

A Novel Multiple Experts and Fusion Based Segmentation Algorithm for Cursive Handwriting Recognition

Hong Lee and Brijesh Verma

Abstract — This paper presents a novel segmentation algorithm for offline cursive handwriting recognition. An over-segmentation algorithm is introduced to dissect the words from handwritten text based on the pixel density between upper and lower baselines. Each segment from the over-segmentation is passed to a multiple expert-based validation process. First expert compares the total foreground pixel of the segmentation point to a threshold value. The threshold is set and calculated before the segmentation by scanning the stroke components in the word. Second expert checks for closed areas such as holes. Third expert validates segmentation points using a neural voting approach which is trained on segmented characters before validation process starts. Final expert is based on oversized segment analysis to detect possible missed segmentation points. The proposed algorithm has been implemented and the experiments on cursive handwritten text have been conducted. The results of the experiments are very promising and the overall performance of the algorithm is more effective than the other existing segmentation algorithms.

I. INTRODUCTION

THE importance and need of handwriting recognition has been arising in many real world applications such as postal address recognition, bank cheques processing, forms processing, conversion of field notes and historical manuscripts. Despite intensive research for more than four decades, off-line cursive handwriting recognition still remains an open problem [1]-[4]. The general off-line handwriting recognition system [5], [6] is shown below in Fig. 1.

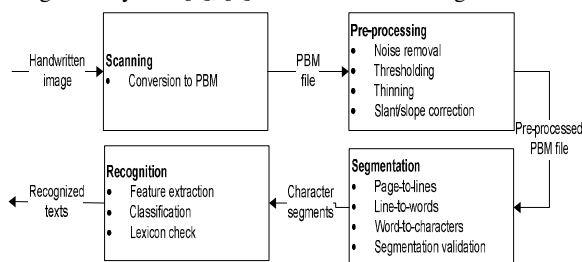


Fig. 1. Overview of off-line handwriting recognition system

Segmentation of cursive words into characters is one of the most difficult processes in handwriting recognition and it is also defined as one of the most important processes because it directly affects the result of recognition process [7]-[10].

Currently available segmentation techniques are dissection techniques [11], recognition-based segmentation [12], [13], over-segmentation [14]-[16], and holistic approaches. As the segmentation constraints, the main factors are the non-separability of characters, the diversity of character patterns, ambiguity and illegibility of characters, and the overlapping nature of many characters in a word [17]. The holistic strategy avoids segmentation process, but it is not practical in a large lexicon environment [18].

The rest of this paper is organized into four sections. Section II describes the proposed segmentation algorithm in detail. Section III presents the experimental results. An analysis of experimental results and a comparison are presented in Section IV. Finally, Section V concludes the paper.

II. PROPOSED SEGMENTATION ALGORITHM

The overview of the proposed segmentation methodology is shown below in Fig. 2.

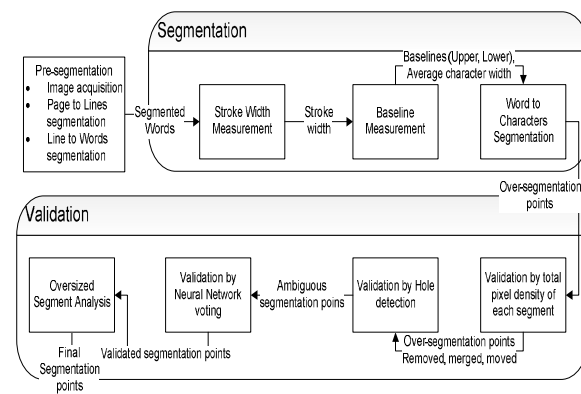


Fig. 2. Proposed segmentation methodology

A. Pre-Segmentation Process

The pre-segmentation module is intended for preparation of the handwritten image for the word-to-characters segmentation process. Starting from the image acquisition process by scanning the natural handwritten text image, page-to-lines and line-to-words segmentation processes are also performed in this step. At the end of this process, word image tokens are produced and delivered to segmentation process.

Manuscript received December 15, 2007.

Lee, H. is with Central Queensland University, Australia (phone: +61 7 4150 7052; e-mail: h.lee1@cqu.edu.au).

Verma, B. is with Central Queensland University, Australia (phone: +61 7 4930 9058; e-mail: b.verma@cqu.edu.au).

