Binary Segmentation with Neural Validation for Cursive Handwriting Recognition

Hong Lee and Brijesh Verma

Abstract —**Over-Segmentation and Validation (OSV) is a well anticipated segmentation strategy in cursive off-line handwriting recognition. Over-Segmentation is a means of locating all possible character boundaries, and the excessive segmentation points called over-segmentation points. Validation is a process to check and validate the segmentation points whether or not they are correct character boundaries by commonly employing an intelligent classifier trained with knowledge of characters. The existing OSV algorithms use ordered validation which means that the incorrect segmentation points might account for the validity of the next segmentation point. The ordered validation creates problems such as chain-failure. This paper presents a novel Binary Segmentation with Neural Validation (BSNV) to reduce the chain-failure. BSNV contains modules of over-segmentation and validation but the main distinctive feature of BSNV is an un-ordered segmentation strategy. The proposed algorithm has been evaluated on CEDAR benchmark database and the results of the experiments are promising.**

I. Introduction

Off-line Cursive Handwriting Recognition (OCHR) is to automate the conversion process of a handwritten input image into a meaningful character representation in a given language. The overall framework of the OCHR involves a number of sequential processing modules such as pre-processing, segmentation and recognition. The pre-processing step mainly focuses on altering input images to be fitted into the standards for the following processing modules requires. The segmentation is to find the character boundaries and is one of the most difficult and important processes in OCHR because the recognition heavily relies on the outcomes of the segmentation process. Finally, the recognition is to classify the segmented primitives into a character representation [1-7].

There are various factors that make the segmentation process problematic. Researchers found that the handwritten characters are quite often connected, overlapping or broken. Also, the nature of the handwriting inherits non-uniformity of the written elements. That's why some researchers prefer segmentation-free approaches in OCHR, called holistic approach. In the holistic strategy, the whole word is attempted to be recognized rather than individuals. Benouareth et al. [8] used word frames features, and Vinciarelli [9, 10] used sliding window and density features. Similarly, methodologies presented in [11-15] used sliding window and geometrical features.

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

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

However, it is anticipated that the holistic methods [8-15] heavily rely on the size of lexicon, and are not suitable for recognition domain with large lexicons. Because lexicon size greatly affects the recognition performance, reduction of unlikely lexicon words would increase the chances to find correct matching lexicon words during the recognition process. Mozaffari et al. [16] proposed a lexicon reduction scheme for static Farsi handwriting recognition by analyzing dots within characters.

Different from the holistic methods, the segmentation based strategy puts more efforts on recognizing individual characters. In order to recognize the individual characters, it needs to find the characters boundaries first. One of the typical and well established segmentation methods is over-segmentation and validation. The main reason to adopt an over-segmenter is to find possible character boundaries even if it produces unnecessary segmentation points, called over-segmentation points. Then, validation process intends to remove the over-segmentation points. There is handful of researchers favoring the over-segmentation strategy. Viard-Gaudin et al. [17] used stroke-based over-segmentation. Kim et al. [18] proposed a segmentation based on the break points of connected letters. Liu et al. [19] and Verma et al. [20] used over-segmentation by locating the handwriting features. A rule-based over-segmentation and validation is used in [21]. Verma et al. [22] also proposed over-segmentation and validation approach by introducing Borda count. In [23], over-segmented primitives of cursive handwritten month words to process handwritten bank cheques are fed to HMM and neural networks to find the optimum segmentation paths. Vellasques et al. [24] distinguishes the validation from filtration of the over-segmentation points. They also define that the excessive amount of unnecessary over-segmentation point directly affects the recognition performance, and the filtration can remove up to 83% of the unnecessary over-segmentation points.

