACT

The purpose of this paper is to present a novel neural network based algorithm to improve the segmentation process of cursive handwriting recognition and a detailed analysis of the performance of the algorithm on a benchmark database. The algorithm is based on a technique to fuse left character, center character and neural validation confidence values. A technique is proposed to extract a character between two segmentation points, which avoids vertical segmentation. Also a fusion technique and a technique to over-segment the words are described in this paper. A large number of experiments were conducted and an extensive analysis of comparative results on a benchmark database is included. The segmentation results obtained are very promising.

1. INTRODUCTION

Handwriting recognition is one of the very challenging and difficult problems. Recently, a number of papers have been published with research detailing new techniques for the classification of handwritten numerals and words. Many researchers have obtained very promising results for isolated/segmented numerals using neural network based techniques [1-5]. Unfortunately, satisfactory results have not yet been obtained for the segmentation and recognition of handwritten words. 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 only been a few researchers using intelligent techniques for segmentation of handwriting [10, 11]. However, as it is mentioned in the literature [12-15], segmentation plays an important role in the overall process of handwriting recognition.

In this research we focus on the segmentation process of the handwriting recognition system. One of the most common reasons for the inadequate performance of cursive handwriting recognition systems has been attributed to inaccurate segmentation. Segmentation is the process of separating the characters in a word, so that they may be used to assist in final word interpretation. Some systems use the method of over-segmentation to dissect the word at many intervals into primitives. The term "primitive" refers to entire characters or character components. Following initial over-segmentation, various techniques may be used to correctly assemble the primitives using contextual processing to recognise entire words. The removal of incorrect segmentation points from over-segmented words is still a difficult problem. A solution to this problem would guarantee a higher success rate for handwritten word recognition. This research investigates a novel algorithm for removal of incorrect segmentation points. The algorithm analyses the surroundings of every suspicious segmentation point found by an initial heuristic search and incorporates a novel rule-based fusion technique to combine three neural network based confidence values for verification of correct and incorrect segmentation points. The algorithm also incorporates a novel technique for extraction of characters between two segmentation points, which avoids the problems of vertically segmenting characters.

The remainder of the paper is broken down into 5 sections. Section 2 describes the proposed segmentation algorithm, Section 3 provides experimental results and Section 4 presents a detailed comparative analysis, and a conclusion is drawn in Section 5.

2. PROPOSED SEGMENTATION ALGORITHM

This section describes the proposed segmentation algorithm and sub-algorithms incorporated in it. An overview of the segmentation algorithm is shown in Figure 1 and a stepwise algorithm is given below:

Step 1: Detect Baseline	(Section 2.1)
Step 2: Over-segment Word	(Section 2.2)
Step 3: Extract Left and Center Characters	(Section 2.3)
Step 4: Evaluate and Fuse the Confidence Values	(Section 2.4)

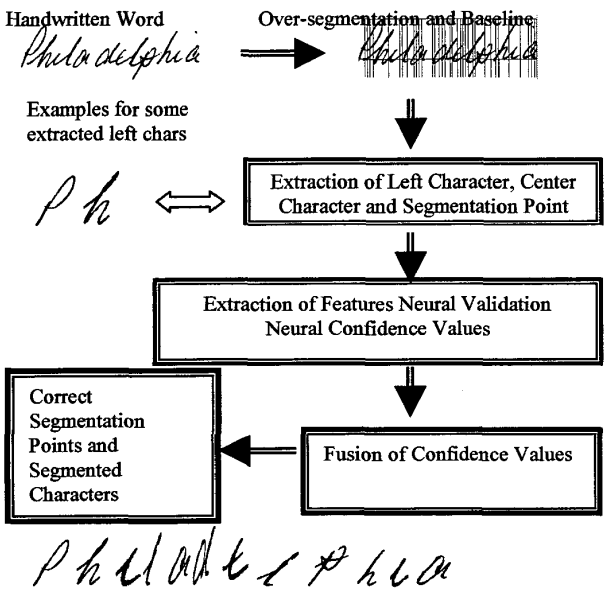

Figure 1: An Overview of the Segmentation Algorithm

2.1 Baseline Detection

The baseline detection algorithm incorporated in our research is very simple and is similar to those employed by other researchers in the area. It is briefly described below:

Step 1: Calculate the horizontal histogram row by row.