B. Segmentation Process

The intention of this module is to dissect the text words into characters, and the main tasks of this section are stroke width measurement, baseline calculation, and over-segmentation.

1) Stroke Width Measurement

In the proposed system, the stroke width is measured on word level to reflect the variation between words. To prevent the over-measurement of the stroke width, the maximum boundary has been set to a value of $(\text{WordImageHeight} \div 4)$ before scanning. Through the horizontal and vertical scan of the image for the transition distances of foregrounds, the most occurring transition distance becomes the stroke width of the word image.

2) Baseline Calculation

Calculating the baselines by the horizontal pixel density histogram is one of the favourite methods among researchers. However, it has many problems so in the proposed system, a novel approach has been used to find more accurate baselines. In the proposed strategy, the upper baseline candidates are nominated by measuring the distance from the upper-most pixel to the first foreground pixel. Secondly, the number of vertical transitions is measured to exclude the extensive horizontal line letters like 'T', 'L', etc. After the calculation of the number of transitions and the distances of every single column, a search algorithm for best upper baseline is applied. Firstly, for each row a temporary upper and lower boundary is set with the row in the middle and the temporary boundaries height is set to the same as the stroke width. Finally, summation of the occurrence of the distances to the first foreground pixel decides the upper baseline. Likewise, the lower baseline is measured. However, this method is effective on words containing letters with holes or partially closed area, such as 'c', 'e', etc. Fig. 3 compares the results of baselines' calculation by horizontal histogram and by the proposed approach, and indicates the latter produces better outcomes.

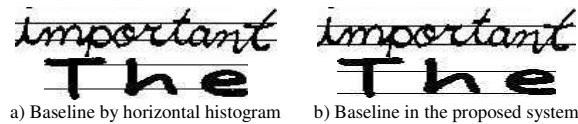


Fig. 3. Baseline comparison

3) Over-Segmentation between Baselines

Before over-segmenting, there is an important pre-task to be conducted, which is to decide a threshold. The vertical pixel density between upper and lower baselines is compared to the threshold, and the decision is made whether the appointed points are appropriate as candidate segmentation points. Finally, a continuous region of the candidate segmentation points is dissected into smaller sizes having the same width as the stroke width to prevent under-segmentation. However, this algorithm fails on touching points bigger than $(\text{StrokeWidth} \times 2)$. Fig. 4 describes the algorithm of the proposed over-segmentation. Fig. 5 displays the different results on different segmentation criteria, and the

segmentation criteria in the example b) produces better results because it prevents under-segmentation problems.

C. Multiple Expert-Based Validation Process

This section describes the validation processes used in the proposed approach.

1) Segment Total Foreground Pixel Comparison (STFPC)

Assuming that a legal character contains the minimum amount of foreground pixels equal to a criterion, $(\text{StrokeWidth} \times (\text{LowerBaseline} - \text{UpperBaseline}))$, each segment from neighbouring segmentation points is inspected. If the segments are not conformed to the criteria, its segmentation points are merged and moved to a prospective new segmentation point within the segment. This algorithm scans through all segmentation points recursively from left end to the right end. In Fig. 6 and Fig. 7, the algorithm and the result of the validation rule are shown.

```

SET UpperBaseline = CalculateUpperBaseline()
SET LowerBaseline = CalculateLowerBaseline()
SET VerticalPixelDensity = CalculatePixelDensity(UpperBaseline, LowerBaseline)
SET StrokeWidth = CalculateStrokeWidth()
SET SegmentCriteria = StrokeWidth + StrokeWidth ÷ 2
SET SegmentPoints = CreateSegmentationPointsArray()
SET Index = 0
FOR EACH VerticalPixelDensity
  IF VerticalPixelDensity[Index] < SegmentCriteria
    SegmentPoints[Index] = VALID
  ELSE
    SegmentPoints[Index] = INVALID
END FOR
WHILE (Continuous = CheckContinuousSegmentationPoints())
  Dissect Continuous into sizeof(StrokeWidth)
END WHILE

