A character segmentation algorithm for off-line handwritten script recognition

Nelson H C Yung, Andrew H S Lai* and Perry Z P Chua

The University of Hong Kong, Department of Electrical & Electronic Engineering
Pokfulam Road, Hong Kong

*University of Surrey, Department of Electrical & Electronic Engineering
Guildford, Surry GU2 5XH, United Kingdom

ABSTRACT

In this paper, a new character segmentation algorithm for dealing with off-line handwritten script recognition is presented. The X-axis projection, Y-axis projection and geometric classes techniques used by the algorithm proves to be successful in segmenting normal handwriting with a success rate of 93.5%. As a result of this development, detailed understanding of geometric classes of English characters and the difficult cases in segmentation was gained. Although the algorithm works quite well with a randomly chosen sample, results of a detailed analysis may shed new light into the tuning of the algorithm especially for segmenting the identified difficult cases.

Keywords: handwritten script, character segmentation, geometric classes, X-axis projection, Y-axis projection

1. INTRODUCTION

Recognition of handwritten scripts in both the on-line and off-line cases have been an active research area around the world because of the need to store the ever growing amount of handwritten documents in computer form, and retrieve them for reference[1]. Some of the research activities are for example, interpreting handwritten addresses[2], recognizing numerals[3], recognizing signatures[4] and characters other than English[5]. To some extend, these problems are specific by focusing on the description and recognition rather than the segmentation of characters. For instance, the interpretation of handwritten addresses and signature recognition concentrate on the whole word, whereas the handwritten numerals recognition assumes the numerals are written separated. If we consider the problem in general, the wide varieties of qualities, shapes and personal styles of handwriting continue to pose a major challenge to researchers in this field. Although many optical character recognition (OCR) systems for printed characters are available commercially, many system designs and algorithms that work with printed characters may not work well with handwriting[6]. Essentially, printed characters and handwriting have a number of major differences in characteristics. In summary, a printed page often has characters of different sizes and fonts, and a page is normally very well structured with graphics, lines, words and most characters are clearly separated by gaps[7]. In handwriting, the variation in sizes and fonts is probably minimal, but a page is usually less structured and unconstrained with lines not necessarily straight or horizontal, and more importantly, characters are mostly connected and sometimes overlapped to form a word with ambiguities between characters. At worst, handwriting could be so 'personalized' that classification of individual characters by a machine is almost impossible[4]. Even for the moderate cases, the problems associated with classification and character segmentation intertwine to create a multitude of problems.

In this paper, a character segmentation algorithm is presented for off-line handwritten script recognition. This work is driven by the believe that a good and accurate character segmentation process will greatly improve the recognition rate of the system. This is contrary to leaving the problem to a later stage, relying on the classifier and/or spell checker/lexicon, as adopted by some other groups[6,7]. The segmentation algorithm described here relies on three principles: X-axis projection (vertical), Y-axis projection (horizontal) and geometric classes of characters and words. By employing these three principles appropriately in the algorithm, minimum width characters can be segmented out without much problem. By evaluating the algorithm against a randomly chosen written sample, the segmentation success rate is found to be 93.5%. Apart from requiring the written sample to have a consistent writing style, words and lines are separated by detectable gaps, and lines should be near horizontal, all the other normal variations in handwriting are allowed and can be accommodated by the algorithm. In essence, the use of Y-axis projection in conjunction with the concept of geometric classes improves the X-axis projection results and fine tune the final list of segmentation points. The detailed analysis of the segmentation results shows that there exist characters such as ‘n’, ‘m’, ‘u’,

0-8194-1858-7/95/$6.00

'w' that are naturally difficult to be segmented. Fortunately, the occurrence of some of these characters in a page of text is not high compared with some others such as 'e', 'i', 's'. Based on this argument, normal segmentation success rate is expected to be well above 90%.