In previous segmentation studies [17-24], validation is to check the primitive bound by two neighboring segmentation points for the legality of being a letter. The validation also occurs in orderly manner, normally left-to-right. Because it checks the spatial relationship between primitives, a validation of the current segmentation point might be affected by the incorrect previous segmentation point, namely chain-failure. To reduce the cause of the validation problem, BSNV is presented in our proposal. BSNV works with Suspicious Segmentation Points (SSP) generation module. It intends to separate an image into two sub-images on a Nominated Suspicious Segmentation Point (NSSP). NSSP is a SSP nominated by selection rules. Depending on sub-image

978-1-4244-3553-1/09/$25.00 ©2009 IEEE

selection criteria, one sub-image is nominated and segmented into two sub-images successively. The selection and segmentation processes are continuously repeated until the termination conditions are satisfied. Finally, neural validation attempts to sort out the valid segmentation point from the resulting segmentation of the prior process. The resulting segmentation points of neural validation are the final segmentation points.

The rest of this paper is organized into four sections. Section II describes the proposed binary segmentation with neural validation in detail. Also, it is described how over-segmentation, binary segmentation and neural validation work together. 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. Binary Segmentation with Neural Validation

The overview of the proposed Binary Segmentation with Neural Validation strategy is shown below in Fig. 1.

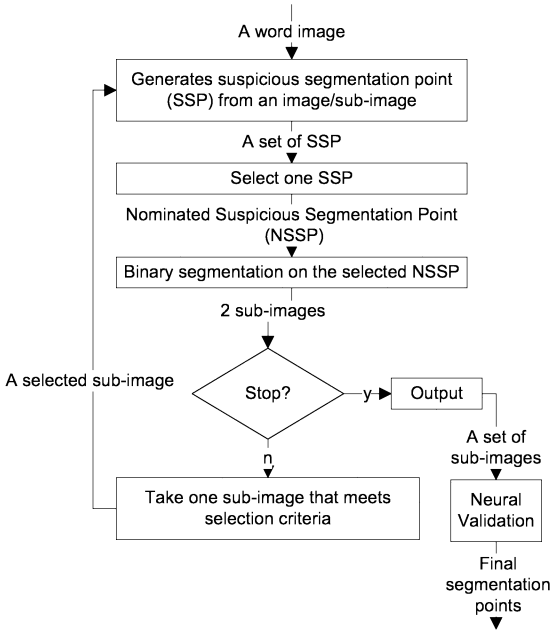

Fig. 1. Overview of binary segmentation with neural validation

A. Suspicious Segmentation Points (SSP)

The intention of this module is to dissect the text words into primitives by over-segmentation as described in [25]. As a summary of the over-segmentation process, firstly parameters of stroke width and baselines are calculated from each word image. Based on the parameters, over-segmentation between baselines is performed to produce raw over-segmentation points. Those raw over-segmentation points are passed through multiple validation modules such as Hole Detection and Total Foreground Pixel Comparison, to decide the final Suspicious Segmentation Points (SSP). The great details of the over-segmentation adopted here are in [25].

B. Sub-image Selection Criteria

BSNV is repetitive method which produces 2 segmented sub-images at each iteration. So, at the end of each step, one of sub-images needs to be selected for next segmentation until no more segmentation on sub-images are required. In the proposal, the selection criteria were devised based on the conventional knowledge that the bigger the image, the more letters it may contain. Given an image $\mathbb{I}(x,y)$ where x and y indicate the width and the height of a image, y is constant over sub-images. So, the chance to be selected among candidate sub-images grows linearly to x. Simply, the wider the image, the higher chance to be chosen for next segmentation. However, the largest x value of an image doesn't guarantee that the image is selected for next segmentation, because the selection process not only depends on the value of x, but also it depends on other criteria, which concerns the Suspicious Segmentation Points (SSP) and non-connecting overlapping characters.