```

Fig. 4. Over-segmentation algorithm

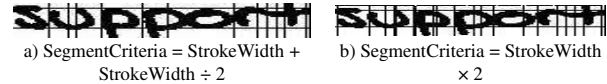


Fig. 5. Over-segmentation results depending on SegmentCriteria: a) Fails to segment between 'p' and 'o', b) 'p' and 'o' are correctly segmented.

```

SET ValidateCriteria = StrokeWidth × (LowerBaseline - UpperBaseline)
FOR EACH Segment
  SET TotalPixelDensity = GetTotalPixelDensityOf(Segment)
  IF TotalPixelDensity < ValidateCriteria
    RemoveOrMergeOrFindNewSegmentationPointsOf(Segment)
END FOR

```

Fig. 6. STFPC algorithm

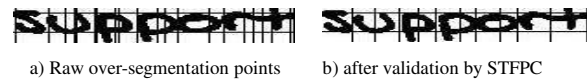


Fig. 7. Validation results before and after applying STFPC

2) Hole Detection

Hole detection algorithm is applied to every single over-segmentation point, and the detected points are immediately removed from the candidate segmentation points. As shown in Fig. 8, the segmentation points on the characters, 'P's and 'O' have been removed by the algorithm.

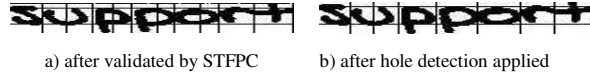


Fig. 8. Hole detection algorithm has been applied to the filtered points by STFPC

3) Recursive Validation by Neural Network Voting

In the recursive neural voting validation process, each segmentation point is tested and validated from left to right order. Invalid segmentation points are removed immediately and reset the segmentation point status. Inside the neural voting system (Fig. 9), the left, right and joined segment checkers get a left, right, joined segment region (shown in Fig. 10) from the testing segmentation point. The left and right checkers throws a positive vote on positive result. On the other hand, the joined checker throws negative vote on positive result. If the result of the left segment checker and the right segment checker is positive, then the lexicon checker looks up the sequence of the left and right characters in the dictionary.

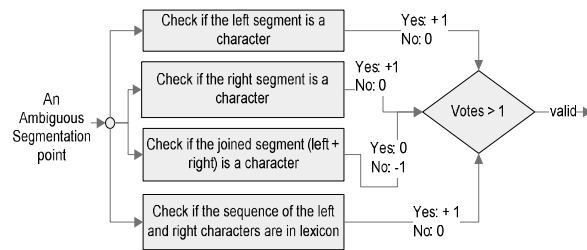


Fig. 9. Internal architecture of the neural network voting system

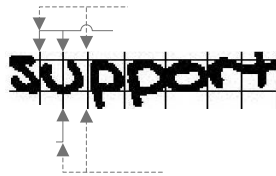


Fig. 10. Segment regions based on testing segmentation point

a) Neural Network Classifier

The engine behind all the checkers in the voting system is a neural network classifier trained on foreground transition features. Initially, it is trained on pre-segmented characters

with rubbish characters and tested on a training set and a testing set to find the best parameters such as hidden units, etc. It takes 100 transition features [19] of a character as an input and outputs one of 53 classes (52 classes for lower and upper cases, and 1 for rubbish characters). The overall architecture of the neural network classifier is described in Fig. 11. Fig. 12 shows a sample output of neural classification process.

4) Oversized Segment Analysis

The oversized segment analysis is adapted to detect any possible missed segmentation points from oversized segments. In order to find whether a segment is oversized or not, a criteria needs to be set. In the proposed approach, a maximum segment height from the testing word is set to the criteria. On detection of an oversized segment, a middle point of the segment is set to a new segmentation point. Fig. 13 describes an example of this process

