A Novel Feature Extraction Technique for the Recognition of Segmented Handwritten Characters

Author

Blumenstein, Michael, Verma, Brijesh, Basli, Hasan

Published

2003

Conference Title

Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR '03)

DOI

Copyright Statement

Downloaded from

Link to published version

http://ieeexplore.ieee.org/servlet/opac?punumber=8701

Griffith Research Online

https://research-repository.griffith.edu.au

A Novel Feature Extraction Technique for the Recognition of Segmented Handwritten Characters

M. Blumenstein, B. Verma and H. Basli

School of Information Technology, Griffith University-Gold Coast Campus, Australia
E-mail: {m.blumenstein, b.verma}@griffith.edu.au

Abstract

High accuracy character recognition techniques can provide useful information for segmentation-based handwritten word recognition systems. This research describes neural network-based techniques for segmented character recognition that may be applied to the segmentation and recognition components of an off-line handwritten word recognition system. Two neural architectures along with two different feature extraction techniques were investigated. A novel technique for character feature extraction is discussed and compared with others in the literature. Recognition results above 80% are reported using characters automatically segmented from the CEDAR benchmark database as well as standard CEDAR alphanumerics.

1. Introduction

The literature is replete with high accuracy recognition systems for separated handwritten numerals and characters [1],[2]. However, research into the recognition of characters extracted from cursive and touching handwriting has not had the same measure of success [3]-[5]. One of the main problems faced when dealing with segmented, handwritten character recognition is the ambiguity and illegibility of the characters. Although a difficult problem, the accurate recognition of segmented characters is important in the context of segmentation-based, word recognition [4]. In this research, various neural architectures for character recognition are investigated. Two feature extraction techniques are tested including a novel technique that extracts features based on the direction of line segments within a character image.

The remainder of this paper is broken down into 4 sections. Section 2 describes the neural-based character classifiers, Section 3 provides experimental results, a discussion of the results takes place in Section 4, and finally Section 5 presents conclusions and future research.

2. Segmented character recognition

Two feature extraction techniques were investigated in this research, the first of which is proposed here for the first time. Also, two neural classifiers were used for experimentation and are described in this section.

2.1. Character extraction

The process of extraction was only employed for those characters obtained from handwritten words. To summarise the character extraction process, our technique first proceeded to sequentially locate all non-cursive/printed character components through the use of character component analysis. Next, x-coordinates (vertical segmentations) for each connected character component (identified by a heuristic segmenter [6],[7]) were used to define the vertical boundaries of each character matrix.

The first character data set used for training and testing was extracted from words in the training and test directories (CITIES/BD) of the CEDAR CD-ROM [8]. This will be referred to as the CEDAR Automatically Segmented (CAS) data set. To obtain input vectors of uniform size, the extracted characters were either re-scaled, padded or alternatively local averaging was performed on the feature vector. The second data set was comprised of pre-segmented, Binary Alphanumeric Characters (BAC) from the CEDAR CD-ROM found in the BINANUMS/BD & BL directories.

2.2 Preprocessing

During initial processing, word and character images were converted from the standard CEDAR format to a .pbm format. In the case of words, the images were also thresholded and slant corrected [7]. For the proposed feature extraction technique (direction feature), further preprocessing was undertaken. The proposed technique sought to locate individual strokes or line segments in the

IEEE COMPUTER SOCIETY

character image to serve as features to the classification stage. To facilitate this, two types of preprocessing were investigated using the direction feature technique: 1) Thinning [9] and 2) Boundary extraction [10].

2.3. Direction feature

The first technique (direction feature) sought to simplify each character's boundary or thinned representation through identification of individual stroke or line segments in the image. Next, in order to provide a normalized input vector to the neural network classification schemes, the new character representation was broken down into a number of windows of equal size (zoning) whereby the number, length and types of lines present in each window was determined. The line segments that would be determined in each character image were categorised into four types: 1) Vertical lines, 2) Horizontal lines, 3) Right diagonal and 4) Left diagonal. Aside from these four line representations, the technique also located intersection points between each type of line.