The second selection criteria checks if candidate sub-images have SSP, so BSNV can segment on. Those who do not have SSP are not segmentable, so they are removed from the selection candidates. As stated before, the SSP is generated by over-segmenting a sub-image. The sub-image is binary segmented at a Nominated Suspicious Segmentation Point (NSSP), which is a chosen SSP with the highest confidence value to find a character boundary out of all other SSP for the sub-image. Also, it is believed to be a crucial task to choose a right NSSP out of the SSP. At the current stage, SSP are represented as the sub-x-coordinates of Image $\mathbb{I}(x,y)$. So, a SSP is chosen to be a NSSP if the value of $|x \div 2\text{-SSP}|$ is the smallest among the SSP in a sub-image.

In summary of the selection criteria, a sub-image is to be selected for segmentation if its x value is the biggest among others and has SSP. However, if a sub-image is separable by space between characters, it is binary segmented through the space regardless of a NSSP. This rule is employed to give higher priority for non-connected overlapping characters to be segmented first.

C. Stop Conditions

As mentioned in the previous section, the principle of the binary segmentation is to dissect the wider segment first. However, to stop the repetitive binary segmentation, a threshold needs to be set first. Those who are smaller in width than the threshold should not be segmented any further. However, it is impossible to decide the perfect threshold value for handwritten characters, since the sizes of individual characters vary. It is predictable that if the threshold value is too big, there will be more likely for a sub-image to contain more than a character. On the other hand, the smaller threshold value will tend to produce sub-images containing only partial of a letter, not whole. In the proposed approach, Average Character Width (ACW) and Minimum Character Width (MCW) are estimated before segmentation on a root image, so ACW and MCW are constants over all sub-images through segmentations. Those values are used as threshold values.

Another terminating condition was articulated based on the number of characters of the longest lexicon word. Each segmentation point represents a boundary between neighboring letters. Since it's known that the maximum

number of characters, the Maximum Number of Segmentation (MNS) can be set accordingly. For any given words, the total number of segmentation should not exceed MNS. In other words, each sub-image represents a character, which means that the number of sub-images in a word image shouldn't be more than MNS. So, the binary segmentation process can be stopped by counting the sub-images (e.g. if the number of sub-images is equal to MNS). This condition can be also described from the perspective of iteration of binary segmentation. An iteration of BSNV takes one image and emits two sub-images, which means that there will be n+1 sub-images after n iteration. Likewise, the binary segmentation process can be stopped when n-1 equals to MNS.

Next terminating condition for BSNV is to ensure that the x value of every sub-image $\mathbb{I}(x,y)$ is between MCW and MCW+ACW. This condition assumes that every sub-image is close to the size of characters, so no further segmentation is necessary. Up to this point, termination of segmentation relies on the size of sub-images and the number of iteration of binary segmentation. There is another condition added, which is to check if there is any sub-images over-segmentable to generate SSP. Since binary segmentation chooses one NSSP out of SSP, no SSP means no more segmentation can be done.

To put all the terminating conditions of binary segmentation, firstly the number of iteration is limited to MNS. Secondly, the sizes of sub-images are checked. Finally, there should exist a sub-image over-segmentable.

D. Contour Tracing

In the proposed approach, foreground pixel contour tracing algorithm is used to find a segmentation path. On a given NSSP (Black dashed line in Fig. 2), searching for a segmentation path starts tracing from an end pixel of lower bound of the given NSSP. The tracing ends when a tracing pixel reaches a pixel on upper bound. The underlying strategy is to minimize crossing through the continuous foreground pixels (stroke) and to avoid cutting characters multiple times. In Fig. 2, the NSSP (black dashed line) crosses two strokes (Stroke crossing 1 & 2), but the crossing through the first stroke (Stroke crossing 1) is unnecessary because tracing can continue through background pixels until it gets cross the second stroke (Stroke crossing 2). The segmentation path with gray solid line is the one traced with the algorithm.