The organization of this paper is as follows: Section 2 overviews the off-line handwritten script recognition system and discusses the significance of various components in the processing chain. Section 3 describes the segmentation algorithm in detail with sub-sections covering the goal and objectives of the algorithm, the writing styles and constraints, the concept of geometric classes and the signal flow of the algorithm itself. Section 4 presents the segmentation results with a brief description of the implementation. The original text, the original image, thinned version and segmented version are given and a detailed analysis of the success rate, rejection rate and others will also be given. Section 5 concludes the paper by evaluating the merits and drawbacks of the algorithm. Further research direction will also be discussed.

2. OFF-LINE HANDWRITTEN SCRIPT RECOGNITION

It is well-known that the problem of off-line handwritten script recognition is more demanding than the on-line case as most of the dynamic information is lost in the static representation. Techniques making use of timing or pressure information can no longer be applied[9]. However, if existing handwritten documents are to be integrated into computer forms, sophisticated and robust algorithms must be specially designed to handle this more difficult case with acceptable recognition success rate. From this observation, this research is directed to develop an off-line recognition system that can convert static handwritten script into ASCII text. Currently, the recognition system has a processing chain which can be divided into six stages: image acquisition, page management, thinning, feature segmentation, description and classification. The block diagram representing this system architecture is depicted in figure 1.

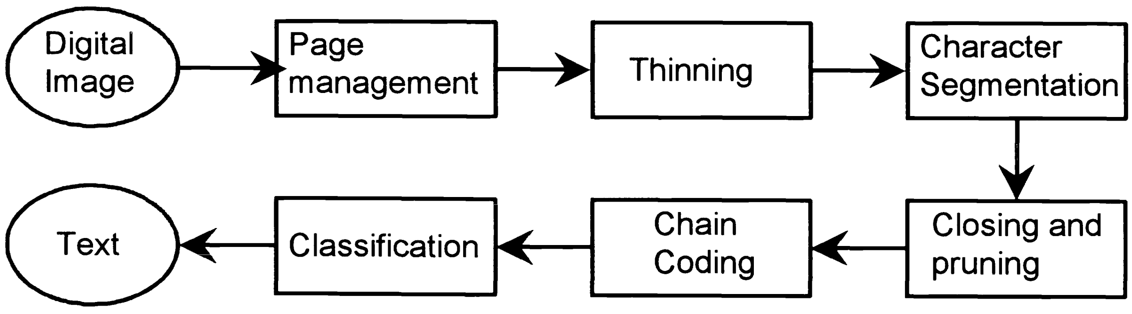


Figure 1 : Block diagram of an off-line handwritten script recognition system

The critical components of the system are likely the character segmentation, closing and pruning, chain coding and classification. Image acquisition is relatively simple with today's scanner technology. For instance, the binary image used in Section 4 was acquired through a HP ScanJet IIcx flat-bed scanner of which similar technology is wide accessible. Page management can be rather straight-forward if we assume the A4 page consists of handwritten script only. Other objects such as graphs , diagrams, and etc. are assumed to be absent from the page.

In terms of the thinning process, as long as the thinning algorithm produces a set of unique skeletons representing the original handwriting, any additional or complex thinning features are not necessary. The thinning algorithm used in our case is the standard Medial Axis Transformation. The reason for choosing this algorithm is because of its algorithmic and implementation simplicity. Unfortunately, it is well-known that the result of thinning is far from satisfactory as it tends to leave holes and parasitic components that will likely affect the recognition of characters at a later stage (figure 2). Although this drawback of thinning could introduce a considerable amount of recognition error, it is actually used to the advantage of the character segmentation algorithm, giving the segmentation algorithm a lot more unique features per character to work on. The success of the segmentation tests given in Section 4 proves this point. On the other hand, if these holes and parasitic elements are left to the description and classification stages, the recognition rate will certainly be degraded due to these inaccuracies. To avoid this, closing and pruning are performed before the thinned segmented characters are being described by the chain code. By using morphological operators, this task can be accomplished easily.