Step 2: Calculate the change between the histogram and save the values to an array.

Step 3: Find the maximum value of the histogram and its corresponding row.

Step 4: Start from the row that has the maximum histogram value, look for the biggest change in the histogram density or until the bottom of the image is reached. The row that has the biggest change in histogram density will be the baseline.

2.2 Over-segmentation

To assign Prospective Segmentation Points (PSPs) that could be validated through further processing, a heuristic, over-segmentation algorithm was developed [19,20]. Initially, a number of preliminary processing steps were required prior to segmentation point assignment. Following these steps, an algorithm was devised to propose segmentation points based on various handwriting features such as upper and lower word contours, holes, upper and lower contour minima, vertical density histograms and confidence assignment.

2.2.1 Preliminary Processing

The first step of processing required connected components in the image to be determined. These components represented isolated characters or sections of joined or cursive handwriting. The larger components would be split into smaller components in further steps. Conversely, small connected components were deemed as being "noise" in the word image and were hence removed to facilitate the segmentation process.

The next steps of processing required calculation of certain measurements that would assist in locating segmentation points: 1) Stroke width estimation, 2) Word height estimation and 3) Average character width estimation. These measurements were invaluable for serving as threshold values when assigning segmentation points (refer to Section 2.2.2).

2.2.2 Segmentation Point Assignment

Segmentation points in each word were assigned based on confidence values derived from examining various features located in the word image. Confidence values for PSPs were increased if the particular area under examination contained minima in the upper and lower contour (i.e. possible ligatures in cursive writing) and areas in the word that exhibited a low vertical pixel density. Finally, a PSP's confidence was decreased if it was located in an area of the word that contained "holes" (i.e. found in such letters as "a" and "o"). The PSPs bearing the highest confidences were retained for further examination i.e. redundant segmentation point removal, and equally distributing segmentation points throughout the word. Hence the PSPs generated where used as input to the next phase of the entire segmentation procedure. The reader may refer to [19] for a detailed description of the over-segmentation algorithm.

2.3 Left and Center Character Extraction Technique

As described above, our segmentation algorithm needs additional information about a given segmentation point such as left character, center character, etc, however the standard neural validation algorithm performs vertical segmentation which sometimes cuts a character in half or it takes part of another character which causes problems in further steps. An example of such a problem is shown below:

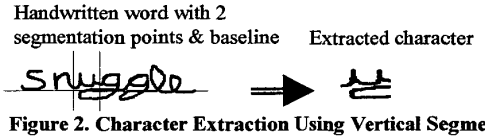

Figure 2. Character Extraction Using Vertical Segmentation

To overcome the above problem in our segmentation algorithm, we propose a novel character extraction technique. The main aim of our novel character extraction technique is to extract a character as follows:

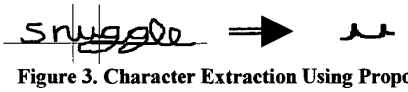

Figure 3. Character Extraction Using Proposed Algorithm

The algorithm is described below in detail:
The main characteristic of the character extraction algorithm is that it relies on getting all connected black pixels starting from a particular pixel between segmentation points one and two. Consequently, at the start of the algorithm the black lines at segmentation points one and two are disconnected so that when it is finding all connected black pixels, it does not go past these points. Compared to simple vertical segmentation extraction (Figure 2), this approach always extracts a cleaner character, especially when extracting from a slanted word (Figure 3). Due to the awkward slant and connective nature of some cursively written words, the algorithm may sometimes attempt to find connected black pixels beyond the regions marked by segmentation points one and two. To combat this, the algorithm detects the direction and stops finding connected black pixels if it is heading horizontally past these segmentation regions. Without this detection, whole words may be extracted at times because of the nature of the algorithm which searches for all connected black pixels. The following is a description of the character extraction algorithm.