To facilitate the extraction of direction features, the following steps were required to prepare the character pattern:

1. Starting point and intersection point location
2. Distinguish individual line segments
3. Labeling line segment information
4. Line type normalization

2.3.1 Starting point and intersection point location. To locate the starting point of the character, the first black pixel in the lower left hand side of the image is found. The choice of this starting point is based on the fact that in cursive English handwriting, many characters begin in the lower, left hand side. Subsequently, intersection points between line segments are marked (these locations are a component of the final feature vector). Intersection points are determined as being those foreground pixels that have more than two foreground pixel neighbours.

2.3.2 Distinguish individual line segments. As mentioned earlier, four types of line segments were to be distinguished as compromising each character pattern. Following the step described in Section 2.3.1, neighbouring pixels along the thinned pattern/character boundary were followed from the starting point to known intersection points. Upon arrival at each subsequent intersection, the algorithm conducted a search in a clockwise direction to determine the beginning and end of individual line segments. Hence, the commencement of a new line segment was located IF:

1. The previous direction was up-right or down-left AND the next direction is down-right or up-left OR

2. The previous direction is down-right or up-left AND the next direction is up-right or down-left OR
3. The direction of a line segment has been changed in more than three types of direction OR
4. The length of the previous direction type is greater than three pixels

The above rules were consulted for each set of pixels in the character pattern. The thresholds in rules 3 and 4 above were determined by manually inspecting a subset of the character database.

Firstly, it was noted that the object of "line type normalization" (Section 2.3.4.) was to normalize individual lines in a character pattern matching one of the four line types described earlier; specifically pixels which did not significantly deviate from one single direction type. Hence, it was posited that a line changing in more than three directions could not be called a single line.

Secondly, in rule 4 above, the threshold of three was chosen as it was found through visual inspection that a suitable "minimum line length" could be deemed as being one composed of at least four pixels all marked with the same direction value.

2.3.3 Labeling line segment information. Once an individual line segment is located, the black pixels along the length of this segment are coded with a direction number as follows: Vertical line segment - 2, Right diagonal line - 3, Horizontal line segment - 4 and Left diagonal line - 5. Figure 1 illustrates the process of marking individual line segments.

2.3.4 Line type normalization. Following the steps described in Sections 2.3.2 and 2.3.3, line segments were composed of marked direction pixels (Figure 1).
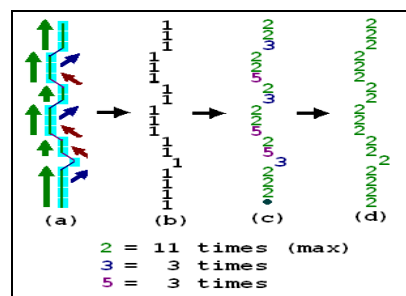
Figure 1: (a) Original line, (b) Line in binary file, (c) After distinguishing directions, (d) After direction normalization

As line segments were marked by following either the character boundary or thinned image pattern on a pixel-by-pixel basis, some spurious direction values may have been marked in any particular line segment due to the presence of anomalous pixels introduced through the thinning or boundary extraction process. Hence to "normalize" a particular line segment (discarding spurious

direction values), the number of values belonging to each direction type was tallied in a particular line. The direction value most frequently represented in a particular line segment was used to replace spurious direction values (Figure 1).

2.3.5 Formation of feature vectors through zoning. As neural classifiers require vectors of a uniform size for training, a methodology was developed for creating appropriate feature vectors. In the first step, the character pattern marked with direction information was zoned into windows of equal size (the window sizes were varied during experimentation). If the image matrix was not equally divisible, it was padded with extra background pixels along the length of its rows and columns. In the next step, direction information was extracted from each individual window. Specific information such as the line segment direction, length, intersection points, etc. were expressed as floating point values between –1 and 1.