In principle, our emphasis is placed on performing a robust and reliable character segmentation with high success rate rather than leaving the segmentation problem to a later stage. The advantage of this approach is that the eventual classification errors would be more likely due to the classifier rather than the character segmentation. The principle of our segmentation algorithm relies on the geometric classes and X-axis, Y-axis projections of a word. A character is segmented from a word based on these these calculated features. Details of the algorithm will be presented in Section 3.

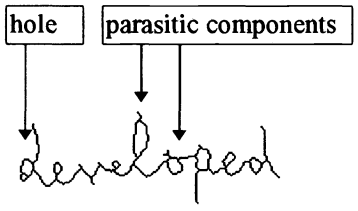


Figure 2 : Holes and parasitic components in thinned characters

The chain coding algorithm does not need to be sophisticated as such. Moreover, the only requirement is that the coding must be consistent for characters with branching points. The reason is that a depth-first-search is usually employed in chain coding. When reaching a branching point, the chain code will traverse all its neighboring pixels in a clockwise (or anti-clockwise) direction disregarding the original strokes of the writer. To alleviate the problem, a modification is introduced for determining the direction of strokes at a branching point. This includes calculating the current direction at the branching point and all the possible outgoing directions. With such knowledge, the correct outgoing direction is simply the one closest to the current incoming direction. The standard Freeman chain code used in the system has this slight modification implemented and the resultant chain code is consistent at branching points.

The classification algorithm is as important as the character segmentation algorithm. The success of the system depends largely on the recognition success rate. At present, a Bayes classifier and a structural classifier are being implemented and tested.

3. CHARACTER SEGMENTATION ALGORITHM

3.1 Goal and concept of the algorithm

The ultimate goal of the algorithm is to segment the characters with minimum width in a word with at least 90% of segmentation success rate. The term 'minimum width' refers to the resultant segmented width of a character is minimum such that long connecting strokes between characters will not be included as part of the character. With that, the segmented width of a character would be relatively constant disregarding the connecting strokes between the character and its neighboring characters. Furthermore, a high segmentation success rate is necessary if the handwritten script recognition system is to give acceptable recognition success rate.

The concept of the algorithm is derived from the ideas that firstly, the horizontal point that separates two thinned characters are either a clear space or a single pixel. By calculating the X-axis projection, this can be determined easily. Secondly, a stroke connecting two characters must be a monotonic function either increasing or decreasing. This function is characterized by its almost identical X-axis and Y-axis projections. Further it is bascially different from the projections of most characters. Thirdly, English characters can be classified according to four geometric classes in which every character has a middle portion. If only this portion is considered instead of the whole character, segmentation will be more accurate and problems such as intrusion may be handled correctly.

3.2 Writing styles and constraints

In general, there is no strict requirements on the writing style except for the few constraints discussed below. However, the scripts that have been successfully tested are at least 0.5 cm in character height, small size and style variation, words are separated by clear spaces, lines are near horizontal and non-overlapped and there is no graphics and other non-textual objects on the same page.

The first constraint is that words on a page must be properly written and separated by detectable gaps. Although connected words do not impose any problems to the segmentation algorithm, it will introduce problems during word recognition or spell checking. The second constraint is that lines should be near horizontal. Slightly slanted lines are tolerable. Severe slanting must be corrected before the segmentation algorithm can perform correctly. The third constraint is that lines must not overlap with each other. If that is the case, the whole segmentation and recognition process will fail. Other than these constraints, characters in a word can be connected or separated, and their slight variation on the same horizontal line is acceptable.

3.3 Concept of geometric classes

After an in-depth study of the geometric characteristics of English characters, we can conclude that these characters can be divided into four mutually exclusive classes: UPPER, LOWER, MIDDLE and WHOLE. If we consider dividing a character space into three roughly equal horizontal regions, a character must have a middle portion. This may be the only portion in the character for characters such as 'a', 'c', 'e'. Moreover, there are characters that have the middle portion plus a portion in the upper region, or lower region or both. For example, an UPPER class character such as 'b', 'd', 'f' have an upper portion above the middle portion. On the other hand, characters such as 'g', 'p' have a lower portion below the middle portion, and 'j' has both upper and lower portions. The definitions of classes and their members are summarized in Table 1. In general, English words can also be classified in exactly the same way. Some examples are given in Table 2.