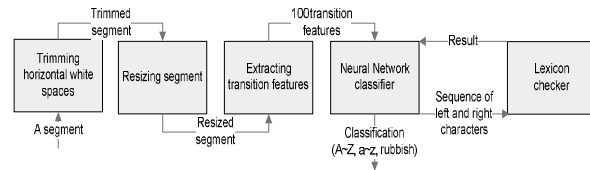


Fig. 11. Neural network classifier architecture

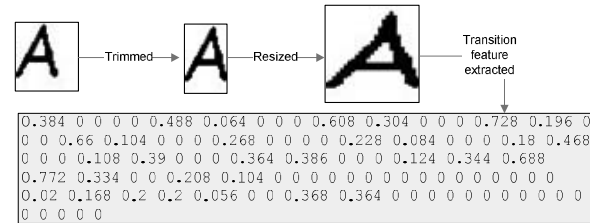


Fig. 12. Segment trimming, resizing and feature extraction

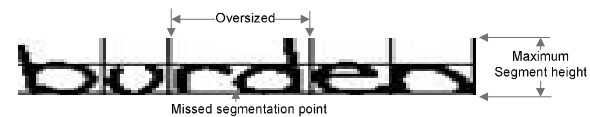


Fig. 13. Oversized segment analysis

III. EXPERIMENTAL RESULTS

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

A. Implementation

In the proposed system, all algorithms have been implemented in C++ programming language using object oriented principles.

B. Database Preparation

In the current experiments, the character database has been populated from segmented cursive handwritten text. Firstly, total of 744 characters are pre-segmented and separated into two sets on 5 to 1 ratio. The first dataset is intended to train the neural classifier, and it contains 620 characters, 520 for 10 sets of alphabet letters (A ~ Z, a ~ z), plus 100 for rubbish segments. The second database is for testing purpose during the neural network training. It consists of 124 elements, 104 for 2 sets of alphabet letters and 20 for rubbish. Finally, two other databases of real world handwritten text with 40 words (simple handwriting) and 218 words (cursive handwriting) are prepared to measure the segmentation performance.

C. Neural Network Training

In the proposed approach, an Error Back-Propagation (EBP) algorithm is employed. In Table 1, the recognition accuracies are recorded to find the best configurations of hidden units, iterations, and accuracies.

D. Segmentation Results

As described in [8], the numbers of over-segmentation, under-segmentation, and bad-segmentation points are counted by manual inspection. The over-segmentation is defined as a character is segmented into more than three segments. Under-segmentation points are the missing segmentation points between two neighbouring characters. Finally, the bad-segmentation is the rest of inappropriate cuts that do not belong to under-segmentation and over-segmentation, and do not separate two characters correctly.

The results shown below in Table 2 are the statistics of the under-segmentation points based on different threshold values of segmenting criteria to measure over-segmentation performance. For STFPC performance, the under-segmentation points are calculated and shown in Table 3. For each configuration of EBPNN, the recursive neural voting validation was performed and the results are presented in Table 4. The performance of the oversized segment analysis is measured by calculating the differences of over, under, and bad segmentation ratio from the result of the neural voting validation. The performance results are shown in Table 5. The final segmentation performance result using 218 words is shown in Table 6.

TABLE 1. NEURAL NETWORK CLASSIFICATION ACCURACY

Algorithm	Train set size	Hidden units	Iterations	Classification Accuracy (%)	
				Train set	Test set
EBP	620	50	1000	91.6	68.5
	620	78	1500	99.3	83.8
	620	100	1000	95.4	62.0

IV. ANALYSIS AND DISCUSSION

As shown in Table 2, the performance of the over-segmenter was very successful, showing a zero under-segmentation ratio on the segmentation criteria of (StrokeWidth \times 2). As the results shown in Table 3, the total segmentation points from

the over-segmentation were reduced into half. However, the validation rule produced three under-segmentation points. Potentially, the under-segmentation points could be cut down to less or zero, by using different comparison threshold of the total foreground pixel.

As shown in Table 4, the classification accuracies for characters produced by neural networks vary, and the maximum was around 83.8%. Despite the increase of ANN accuracy, the under-segmentation error rate remains unchanged (4.71%). However, over and bad segmentation error rates decrease gradually. Therefore the neural voting validation results are progressive when the classification accuracy of the neural network classifier increases as shown in Fig. 14. Table 5 shows that the oversized segment analysis process was very effective and it has reduce the under segmentation error by 7.06%. As side effects of the algorithm application, however, error rates of over and bad segmentation have been increased by 1.18% and 1.76% respectively. Therefore, overall improvement has been made by the oversized segment analysis process by average 1.37%. As shown in Table 6, over segmentation error is 1.07% in final segmentation. Whereas, under and bad segmentation errors are relatively higher than over segmentation ones. Overall, the proposed segmentation technique produced average segmentation error of 5.25%.