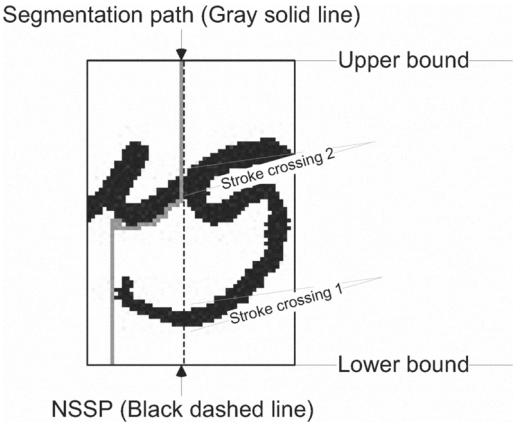

Fig. 2. Contour tracing for segmentation path

E. Binary Segmentation Steps

Step 1. Input a handwritten word image for segmentation and a lexicon.

Step 2. Select a word with the maximum number of characters from the lexicon. Set a variable, Maxinum Number of Segmentation (MNS) as:
MNS = count characters of the chosen word in the lexicon

Step 3. Calculate baselines

Step 4. Calculate the following parameters.
StrokeWidth = estimate Stroke Width
AvgCharWidth = a distance between baselines

Step 5. Generate Suspicious Segmentation Points (SSP).

Step 6. Find middle segmentation point. (segmentation point closest to middle pixel column of the image), and set it as Nominated Suspicious Segmentation Point (NSSP)

Step 7. Search a segmentation path by contour tracing algorithm.

Step 8. Divide the image by the segmentation path.

Step 9. After dividing image into 2 parts, check both parts for following conditions.
a. If both parts have larger height than minimum character height.
b. If both parts have larger width than minimum character width.
c. If both parts have more than the minimum amount of foreground pixels.
If any of these conditions fail, then remove the selected suspicious segmentation point, and go to Step 6 to select other one.

Step 10. Collect all the primitives created so far and count them how many there are. If the count is bigger than *MNS*, then STOP.

Step 11. If not, select one that meets following conditions from the collected primitives.
a. Has suspicious segmentation points
b. The width is bigger than *AvgCharWidth*
c. The largest one in width meeting a) & b)
If found one, then take it and go to Step 6. Otherwise, STOP.

Step 12. Repeat Step 5 – 11 until STOP conditions are satisfied.

F. Neural Validation

Neural validation is performed by a neural character classifier trained on pre-segmented characters. The neural classifier attempts to resolve the three segments if each segment is a legal character or not. The three segments are defined as left, right and joined segment. As shown in Fig. 3, the left segment is a sub-image formed between the left-neighbouring segmentation path and the testing segmentation path. Likewise, the right segment is defined by the right-neighbouring segmentation path instead of left one. Finally, the joined segment is formed by the left and right neighbouring segmentation paths. The neural classifier is capable of distinguish which sub-images are characters or rubbish. For each validation of a segmentation path, three

classifications are made. Correct classification of the left or right segments gives positive votes. However, correct classification of the joined segment gives negative votes, because it implies that the validating segmentation path is cutting a character into half. After the three classifications for a validating segmentation path, the votes are counted. If the votes are less than the threshold, the validating segmentation path is regarded as invalid, and will be removed from the final segmentation paths. The neural validation architecture is shown in Fig. 4.