Class	Definition	Members
UPPER	Characters having an upper portion above the average height of small case letters	b, d, f, h, i, k, l, t, all capital letters
LOWER	Characters having a lower portion below the bottom line of letters	g, p, q, y
MIDDLE	Characters do not have any properties of UPPER and LOWER classes, i.e. their heights are close to the average height of small case letters	a, c, e, m, n, o, r, s, u, v, w, x, z
WHOLE	Characters having all the properties of UPPER and LOWER classes	j

Table 1 : Classification of English characters according to their geometric characteristics

Class	Examples
UPPER	HKU, One, handwritten characters
LOWER	pay, gone, page, query
MIDDLE	area, one, come, corns
WHOLE	Pay, jump, funny, galaxy

Table 2 : Classification of some of the English words according to their geometric characteristics

In the case of handwritten script, characters do not necessary fall into these classes as distinctively as printed characters. Depending on the style of the writer, there can be a wide degree of variation. Initial study shows that most properly written characters can be successfully classified. The small percentage of misses is usually due to inconsistent or poor writing style. In those cases, even human would have difficulties classifying them. If the writing style is consistent, such problem could be eliminated.

In the character segmentation algorithm presented here, this geometric class information is used to differentiate which region the Y-axis projection will be performed, and eventually determine which points are to be nominated as segmentation points. So far, the use of geometric classes enable us to handle words with overlapping characters and successfully generate good segmentation points.

3.4 Character segmentation

After a page of script is thinned, a valid separation between two lines must be a gap between the two lines. When considering the projection of the image onto the Y-axis, this gap should be denoted by a number of consecutive locations with L-Frequency=0, where L-Frequency is defined by equation (1) for an N by M image.

$$\text{L-Frequency} = \sum_{i=0}^{N-1} f(i,y) \qquad \text{for } y=0, 1, \ldots, M-1 \qquad (1)$$

A valid separation between two words must also be a gap between the words. It is defined as when the word is projected onto the X-axis, a separation is denoted by a number of consecutive locations with W-Frequency=0, where W-Frequency is defined by equation (2)

$$\text{W-Frequency} = \sum_{j=y}^{y+W_h} f(x,j) \qquad \text{for } x=0, 1, \ldots, N-1 \qquad (2)$$

where W_h is the maximum word height and y denotes the y-coordinate of the word's origin. In the case of a valid separation between characters in a word, it must be either a gap with C-Frequency=0 or with a thin and long segment that when projected onto the X-axis, have C-Frequency=1 (equation (3)).

$$\text{C-Frequency} = \sum_{j=y_i}^{y_i+W_h} f(x,j) \qquad \text{for } x=x_i, x_i+1, \ldots, x_i+W_w \qquad (3)$$

where W_w is the word width and (x_i y_i) is the origin of the word. From these arguments, we derived the algorithm as follows with the block diagram depicted in figure 3.