The algorithm for extracting and storing line segment information first locates the starting point and any intersections in a particular window. It then proceeds to extract the number and lengths of line segments resulting in an input vector containing nine floating-point values. Each of the values comprising the input vector was defined as follows: **1.** The number of horizontal lines, **2.** The total length of horizontal lines, **3.** The number of right diagonal lines, **4.** The total length of right diagonal lines, **5.** The number of vertical lines, **6.** The total length of vertical lines, **7.** The number of left diagonal lines, **8.** The total length of left diagonal lines and **9.** The number of intersection points (Figure 2).

As an example, the first floating point value represents the number of horizontal lines in a particular window. During processing, the number starts from 1.0 to represent "no line" in the window. If the window contains a horizontal line, the input decreases by 0.2. The reason a value commencing at 1.0 and decreasing by 0.2 was chosen was mainly because in preliminary experiments, it was found that the average number of lines following a single direction in a particular window was 5. However in some cases, there were a small number of windows that contained more than 5 lines, and hence in these cases the input vector contained some negative values. Hence values that tallied the number of line types in a particular window were calculated as follows:

$$value = 1 - ((\text{number of lines}/10) \times 2) \qquad (1)$$

For each value that tallied the number of lines present in a particular window, a corresponding input value tallying the total length of the lines was also stored. To illustrate, the horizontal line length can be used as an example. The number starts at 0 to represent "no horizontal lines" in a particular window. If a window has a horizontal line, the input will increase by the length of the

line divided by the maximum window length or window height, (depending on which one is the largest) multiplied by two. The reason this formula is used, is because it is assumed that the maximum length of one single line type is two times the largest window size. As an example, if the line length is 7 pixels and the window size is 10 pixels by 13 pixels, then the line length will be $7 / (13 \times 2) = 0.269$.

$$length = \frac{\text{number of pixels in a particular direction}}{(\text{window height or width}) \times 2} \qquad (2)$$

The operations discussed above for the encoding of horizontal line information must be performed for the remainder of directions. The last input vector value represents the number of intersection points in the character. It is calculated in the same manner as for the number of lines present. Figure 2 illustrates the process of input vector creation.
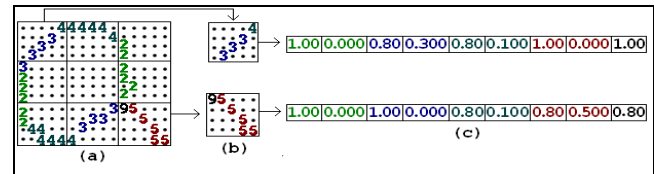
Figure 2: (a) Processed image, (b) Zoned windows, (c) Input vector components

2.4. Transition feature

The second feature extraction technique investigated in this research was based on the calculation and location of transition features from background to foreground pixels in the vertical and horizontal directions. A number of researchers have proposed feature extraction techniques based on transition information, examples may be found here [4],[11]. This technique operates on the raw character image and does not require resizing. The transition feature technique was investigated and compared to the proposed feature extraction technique. It was selected for two main reasons. Firstly, it is a technique that has performed well in the literature and secondly it was used to determine character confidences in our initial segmentation system.

2.5 Configuration of the neural classifiers

The neural classifiers chosen for the task of character recognition were Back-Propagation (BP) and Radial Basis Function (RBF) networks. For experimentation purposes, the architectures were modified varying the number of inputs, outputs, hidden units, hidden layers and the various learning terms.

The number of inputs to each network was associated with the size of the feature vector for each image. The most successful vector configurations were of size 100 for the transition feature and 81 for the direction feature. With

the CAS data set, two separate neural networks were trained for lowercase and uppercase characters. The number of outputs considered for this data set was 27 (characters a-z and 1 reject neuron for non-character patterns). The non-character category was specifically used to identify character patterns that were smaller than approximately half a regular character and those character components that had not been correctly split i.e. multiple characters. For the BAC data set, a single neural network architecture was considered with 36 outputs similar to that proposed by Singh and Hewitt [12].