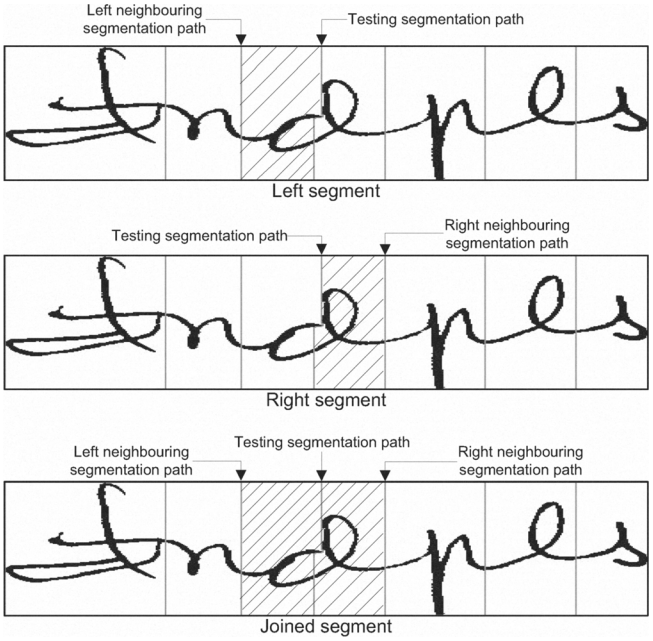

Fig. 3. Definition of segments

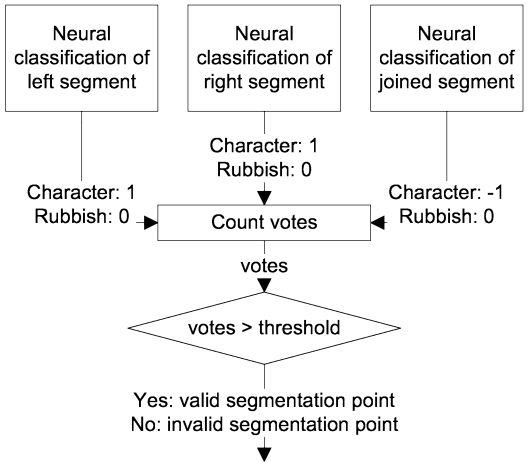

Fig. 4. Neural validation architecture

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

The experiment is conducted on a CEDAR benchmark database to perform the comparative analysis against the segmentation results from the literature. The final segmentation was experimented on 311 words out of 317 words from CEDAR\TEST\CITIES\BD directory.

C. Neural Networks Training

A MLP neural network with a single hidden layer was trained on pre-segmented characters with back-propagation learning algorithm. It takes 100 inputs, and produces 53 outputs. The 53 outputs represent 52 alphabets (upper and lower cases) and 1 rubbish character. The number of hidden units and the number of iteration were incremented by an interval for each training. The number of hidden units with the best training result was used in the experimental.

D. Segmentation Performance Criteria

As described in [26], 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 don't belong to under-segmentation and over-segmentation, and don't separate two characters correctly. The final segmentation results are calculated by dividing each categorical result by total number of characters used in experiment.

Fig. 5 shows an example of the transitional progressive results when suspicious segmentation point generation, binary segmentation and neural validation were applied sequentially. Fig. 6 shows the sequential view of the sub-image combined together after each step of binary segmentation. Table 1 displays the final segmentation results on the CEDAR database.

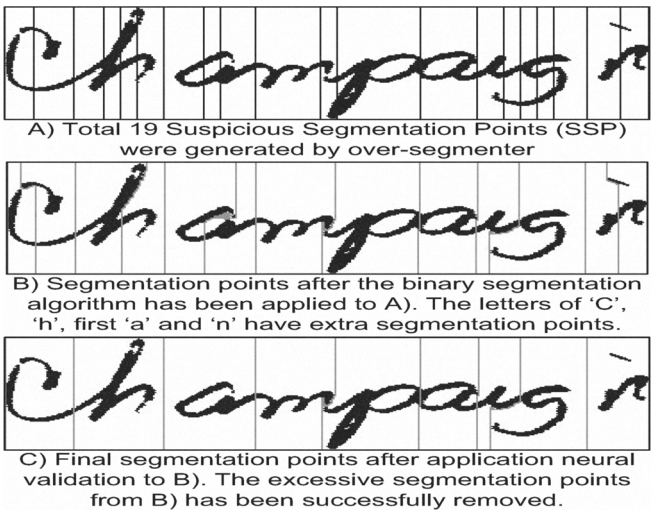

Fig. 5. This describes an example of the transitional progressive results between applications of modules like A) Suspicious Segmentation Point generation, B) binary segmentation, and C) neural validation.