Input:
Source word that we extract from WORD
Segmentation point 1 (x-coordinate) SEG_PT_1
Segmentation point 2 (x-coordinate) SEG_PT_2
Baseline of the source word (y-coordinate) BASELINE
Output:
Extracted character DESTINATION

Begin
Create an empty DESTINATION word of same size as WORD
Create a TEMP word that is a copy of WORD
Disconnect the line vertically closest to TEMP[BASELINE][SEG_PT_1]
Disconnect the line vertically closest to TEMP [BASELINE][SEG_PT_2]
Find the start pixel
START_PIXEL = black pixel closest to TEMP [BASELINE][SEG_PT_1]
where the START_PIXEL x-coordinate >= SEG_PT_1 and <= SEG_PT2
Starting at TEMP[START_PIXEL], find all connected black pixels and store in DESTINATION
Stop find all connected black pixels if current black pixel is just past segmentation point 1 or 2 and close to the baseline
Trim DESTINATION so there are no white spaces on left or right sides
Return DESTINATION (extracted character)
End

We have separately tested the above algorithm and it works very well. Some examples, are shown in Figure 4 below:

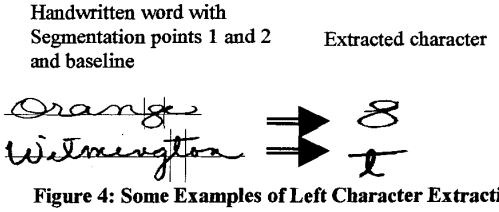

Figure 4: Some Examples of Left Character Extraction

2.4 Evaluation and Fusion of Confidence Values

The confidence values from three neural networks i.e. left character, center character and segmentation area were obtained and fused. The following rules were used for this purpose.
A Correct Segmentation Point (CSP) is found:
if $f_{SPV\_Ver}(ft1)>=0.5$ AND
$f_{LCC\_Ver}(ft2)$ is a high character confidence AND
$f_{CCC\_Ver}(ft3)$ is a high non-character confidence;
$f_{CSP}(ft1, ft2, ft3)=f_{SPV\_Ver}(ft1)+ f_{LCC\_Ver}(ft2)+ f_{CCC\_Ver}(ft3)$
where, f_{SPV_Ver} - Confidence value from the Segmentation Point Validation neural network.
f_{LCC_Ver}– Left Character Confidence (LCC) value from character neural network.
f_{CCC_Ver}– Centre Character Confidence (CCC) from character neural network (reject neuron output).
An Incorrect Segmentation Point is found:
if $f_{SPV\_Ver}(ft1)<0.5$ AND
$f_{LCC\_Ver}(ft2)$ is a high non-character confidence AND
$f_{CCC\_Ver}(ft3)$ is a high character confidence;
$f_{ISP}(ft1, ft2, ft3)=(1-f_{SPV\_Ver}(ft1))+f_{LCC\_Ver}(ft2)+ f_{CCC\_Ver}(ft3)$
where, $f_{SPV\_Ver}(features)$ - Confidence value from Segmentation Point Validation neural network .
$f_{LCC\_Ver}(features)$ – Left Character Confidence value from character neural network (reject neuron output).
$f_{CCC\_Ver}(features)$ – Centre Character Confidence value from character neural network (highest confidence from 26 character neuron outputs).
Finally, the outcome of fusion is decided by the following equation: f(confidence)=max(f(CSP), f(ISP))

3. EXPERIMENTAL RESULTS

3.1 Implementation and Database

The segmentation algorithm and most of the associated sub-algorithms/techniques were implemented in C++ on a UNIX platform. However, some of the techniques such as the character extraction algorithm, baseline detection, etc. were implemented in C++ on a Windows platform. The handwritten words from the CEDAR benchmark database [16] were employed to train the various neural networks used to assist in the segmentation process. For testing, 300 words were obtained from the "BD/cities" test set. A subset of 79 words was used for an analysis of segmentation performance.

3.2 Experimental Results

To test the accuracy of our novel segmentation algorithm described in this paper, two major experiments were conducted. The first experiment was conducted using heuristic over-segmentation followed by neural validation, vertical left character extraction and the fusion of confidences. The second experiment was conducted using the same techniques as the second experiment, except that the vertical character extraction technique was replaced by the left-character extraction technique described in Section 2.3. In all experiments described above, the over-segmentation technique provided a number of PSPs for each word in the test set. These PSPs were truthed earlier by a human operator i.e. separated into "correct" and "incorrect" segmentation points. The proposed segmentation technique was subsequently executed with and without the incorporation of the new character extraction technique. Each segmentation point that was given a high "correct" confidence after validation was "retained", while those segmentation points that were given a low "incorrect" confidence were "discarded". Following each of the aforementioned experiments, the results of the proposed segmentation algorithm were compared to the truth-values assigned by the human operator. The results are displayed in Table 1.