TABLE 2. OVER-SEGMENTATION PERFORMANCE

Over-segmenting criteria	Words	Under segmentation points	
		Total	Per Word
VerticalPixelDensity < StrokeWidth	40	23	0.5
VerticalPixelDensity < (StrokeWidth \times 1.5)		13	0.3
VerticalPixelDensity < (StrokeWidth \times 2)		0	0

TABLE 3. PERFORMANCE OF THE SEGMENT TOTAL FOREGROUND PIXEL COMPARISON (STFPC) VALIDATION PROCESS

	Words	Number of segmentation points		Under segmentation ratio per word
		Total	Under	
Raw over-segmentation	40	599	0	0
After the STFPC validation		274	3	0.075

TABLE 4. NEURAL VOTING VALIDATION PERFORMANCE ON DIFFERENT ANN CONFIGURATIONS (40 WORDS)

Recognition Accuracy (%)	Segmentation rate (%)			
	Under	Over	Bad	Average
41.1	4.71	1.18	13.53	6.47
62.9	4.71	1.18	12.94	6.27
83.8	4.71	1.76	11.76	6.08

TABLE 5. OVERSIZED SEGMENT ANALYSIS PERFORMANCE RESULTS (40 WORDS)

Recognition Accuracy (%)	Segmentation error difference (%)			
	Under	Over	Bad	Average
83.8	-7.06	1.18	1.76	-1.37

TABLE 6. FINAL SEGMENTATION PERFORMANCE RESULTS (218 WORDS)

Recognition Accuracy (%)	Segmentation rate (%)			
	Under	Over	Bad	Average
83.8	5.79	1.07	8.37	5.25

A. Comparative Analysis

A comparison of results with other approaches and algorithms in the literature is very difficult because many authors do not list the segmentation results in their papers. We have compared our algorithm with two other algorithms published in the literature and a comparative analysis is provided to give a relative look of the effectiveness of the proposed algorithm. In the proposed approach, the average incorrect segmentation rate is 5.25% which is slightly lower than the ones in [17], [20]. Especially the over segmentation error has been remarkably improved, so there is an error rate of only 1.07% for the category, which is much lower figures than the ones in the literature. However, the under segmentation ratio obtained by the proposed algorithm is fairly higher than the algorithms in the literature. For the bad segmentation error, the proposed technique produces slightly less segmentation errors than the one in [20], but shows fairly improved performance than [17]. As shown in Table 7 below, the proposed algorithm's overall segmentation performance is better than the existing algorithms [17], [20].

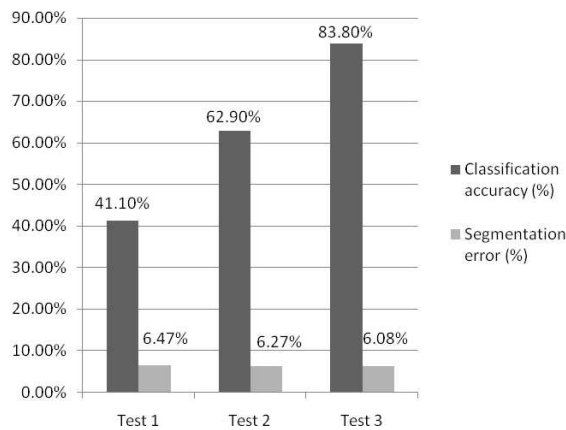


Fig. 14. Average incorrect segmentation rate on different neural network classification accuracy

TABLE 7. FINAL SEGMENTATION PERFORMANCE COMPARISON

	Segmentation rate (%)			
	Over	Under	Bad	Average
[17]	7.4	2.0	11.6	7.0
[20]	10.0	0.2	8.7	6.3
Proposed system	1.07	5.79	8.37	5.25

V. CONCLUSIONS AND FUTURE RESEARCH

In this paper, a novel segmentation paradigm for off-line handwritten text recognition has been proposed and investigated. The segmentation paradigm contains a baseline pixel-based over-segmenter, hole detection, segment

foreground pixel comparison and a neural voting based validation. Also, oversized segment analysis is performed before producing final segmentation points. The new segmentation paradigm has been tested on cursive handwritten text. The proposed segmentation approach produced lowest errors in comparison to existing approaches.