TABLE 1. FINAL SEGMENTATION PERFORMANCE RESULTS

Segmentation rate (%)			
Under	Over	Bad	Average
16.55	3.85	5.62	8.68

IV. ANALYSIS AND DISCUSSION

As shown in Table 1, the under segmentation rate is 16.55% which is much higher than the over segmentation rate (3.85%) and bad segmentation rate (5.62%). 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 the proposed algorithm with four other algorithms published in the literature and a comparative analysis is provided to give a relative look of the effectiveness of the proposed algorithm. The overall segmentation performance of the proposed approach was lower than the segmentation accuracies published in the literature. Especially, under segmentation error was much higher than the segmentation results published in the literature. However, over and bad segmentation errors in the proposed method were improved comparing to [21]. Noticeably, the proposed approach produced much higher segmentation errors than [26, 27]. Authors in [26] mentioned in their research that their database has been carefully generated from group of writers to minimize the slope and slant angles. So, it is hypothetical why their method has produced much less segmentation errors. The segmentation error was very low in [27], possibly because the database was reduced to 200 out of 317 to remove the distorted words.

Fig. 6. This presents the sequential view of the sub-images combined together after an iteration of segmentation. The numbers below the images correspond to the segmentation order.

Comparing to the results published in the literature, the experiments using the proposed approach generated much higher under segmentation errors. One of the main reasons is that the proposed approach performed the neural validation with a neural classifier of 61.4% accuracy. It is plausible that the true negative classification contributed to the higher under segmentation error. So, in the future research, improvement in the neural classifier should be anticipated beforehand.

V. CONCLUSIONS AND FUTURE RESEARCH

In this paper, a novel binary segmentation with neural validation (BSNV) for off-line handwriting recognition has been proposed and investigated. The segmentation and validation approach contains over-segmentation based suspicious segmentation point generator, binary segmentation and neural validation modules. The new segmentation paradigm BSNV has been tested on CEDAR benchmark database. The proposed approach exhibits lower performance in under segmentation errors, but shows competitive performance in bad and over segmentation errors.

REFERENCES

[1] J. Sadri, C. Suen, and T. Bui, "A genetic framework using contextual knowledge for segmentation and recognition of handwritten numeral strings," *Pattern Recognition,* vol. 40, no. 3, pp. 898-919, 2007.

[2] S. Lee, and J. Kim, "Complementary combination of holistic and component analysis for recognition of low-resolution video character images," *Pattern Recognition Letters,* vol. 29, no. 4, pp. 383-391, 2008.

[3] M. Kherallah, L. Haddad, A. Alimi and M. Amar, "On-line handwritten digit recognition based on trajectory and velocity modeling," *Pattern Recognition Letters,* vol. 29, no. 5, pp. 580-594, 2008.

[4] A. Broumandnia, J. Shanbehzadeh, and M. R. Varnoosfaderani, "Persian/arabic handwritten word recognition using M-band packet wavelet transform," *Image Vision Comput.,* vol. 26, no. 6, pp. 829-842, 2008.

[5] H. Fujisawa, "Forty years of research in character and document recognition--an industrial perspective," *Pattern Recognition,* vol. 41, no. 8, pp. 2435-2446, 2008.

[6] B. Verma, and H. Lee, "A segmentation based adaptive approach for cursive handwritten text recognition," *Proceedings of IEEE International Joint Conference on Neural Networks (IJCNN07)*, Orlando, Florida, USA, pp. 2212-2216, 2007.

[7] R. M. Suresh, and S. Arumugam, "Fuzzy technique based recognition of handwritten characters," *Image and Vision Computing,* vol. 25, no. 2, pp. 230-239, 2007.

[8] A. Benouareth, A. Ennaji, and M. Sellami, "Semi-continuous HMMs with explicit state duration for unconstrained Arabic word modeling and recognition," *Pattern Recognition Letters,* vol. 29, no. 12, pp. 1742-1752, 2008.

[9] A. Vinciarelli, "Application of information retrieval techniques to single writer documents," *Pattern Recognition Letters,* vol. 26, no. 14, pp. 2262-2271, 2005.