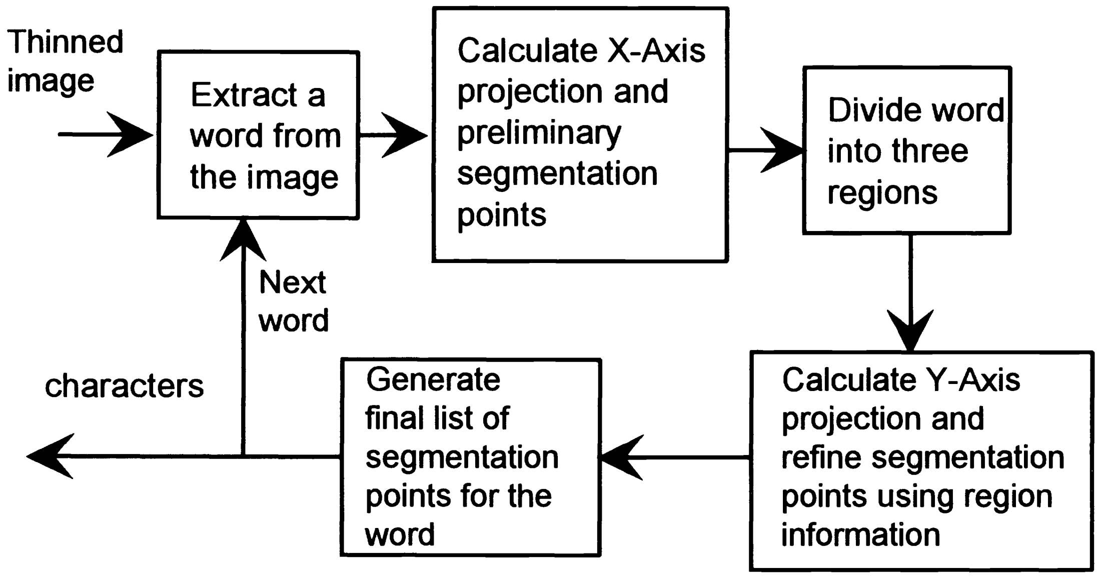

Thinned image
Extract a word from the image
Calculate X-Axis projection and preliminary segmentation points
Divide word into three regions
Calculate Y-Axis projection and refine segmentation points using region information
Generate final list of segmentation points for the word
Next word
characters

Figure 3 : Block diagram of the segmentation algorithm

The first step of the segmentation algorithm is to extract a word from the page. This is achieved by identifying the gaps separating lines and words using equations (1) and (2). Once a word is extracted, the X-axis projection of the word is calculated by using equation (3). This projection allows us to assign a preliminary set of segmentation points according to our definition of separations between characters. The details of assignment is described in the following.

Every point on the X-axis is classified into three classes: GAP (C-Frequency=0), SINGLE (C-Frequency=1) or NO_M (C-Frequency>1). As the total number of points per word can be quite high, the list of classes is further simplified by removing local variations between SINGLE and NO_M. The two local variations are the sequences of SINGLE-NO_M-SINGLE (S-N-S) and NO_M-SINGLE-NO_M (N-S-N) as a result of the first assignment. In these two cases, the center location is reassigned to the class of its two neighbors. For instance, S-N-S becomes S-S-S and N-S-N becomes N-N-N. The reason for doing this is to eliminate the possibility of having a single NO_M being classified as a character because it is between two SINGLE, or a SINGLE being classified as a segmentation point because it is between two NO_M. Furthermore, if two characters are separated by a space, it is likely that our assignment of classes will end up with a long string of GAP. These consecutive GAP should be merged at their mid-point to reduce the amount of computation when assigning preliminary and final segmentation points. If consecutive SINGLE are detected, they are also merged as one SINGLE (and renamed as MID_PT) at their mid-point for the same reason. An example of the X-axis projection and the three classes is depicted in figure 4.