2.6. Preparation of training and testing data

For neural network training it was necessary to include samples for each type of character (a-z, A-Z). In the case of the CAS data set, training and test files needed to be manually prepared, however character matrix patterns were determined based on the output of our heuristic segmenter. Each extracted character was viewed by a human operator and was labelled manually as belonging to a particular character class.

3. Experimental results

The results in this research are displayed in tabular form for each set of experiments. Table 1 presents top results using the CAS dataset, the BP algorithm and the two feature extraction techniques described in Section 2. Separate experiments were conducted for lower case and upper case character patterns. A total of 18655 lower case and 7175 upper case character patterns were generated for training. A further 2240 lower case and 939 upper case patterns were used for testing.

Table 2 presents results once again using the CAS dataset, however in this case the RBF network was used. Finally, Table 3 & Table 4 present results for the BAC dataset using the BP and RBF networks for non-resized patterns. The BAC dataset contained a total of 19145 characters for training and 2183 characters for testing. Resized patterns were excluded from these experiments on the basis that in preliminary experiments with the CAS dataset, the results suggested that on average the non-resized dataset outperformed the resized one.

Table 1. Top character recognition rates using the BP network and the CAS dataset

	Recognition Rate [%]	
	Lowercase	Uppercase
Direction (resized thinned)	**69.78**	78.70
Direction (resized boundary)	69.73	77.32
Direction (non-resized thinned)	69.02	**80.62**
Direction (non-resized boundary)	67.19	79.98
Transition	67.81	79.23

Table 2. Top character recognition rates using the RBF network and the CAS dataset

	Recognition Rate [%]	
	Lowercase	Uppercase
Direction (resized thinned)	69.96	77.21
Direction (resized boundary)	**70.63**	75.93
Direction (non-resized thinned)	70.54	**79.98**
Direction (non-resized boundary)	69.51	79.23
Transition	70.31	79.13

Table 3. Top character recognition rates using the BP network and the BAC dataset

	Recognition Rate [%]
Direction (non-resized thinned)	83.10
Direction (non-resized boundary)	**83.65**
Transition	82.82

Table 4. Top character recognition rates using the RBF network and the BAC dataset

	Recognition Rate [%]
Direction (non-resized thinned)	80.81
Direction (non-resized boundary)	80.99
Transition	**85.48**

4. Discussion of results

4.1 General discussion

As may be seen from the results in Table 1 & 2, the use of non-resized patterns resulted in a comparable or slightly higher recognition rate. Whereas the difference in character classification rate when features were extracted from thinned character images and the character boundary was negligible.

4.2 BP and RBF networks

Upon comparison of results using the RBF and BP networks, the top recognition rates in each case are of a very similar standard. Specifically, the difference in recognition rates for the CAS dataset is negligible. Although the recognition rate was not notably higher, training time was significantly reduced when the RBF network was employed. For the BAC dataset the highest recognition rate was obtained using the RBF network. On average however, the BP network outperforms RBF.

4.3 Direction vs Transition feature

Across all experiments that were performed, it was found that the direction feature outperformed the transition feature in all but one case. However, whilst using the BP network, small differences in recognition rate may sometimes be attributed to variations in the starting conditions of the network. Hence, to confirm the veracity

IEEE
COMPUTER
SOCIETY

of the recognition rates attained, extra experiments, using the BP network configuration that provided the top result for direction and transition features, were conducted. For each feature type, a total of six experiments were conducted and the average recognition rate was calculated. In each case the direction feature outperformed the transition feature technique by 2% for lowercase characters and almost 7% for uppercase characters. As mentioned above, the transition feature outperformed the direction feature in one experiment. Whilst testing the BAC dataset and using the RBF network, the top transition feature result outperformed the top direction feature result by 4.5%. It must however be stressed that the results are still comparable and that this was an isolated occurrence as confirmed in Tables 1-3.

