Copyright © 2009 Institute of Electrical and electronics Engineers, Inc.
All Rights reserved.
Personal use of this material, including one hard copy reproduction, is permitted.
Permission to reprint, republish and/or distribute this material in whole or in part for any other purposes must be obtained from the IEEE.
For information on obtaining permission, send an e-mail message to stds-igr@ieee.org.
By choosing to view this document, you agree to all provisions of the copyright laws protecting it.
Individual documents posted on this site may carry slightly different copyright restrictions.
For specific document information, check the copyright notice at the beginning of each document.

Over-Segmentation and Validation Strategy for Off-line Cursive Handwriting Recognition

Hone Lee
Computing Science
CQUniversity
Bundaberg, Australia
h.lee1@cqu.edu.au

Brijesh Verma
Computing Science
CQUniversity
Rockhampton, Australia
b.verma@cqu.edu.au

Abstract— **This paper presents an over-segmentation and validation strategy for off-line cursive handwriting recognition. Over-segmentation module is employed to find all the possible character boundaries. Then, the incorrect segmentation points from over-segmenting module are removed by validating processes. The over-segmentation was performed based on the vertical pixel density between upper and lower baselines. Wherever the pixel density is less than threshold, an over-segmentation point is assigned. After the over-segmentation is done, validation starts removing over-segmentation points. The first validation module checks if a segmentation point lies in hole region. The second validation module compares total foreground pixel between two neighbouring segmentation points to a threshold value. The third validation module is neural network voting by neural network classifier trained on pre-segmented characters. Finally, the oversized segment validation process checks if there is any missing segmentation point between neighbouring characters. The proposed approach has been implemented, and the experiments on CEDAR benchmark database 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.**

Keywords-off-line handwriting recognition; segmentation; neural networks

I. INTRODUCTION

The ultimate goal of off-line cursive handwriting recognition is machine simulation of human reading. The useful applications have been found in many industrial sectors like postal service, form processing, bank cheque processing, and historical manuscript conversion into electronic format. However, after more than four decades of efforts from many researchers, the performance of off-line cursive handwriting recognition is not good enough for real world application [1, 2].

A. *Off-Line Handwriting Recognition*

The typical internal process of off-line handwriting recognition consists of preprocessing, segmentation, and recognition. However, some of the stages are merged or omitted, depending on the methods of recognition. In general, the pre-processing and normalization algorithms are independent of the recognition approach of the system, but segmentation is tightly coupled with recognition algorithms [3]. The general off-line handwriting recognition system is shown below in Fig. 1.

1) Pre-processing

A series of document analysis tasks are required prior to recognizing letters from scanned documents. Some common processes are thresholding [4], noise removal [5], slant and slope correction as normalization, and thinning. The main objective of the pre-processing is to produce an image containing the word to be recognized without any other disturbing elements.

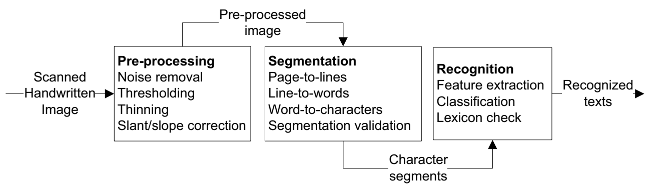

Fig. 1. Framework of off-line handwriting recognition

2) Segmentation

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 [6-8]. 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. Because of such factors, the most existing segmentation algorithms confront major problems, such as inaccurate character cuttings, missing segmentation points, and over-segmentation of a same character [9, 10]. Currently available segmentation techniques are holistic [11, 12], dissection [13], recognition-based segmentation [14], and over-segmentation approaches [15-17].

3) Recognition

During the recognition process, generally the shape features are extracted from the segmented images, and the appropriate class is assigned to the observed character. Various feature extraction techniques are incorporated with various intelligent classifiers in the literature. A black box model [18], contour profile [19], stroke direction [20], and global and local information [21] are fed to neural network classifiers. HMM

Authorized licensed use limited to: CENTRAL QUEENSLAND UNIVERSITY. Downloaded on May 21, 2009 at 21:48 from IEEE Xplore. Restrictions apply.

based classifiers are used with features of angle, distance, horizontal and vertical spans [22].