As a future research, optimizing the configurations of the neural classifier to increase the efficiency and accuracy of the validation processes is very important. Along with the improvements of neural network classifier, the internal architecture and fusion in neural voting system should be improved. The adaptation of slant and slope correction algorithms into the pre-segmentation tasks may achieve better segmentation accuracy so they need to be tested in the future research. In addition, the proposed approach should be tested on a larger handwriting database. Finally, measurements of word classification accuracy based on the segmentation results are recommended to confirm the effectiveness of the segmentation algorithm.

REFERENCES

- [1] L. M. Lorigo, and V. Govindaraju, "Offline Arabic handwriting recognition: a survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 5, pp. 712-724, 2006.
- [2] R. Plamondon and S. N. Srihari, "On-line and off-line handwriting recognition: a comprehensive survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 63-84, 2000.
- [3] F. Camastra, "A SVM-based cursive character recognizer," *Pattern Recognition*, vol. 40, no. 12, pp. 3721-3727, 2007.
- [4] P. Zhang, T. D. Bui and C. Y. Suen, "A novel cascade ensemble classifier system with a high recognition performance on handwritten digits," *Pattern Recognition*, vol. 40, no. 12, pp. 3415-3429, 2007.
- [5] A. Vinciarelli, "A survey on off-line cursive word recognition," *Pattern Recognition*, vol. 35, no. 7, pp. 1433-1446, 2002.
- [6] N. Arica and F. T. Yarman-Vural, "An overview of character recognition focused on off-line handwriting," *IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews*, vol. 31, no. 2, pp. 216-233, 2001.
- [7] X. Xiao and G. Leedham, "Knowledge-based cursive script segmentation," *Pattern Recognition Letters*, vol. 21, no. 10, pp. 945-954, 2000.
- [8] B. Yanikoglu and P. A. Sandon, "Segmentation of off-line cursive handwriting using linear programming," *Pattern Recognition*, vol. 31, no. 12, pp. 1825-1833, 1998.
- [9] R. G. Casey and E. Lecolinet, "A survey of methods and strategies in character segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 18, no. 7, pp. 690-706, 1996.
- [10] A. Elnagar and R. Alhajj, "Segmentation of connected handwritten numeral strings," *Pattern Recognition*, vol. 36, no. 3, pp. 625 - 635, 2003.
- [11] N. Arica and F. T. Yarman-Vural, "Optical character recognition for cursive handwriting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 6, pp 801 - 813, 2002.
- [12] R. Nopsumwanchai, A. Biem and W. F. Clocksin, "Maximization of mutual information for offline Thai handwriting recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 8, pp. 1347 - 1351, 2006.
- [13] L. Prevost, L. Oudot, A. Moises, C. Michel-Sendis and M. Milgram, "Hybrid generative/discriminative classifier for unconstrained character recognition," *Pattern Recognition Letters*, vol. 26, no. 12, pp. 1840-1848, 2005.
- [14] H. Lee, "Segmentation of cursive handwritten text," School of Comp. Sci. Central Queensland University, QLD, Australia, 2007.
- [15] P. Gader, B. Verma and W. Chen, "Fusion of multiple handwritten word recognition techniques," *Pattern Recognition Letters*, vol. 22, no. 9, pp. 991-998, 2001.

- [16] B. Verma, and H. Lee, "A segmentation based adaptive approach for cursive handwritten text recognition", IEEE International Joint Conference on Neural Networks, IJCNN'07, USA, 2007.
- [17] M. Blumenstein and B. Verma, "Analysis of segmentation performance on the CEDAR benchmark database," *Proceedings of the Sixth International Conference on Document Analysis and Recognition (ICDAR)*, Seattle, USA, pp. 1142-1146, 2001.
- [18] C. L. Liu, K. Nakashima, H. Sako and H. Fujisawa, "Handwritten digit recognition: benchmarking of state-of-the-art techniques," *Pattern Recognition*, vol. 36, no. 10, pp. 2271 – 2285, 2003.
- [19] X. Y. Liu, B. Verma and M. Blumenstein, "An investigation of the modified direction feature for cursive character recognition," *Pattern Recognition*, vol. 40, no. 2, pp. 376 – 388, 2007.
- [20] B. Verma, "A contour code feature based segmentation for handwriting recognition," *Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR)*, pp. 1203-1207, 2003.