4.4 Character recognition results

It is always a difficult task to compare results for handwritten character recognition with other researchers in the literature. The main problems that arise are differences in experimental methodology, experimental settings and the handwriting database used. The comparisons presented below have been chosen for two main reasons. The handwriting database (CEDAR) used by the researchers is identical to the one used in this research and the results are some of the most recent in the literature. Yamada and Nakano [3] presented a handwritten word recognition system, which was trained on segmented characters from the CEDAR benchmark database. They recorded recognition rates of 67.8% and 75.7% for the recognition of characters where upper case letters and lower case letters were distinguished and not distinguished respectively. Therefore, if the top lower case (70.63%) and upper case (80.62%) character recognition scores in this research are averaged, a recognition accuracy of 75.63% is obtained, which compares well with their results. Kimura *et al.* [5] used neural and statistical classifiers to recognise segmented CEDAR characters. For case sensitive experiments, their neural classifier produced an accuracy of 73.25%, which was comparable to the lower case and upper case average of 75.63%. Singh and Hewitt [12] employed the modified Hough Transform on characters from the CEDAR. They obtained a recognition rate of 67.3%, our best result using their network configuration (83.65%) compares favourably with their top recognition rate.

5. Conclusions and future research

This paper presented a new feature extraction technique (direction feature) for the recognition of segmented handwritten characters. The direction-based feature extraction technique was compared to another popular technique in the literature using two neural classification schemes. In general, it outperformed the transition feature technique and was comparable to other techniques in the literature. In future research a number of considerations will be addressed including an improved preprocessing methodology, a more automated approach to character image generation, an investigation of a wider variety of global and local features and finally integration into an off-line handwritten word recognition system.

6. References

[1] C. Y. Suen, and 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-B. Cho, "Neural-Network Classifiers for Recognizing Totally Unconstrained Handwritten Numerals", *IEEE Trans. on Neural Networks*, Vol. 8, 1997, pp. 43-53.

[3] H. Yamada and Y. Nakano, "Cursive Handwritten Word Recognition Using Multiple Segmentation Determined by Contour Analysis", *IEICE Transactions on Information and Systems*, Vol. E79-D, 1996, pp. 464-470.

[4] P. D. Gader, M. Mohamed and J-H. Chiang, "Handwritten Word Recognition with Character and Inter-Character Neural Networks", *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, Vol. 27, 1997, pp. 158-164.

[5] F. Kimura, N. Kayahara, Y. Miyake and M. Shridhar, "Machine and Human Recognition of Segmented Characters from Handwritten Words", *4th International Conference on Document Analysis and Recognition (ICDAR '97)*, Ulm, Germany, 1997, pp. 866-869.

[6] M. Blumenstein and B. Verma, "A New Segmentation Algorithm for Handwritten Word Recognition", *Proceedings of the International Joint Conference on Neural Networks, (IJCNN '99)*, Washington D.C., 1999, pp. 2893-2898.

[7] M. Blumenstein and B. Verma, 1999, "Neural Solutions for the Segmentation and Recognition of Difficult Words from a Benchmark Database", *Proceedings of the Fifth International Conference on Document Analysis and Recognition, (ICDAR '99)*, Bangalore, India, pp. 281-284.

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

[9] T. Y. Zhang and C. Y. Suen, 1984, "A Fast Parallel Algorithm for Thinning Digital Patterns", *Communications of the ACM*, Vol. 27, pp. 236-239.

[10] Parker, J. R., *Practical Computer Vision using C*, John Wiley and Sons, New York, NY, 1994.

[11] M. Shridhar and A. Badreldin, "High Accuracy Character Recognition using Fourier and Topological Descriptors", *Pattern Recognition*, Vol. 17, 1984, pp. 515-524.

[12] S. Singh and M. Hewitt, "Cursive Digit and Character Recognition on Cedar Database", *International Conference on Pattern Recognition, (ICPR 2000)*, Barcelona, Spain, 2000, pp. 569-572.