The rest of this paper is organized into four sections. Section II describes the proposed over-segmentation and validation algorithms 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 OVER-SEGMENTATION AND VALIDATION ALGORITHMS

The overview of the proposed over-segmentation and validation approach is shown below in Fig. 2.

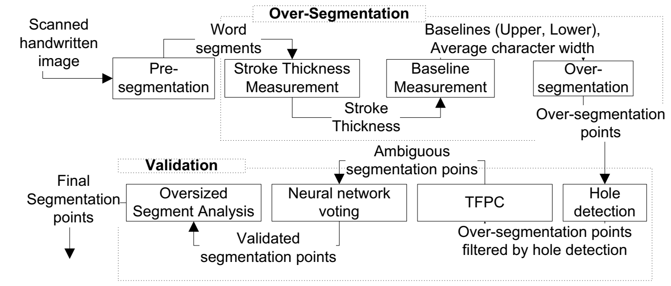
Over-Segmentation
Scanned handwritten image
Word segments
Baselines (Upper, Lower), Average character width
Pre-segmentation
Stroke Thickness Measurement
Baseline Measurement
Over-segmentation
Stroke Thickness
Over-segmentation points
Ambiguous segmentation poins
Validation
Final Segmentation points
Oversized Segment Analysis
Neural network voting
TFPC
Hole detection
Validated segmentation points
Over-segmentation points filtered by hole detection

Fig. 2. Proposed over-segmentation and validation approach

A. Pre-Segmentation Process

The pre-segmentation component prepares the handwritten image for the character segmentation process. Line and word segmentation processes are performed in this step. The outcome of this process is the word image tokens, and they are delivered to over-segmentation process for character segmentation.

B. Over-Segmentation Process

The purpose of this module is to separate the text words into characters. Firstly, parameters of stroke thickness and baselines are calculated from each word image outputted through pre-segmentation process. Based on the parameters, over-segmentation between baselines is performed to produce over-segmentation points. Those over-segmentation points will be passed through multiple validation modules to decide the final segmentation points.

1) Stroke Thickness Measurement

In the proposed approach, the stroke thickness is calculated on each word to reflect the variation between words. To prevent the over-measurement of the stroke thickness, the maximum boundary has been set to a value, one fourth of word image height. Through the horizontal and vertical scan of the image for the transition distances of foregrounds, the most occurring transition distance becomes the stroke thickness of the word image.

2) Baseline Calculation

Measuring the baselines by the horizontal pixel density histogram is one of the favorite methods among researchers. However, the proposed approach uses a novel method 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 thickness. 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 displays the examples of baseline measurement. The first example reflects the intended outcome of the algorithm. However, the second one draws the upper baseline too high because of extensive ligature of the first letter, R.

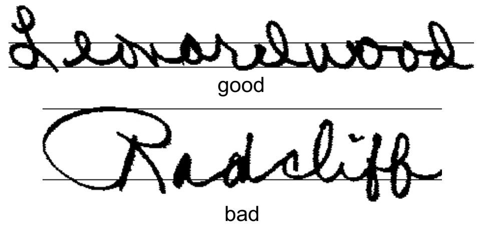

Fig. 3. Baseline comparison

3) Over-Segmentation between Baselines

Before proceeding to over-segmentation, there is a crucial 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 divided into smaller sizes having the same width as the stroke thickness to prevent under-segmentation. However, this algorithm fails on touching points bigger than (StrokeThickness × 2). Fig. 4 displays the different results on different segmentation thresholds, and the segmentation threshold in the example b) produces better results because it prevents under-segmentation problems.

C. Multiple Validation Modules

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

1) 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. 5, the segmentation points on the characters with holes like ‘L’, ‘e’, ‘o’, ‘d’, ‘o’, ‘o’, and ‘d’ from a word ‘Leonardwood’ have been removed by the algorithm.

2) Total Foreground Pixel Comparison (TFPC)

Based on an assumption that a legal character contains the minimum amount of foreground pixels equal to a threshold, (StrokeThickness × (LowerBaseline – UpperBaseline)), each

Authorized licensed use limited to: CENTRAL QUEENSLAND UNIVERSITY. Downloaded on May 21, 2009 at 21:48 from IEEE Xplore. Restrictions apply.

segment between two neighboring segmentation points is checked for legality. If the total foreground pixel of a segment is less than the threshold, its segmentation points are merged or moved to a prospective new segmentation point within the segment. This algorithm scans through all over-segmentation points recursively from left end to the right end. An example of the TFPC application is show in Fig. 6.