[10] A. Vinciarelli, and J. Luettin, "A new normalization technique for cursive handwritten words," *Pattern Recognition Letters,* vol. 22, no. 9, pp. 1043-1050, 2001.

[11] R. Bertolami, M. Zimmermann, and H. Bunke, "Rejection strategies for offline handwritten text line recognition," *Pattern Recognition Letters,* vol. 27, no. 16, pp. 2005-2012, 2006.

[12] S. Gunter, and H. Bunke, "Feature selection algorithms for the generation of multiple classifier systems and their application to handwritten word recognition," *Pattern Recognition Letters,* vol. 25, no. 11, pp. 1323-1336, 2004.

[13] T. H. Su, T. W. Zhang, D. J. Guan and H. J. Huang, "Off-line recognition of realistic Chinese handwriting using segmentation-free strategy," *Pattern Recognition,* vol. 42, no. 1, pp. 167-182, 2009.

[14] H. Bunke, and S. Gunter, "HMM-based handwritten word recognition: on the optimization of the number of states, training iterations and Gaussian components," *Pattern Recognition,* vol. 37, no. 10, pp. 2069-2079, 2004.

[15] R. Nopsuwanchai, A. Biem, and W. F. Clocksin, "Maximization of mutual information for offline Thai handwriting recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 28, no. 8, pp. 1347-1351, 2006.
[16] S. Mozaffari, K. Faez, V. Margner and E. Haikal, "Lexicon reduction using dots for off-line Farsi/Arabic handwritten word recognition," *Pattern Recognition Letters,* vol. 29, no. 6, pp. 724-734, 2008.
[17] C. Viard-Gaudin, P. M. Lallican, and S. Knerr, "Recognition-directed recovering of temporal information from handwriting images," *Pattern Recognition Letters,* vol. 26, no. 16, pp. 2537-2548, 2005.
[18] K. K. Kim, J. H. Kim, and C. Suen, "Segmentation-based recognition of handwritten touching pairs of digits using structural features," *Pattern Recognition Letters,* vol. 23, no. 1-3, pp. 13-24, 2002.
[19] J. Liu, and P. Gader, "Neural networks with enhanced outlier rejection ability for off-line handwritten word recognition," *Pattern Recognition,* vol. 35, no. 10, pp. 2061-2071, 2002.
[20] B. Verma, M. Blumenstein, and M. Ghosh, "A novel approach for structural feature extraction: contour vs. direction," *Pattern Recognition Letters,* vol. 25, no. 9, pp. 975-988, 2004.
[21] 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, 2003.
[22] B. Verma, P. Gader, and W. Chen, "Fusion of multiple handwritten word recognition techniques," *Pattern Recognition Letters,* vol. 22, no. 9, pp. 991-998, 2001.
[23] Q. Xu, L. Lam, and C. Suen, "Automatic Segmentation and Recognition System for Handwritten Dates on Canadian Bank Cheques." *Proceedings of Seventh International Conference on Document Analysis and Recognition (ICDAR'03)*, Edinburgh, Scotland, vol. 2, pp. 704-708, 2003.
[24] E. Vellasques, L. S. Oliveira, A. S. Britto Jr, A. L. Koerich and R. Sabourin, "Filtering segmentation cuts for digit string recognition," *Pattern Recognition,* vol. 41, no. 10, pp. 3044-3053, 2008.
[25] H. Lee, and B. Verma, "A novel multiple experts and fusion based segmentation algorithm for cursive handwriting recognition," *IEEE World Congress on Computational Intelligence (WCCI08)*, Hong Kong, China, pp. 2994-2999, 2008.
[26] B. Yanikoglu, and P. A. Sandon, "Segmentation of off-line cursive handwriting using linear programming," *Pattern Recognition,* vol. 31, pp. 1825-1833, 1998.
[27] X. Xiao, and G. Leedham, "Knowledge-based English cursive script segmentation," *Pattern Recognition Letters,* vol. 21, no. 10, pp. 945-954, 2000.