Table 1. Segmentation Results

Experiment #	Segmentation Technique	# of Correct Seg. points	# of Incorrect Seg. points
1	Neural Validation + Left Character	481/575 (83.65)	333/473 (70.40)
2	Neural Validation + Novel Left Character	488/575 (84.87%)	341/473 (72.09%)

The percentages have been calculated based on the number of "correct" segmentation points that have been "retained" as well as the number of "incorrect" segmentation points that have been "discarded".

4. ANALYSIS OF SEGMENTATION RESULTS

In this section we present a detailed analysis and comparison of results from Section 3. As shown in Table 1, based on correct and incorrect segmentation point analysis, we found that the segmentation algorithm with neural validation and the novel left character extraction technique performed reasonably well. As shown in Table 2 and Figure 5, during the analysis, we found the "over-segmentation" and "bad" errors were quite low at 7.08% and 10.86% respectively. It may be seen that a large number of "bad" segmentations were obtained (11.64 %) in Experiment 1, which was significantly decreased (improved) in Experiment 2. This may be partly attributed to the fact that the segmentation algorithm (Experiment 1) did not locate segmentation paths between characters, and therefore although the anchorage point was correct, the vertical segmentation would at times mistakenly cut adjacent character components. In Experiment 2, we have incorporated our character extraction technique, which does not use vertical segmentation. Instead it extracts the whole character using connected components without cutting characters partially or in half.

Table 2. Segmentation Errors

Experiment #	Over-Segmentation (%)	Missed (%)	Bad (%)	Bad + Correct Anchorage Points (%)
1	7.47	2.04	11.64	5.33
2	7.08	2.33	10.86	6.01

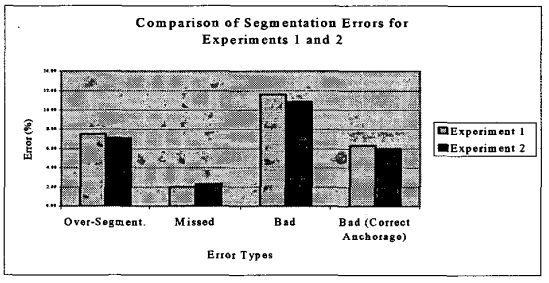

Figure 5. Comparison of Segmentation Errors

On comparing the results of the first experiment to those of the second, an improvement may be seen in the latter one. The over-segmentation error drops by 0.4%. However, at the same time, a slight increase for the number of missed segmentations also occurs (up 0.3%). This slight variation may, in some cases, be attributed to incorrect extraction of character areas for network identification. The change in error distribution may also be attributed to a lack of "weighting" when character confidences are fused. One final and quite positive observation may be made with respect to the "bad" segmentation error. It may be seen that it drops by over 3%. In comparison with other researchers in the field, the results obtained in this research are comparable. For the results presented here, the "missed" rate is within approximately 1% of those found in [17] and [18], although the over-segmentation rate is higher in our research (by approximately 5%). Finally, bad segmentation (not including errors attributed to incorrect segmentation paths) is comparable to those obtained in [17] and [18-19] respectively. It may be noted that it is quite difficult to compare the results obtained here with those in [17,18], as the authors used a different and much simpler data set.

5. CONCLUSIONS

We have presented a novel segmentation algorithm and tested it on real-world handwritten words taken from a benchmark database. As results and analysis shown in previous section, the proposed novel algorithm has done reasonably well. The results obtained with the algorithm using vertical segmentation were promising, however in some cases it incorrectly cuts adjacent characters. Therefore we have developed a novel character extraction technique and employed it to our segmentation algorithm. It can be seen in Figures 3 and 4, that the character extraction technique extracts touching and over-lapping characters accurately without any distortion. Overall, the recognition rate obtained by the algorithm incorporated with character extraction technique is 3-4% higher than the algorithm incorporated with vertical segmentation. The overall segmentation results obtained by the proposed algorithm are comparable to other existing techniques. Further improvements to the algorithm such as size independent feature extraction techniques and re-training of neural networks with extracted characters employing the novel technique proposed in this paper, are currently being investigated.

ACKNOWLEDGMENT

The author would like to thank ARC for supporting this research and two researchers John Zakos and Michael Blumenstein from CIRL at Griffith University for their help in conducting experiments.

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.