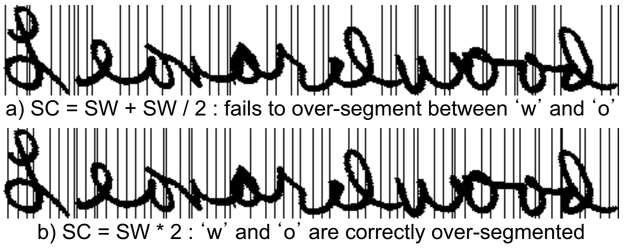
a) SC = SW + SW / 2 : fails to over-segment between 'w' and 'o'
b) SC = SW * 2 : 'w' and 'o' are correctly over-segmented

Fig. 4. Over-segmentation results depending on segment threshold (SC) and stroke thickness (SW)

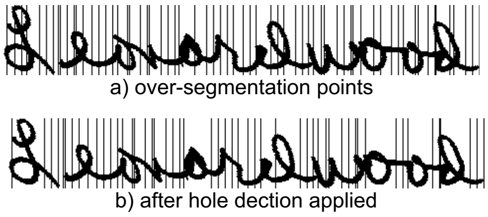

Fig. 5. Examples of hole detection algorithm application

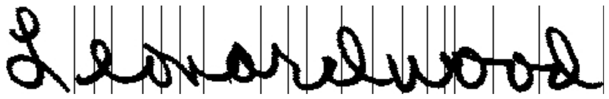

Fig. 6. TFPC application to example b) from Fig. 5

3) *Validation by Neural Network Voting*

Validation by neural network voting is a type of knowledge-based validating process on over-segmentation points. The neural network is trained on pre-segmented character shapes. The neural classifier takes 100 transition features of a character as an input and outputs one of 53 classes, which are 52 classes for lower and upper cases of alphabets, and 1 for rubbish characters. In the 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. 7), the left, right and joined segment checkers get a left, right, joined segment region (shown in Fig. 8) 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.

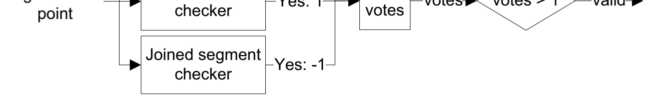

Fig. 7. Architecture of the neural network voting module

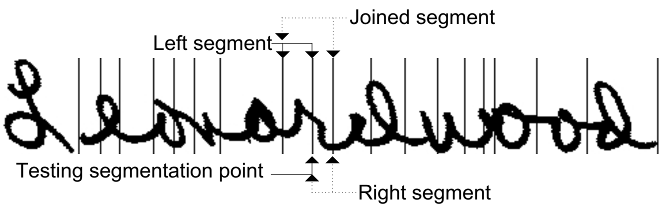

Fig. 8. Segment regions based on testing segmentation point

4) *Oversized Segment Analysis*

The oversized segment analysis is adapted to prevent missing segmentation point between neighboring characters. The detection of oversized segment is based on measuring the width of the segment, and compares the width to pre-defined threshold. In the proposed approach, the threshold is set to the height of each testing word outputted through pre-segmentation module. On detection, a new segmentation point is located on middle of the segment.

III. EXPERIMENTAL RESULTS

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

A. *Implementation*

The proposed approach used C++ programming language to implement all the algorithms described in previous section. The implementation strictly followed the 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. For neural network training, pre-segmented 5708 characters were used from CEDAR\TRAIN\BINANUM\BD and BL directories. The training data set was divided into two data sets for training and testing in 9 to 1 ratios. The final segmentation was experimented on 311 words from CEDAR\TEST\CITIES\BD directory.

The neural character classifier also needs rubbish character database during training. However, there is no benchmark database for rubbish characters. So, the rubbish characters were generated by over-segmenting characters from training set. By manual inspection of the over-segmented primitives, the most dissimilar primitives to characters were selected as rubbish characters.

C. *Neural Network Training*

An Error Back-Propagation (EBP) algorithm is used to train the neural classifier. A 100 transition feature of a character was taken as an input to the classifier. The output of the classifier is one of 53 classes. 52 classes are for 52 English alphabet characters (upper + lower cases), and 1 for the representation of non-character, rubbish character. After multiple attempts with different configurations of neural network parameters, 61.4% of classification accuracy was

Authorized licensed use limited to: CENTRAL QUEENSLAND UNIVERSITY. Downloaded on May 21, 2009 at 21:48 from IEEE Xplore. Restrictions apply.

achieved. In future research, improving the accuracy of the neural classifier should proceed over other improvements.

D. *Final Segmentation Results*

Segmentation results are measured by counting the segmentation errors. The segmentation errors have three categories, over-segmentation, under-segmentation, and bad-segmentation as described in [8]. The over-segmentation error occurs if a character is segmented into more than three segments. Under-segmentation occurs if there is a missing segmentation point between two neighboring characters. Lastly, the bad-segmentation is defined as 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 performance result is shown in Table I. The segmentation error rate was calculated by dividing the number of error for each category by the total number of character in testing database.

IV. ANALYSIS AND DISCUSSION

The final segmentation results shown in TABLE I are the segmentation error rates for each segmentation category. Firstly, the over-segmentation error was the highest as 6.58%, followed by under-segmentation error of 6.47%. The bad-segmentation error was the lowest as 3.49%. The average segmentation error was 5.5%.

Under-segmentation error could be caused by following reasons. Firstly, the over-segmentation module failed to locate the segmentation point initially. But from the previous study, the under-segmentation error from the over-segmenter was close to zero. So, it is unlikely to have caused such higher under-segmentation error in the experiment, but it is possible. The second suspicious module is hole detection process. If two neighboring characters are connected and the connected area forms hole, then any segmentation points located in that region will be removed by the hole detection. In cursive script, many characters are connected by multiple points, and its common example can be found when 'tt' is cursively handwritten. The third reason is the poor performance of the neural classifier. Even if a segmentation point is correctly located, failure to classify left, right and joined segments correctly leads to remove the segmentation point, which causes a missing segmentation point. The final cause is the poor performance of the oversized segment analysis module. The oversized segment analysis module largely depends on the height of the handwritten word image, and the oversized segment detection occurs if a segment pixel width is bigger than the height. In words in CEDAR database, the height of a character is much bigger than the width. The factor would be one of the main reasons to prevent triggering oversized segment detection.

In the experiment, the most segmentation error was over-segmentation error. The over-segmentation points were intentionally generated by over-segmenter to prevent under-segmentation error, and those over-segmentation points were supposed to be removed by the multiple validation experts. Especially, the total foreground pixel comparison and neural voting modules were intended to remove excessive over-segmentation points. However, the total foreground pixel comparison module only removes the over-segmentation point by checking structural validity of a segment. So, the final decision for a segmentation to be correct was up to the neural voting validation module because the neural classifier has the knowledge of the character shape. However, the accuracy of the classifier used in the experiment was only 61.4%. The poor classifier can misclassify rubbish characters into correct characters. Therefore, the segmentation point is regarded as correct, and the factor influences the over-segmentation error. During manual inspection of the segmentation results, the characters like 'w' and 'm' were highly over-segmented comparing to other characters.

The main contributor of the bad-segmentation error could be the neural voting validator with the poor neural classifier. Also, the oversized segment analysis could cause the bad-segmentation error by locating a new segmentation point by finding a middle point of the segment. The method finding a new segmentation point causes a bad-segmentation error if the widths of the characters are not the same.

TABLE I. FINAL SEGMENTATION PERFORMANCE RESULTS ON CEDAR

Classifier accuracy (%)	**Segmentation error rate (%)**			
	Under	***Over***	***Bad***	***Average***
61.4	6.47	6.58	3.49	5.50

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 the proposed 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 shown in Table II, the average incorrect segmentation rate is 5.50% which is lower than the ones in [10, 17]. Especially the bad segmentation error has been remarkably improved. The over segmentation ratio obtained by the proposed algorithm is fairly close to the algorithm in [10] but much lower than the algorithm in [17]. On the other hand, the under segmentation error produced by the proposed algorithm has been significantly unimproved and is much higher than the published error in the literature.

Comparing to the literature, the experiments using the proposed approach generated much higher under-segmentation errors. One of the main reasons is that the proposed approach does not have the mechanisms to correct slope angle of the handwritten words. However, massive slope angles exist in almost every testing word. Those big slope angles do impact on calculating baselines in the proposed approach. The proposed approach heavily depends on baselines for over-segmentation processing, calculating average character width, and analyzing oversized segments. Another possible reason of higher under-segmentation error is that many characters are multi-connected and the connectivity forms a hole-region, which are marked as incorrect segmentation points by hole detection algorithm.

Authorized licensed use limited to: CENTRAL QUEENSLAND UNIVERSITY. Downloaded on May 21, 2009 at 21:48 from IEEE Xplore. Restrictions apply.

TABLE II. SEGMENTATION PERFORMANCE COMPARISON ON CEDAR

	Segmentation rate (%)			
	Over	*Under*	*Bad*	*Average*
[10]	7.4	2.0	11.6	7.0
[17]	10.0	0.2	8.7	6.3
Proposed approach	6.58	6.47	3.49	5.50

V. CONCLUSIONS

In this paper, an over-segmentation and validation strategy for off-line cursive handwriting recognition has been proposed and investigated. The over-segmentation was performed based on pixel density between baselines. Multiple validation module contains hole detection, foreground pixel comparison and neural voting. Also, oversized segment analysis is performed before producing final segmentation points. The new over-segmentation and validation paradigm has been tested on CEDAR benchmark database and on local cursive handwritten text database. The proposed approach produced lowest errors in comparison to existing approaches.

ACKNOWLEDGEMENT

This research was supported by ARC and ISSNIP collaborative research scheme.

REFERENCES

[1] 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.

[2] F. Camastra, “A SVM-based cursive character recognizer,” *Pattern Recognition,* vol. 40, no. 12, pp. 3721-3727, 2007.

[3] S. Alma'adeed, C. Higgins, and D. Elliman, “Off-line recognition of handwritten Arabic words using multiple hidden Markov models,” *Knowledge-Based Systems,* vol. 17, no. 2-4, pp. 75-79, 2004.

[4] R. Milewski, and V. Govindaraju, “Binarization and cleanup of handwritten text from carbon copy medical form images,” *Pattern Recognition,* vol. 41, no. 4, pp. 1308-1315, 2008.

[5] Z. Lin, R. Wang, and H.-Y. Shum, “Rule-based cleanup of on-line English ink notes,” *Pattern Recognition,* vol. 39, no. 6, pp. 1074-1087, 2006.

[6] A. Elnagar, and R. Alhajj, “Segmentation of connected handwritten numeral strings,” *Pattern Recognition,* vol. 36, no. 3, pp. 625-634, 2003.

[7] X. Xiao, and G. Leedham, “Knowledge-based English 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, pp. 1825-1833, 1998.

[9] 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.

[10] 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, WA, USA, 2001, pp. 1142-1146.

[11] 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.

[12] J. Ruiz-Pinales, R. Jaime-Rivas, and M. Castro-Bleda, “Holistic cursive word recognition based on perceptual features,” *Pattern Recognition Letters,* vol. 28, no. 13, pp. 1600-1609, 2007.

[13] 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.

[14] 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.

[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] L. Prevost, L. c. 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.

[17] B. Verma, “A contour code feature based segmentation for handwriting recognition,” *Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR)*, Aug. 2003, pp. 1203.

[18] F. Lauer, C. Suen, and G. Bloch, “A trainable feature extractor for handwritten digit recognition,” *Pattern Recognition,* vol. 40, no. 6, pp. 1816-1824, 2007.

[19] 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.

[20] M. Blumenstein, X. Y. Liu, and B. Verma, “An investigation of the modified direction feature for cursive character recognition,” *Pattern Recognition,* vol. 40, no. 2, pp. 376-388, 2007.

[21] C. O. Freitas, L. Oliveira, S. Aire, and F. Bortolozzi, "Zoning and metaclasses for character recognition," *Proceedings of ACM Symposium on Applied Computing*, Seoul, Korea, 2007, pp. 632-636.

[22] S. Mahmoud, “Recognition of writer-independent off-line handwritten Arabic (Indian) numerals using hidden Markov models,” *Signal Process.,* vol. 88, no. 4, pp. 844-857, 2008.

Authorized licensed use limited to: CENTRAL QUEENSLAND UNIVERSITY. Downloaded on May 21, 2009 at 21:48 from IEEE Xplore. Restrictions apply.

Authorized licensed use limited to: CENTRAL QUEENSLAND UNIVERSITY. Downloaded on May 21, 2009 at 21:48 from IEEE Xplore. Restrictions apply.