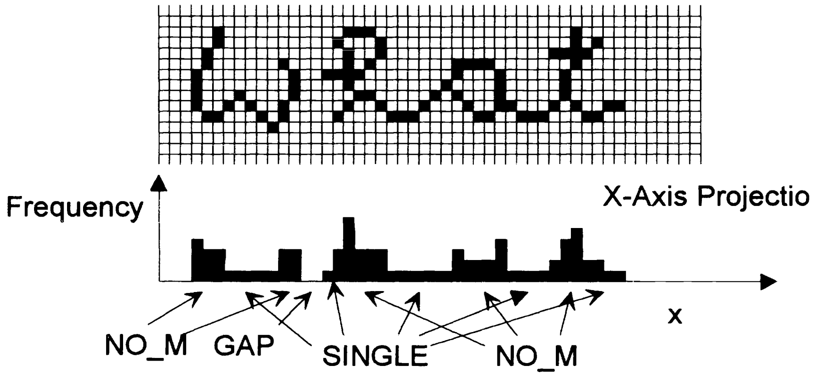


Figure 4 : X-axis projection and the three classes

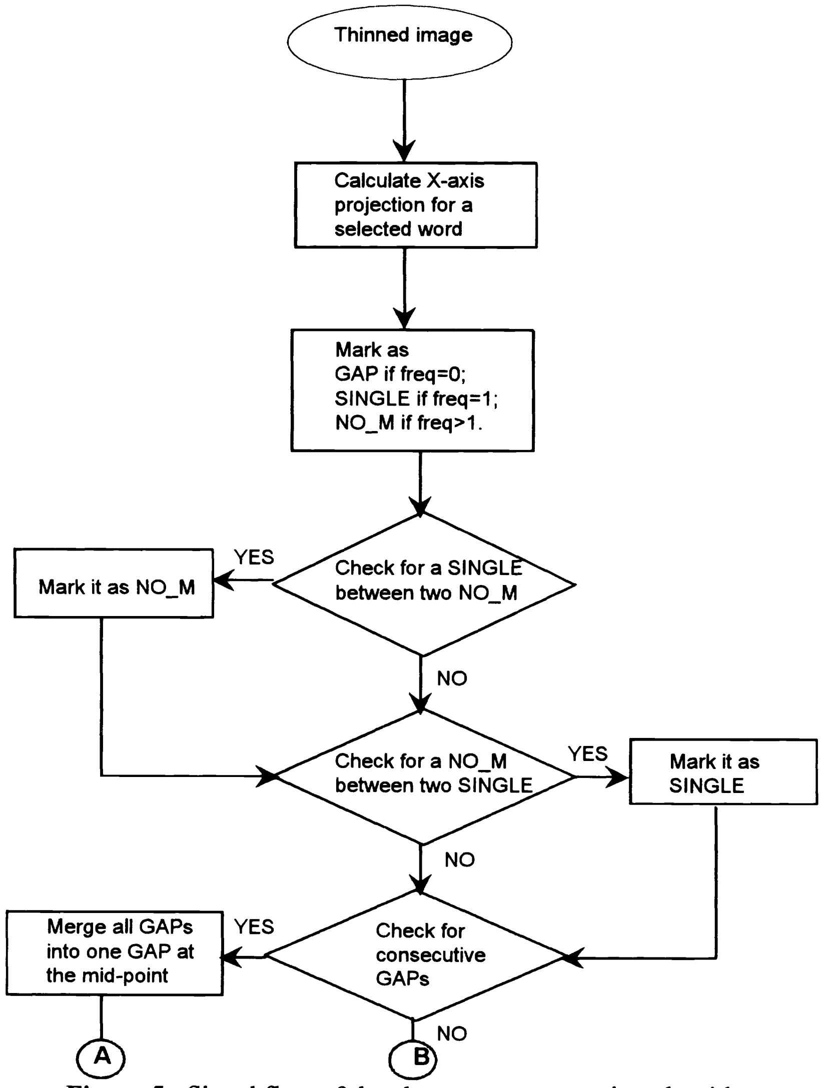

Thinned image
Calculate X-axis projection for a selected word
Mark as GAP if freq=0; SINGLE if freq=1; NO_M if freq>1.
Check for a SINGLE between two NO_M
YES
Mark it as NO_M
NO
Check for a NO_M between two SINGLE
YES
Mark it as SINGLE
NO
Check for consecutive GAPs
YES
Merge all GAPs into one GAP at the mid-point
NO
A
B

Figure 5 : Signal flow of the character segmentation algorithm

The second step of the algorithm is to check through all the MID_PT to see whether they are valid separations. Note that GAP is a definite segmentation point. To do this, we use the geometric class concept introduced in Section 3.3. This is simply done by calculating the height of the word, then divide the word into three horizontal regions with height determined by the distinct valleys in its Y-axis projection, and calculate the pixel concentration for each region. If no distinct valleys are found, then the word is divided into three equal regions and the pixel concentration for each region is calculated. The region with the highest pixel concentration is most likely the region in the middle as all characters have the middle part, but not necessarily the upper or lower parts. However, it could also be other regions such as the upper region in the case of upper-case characters.

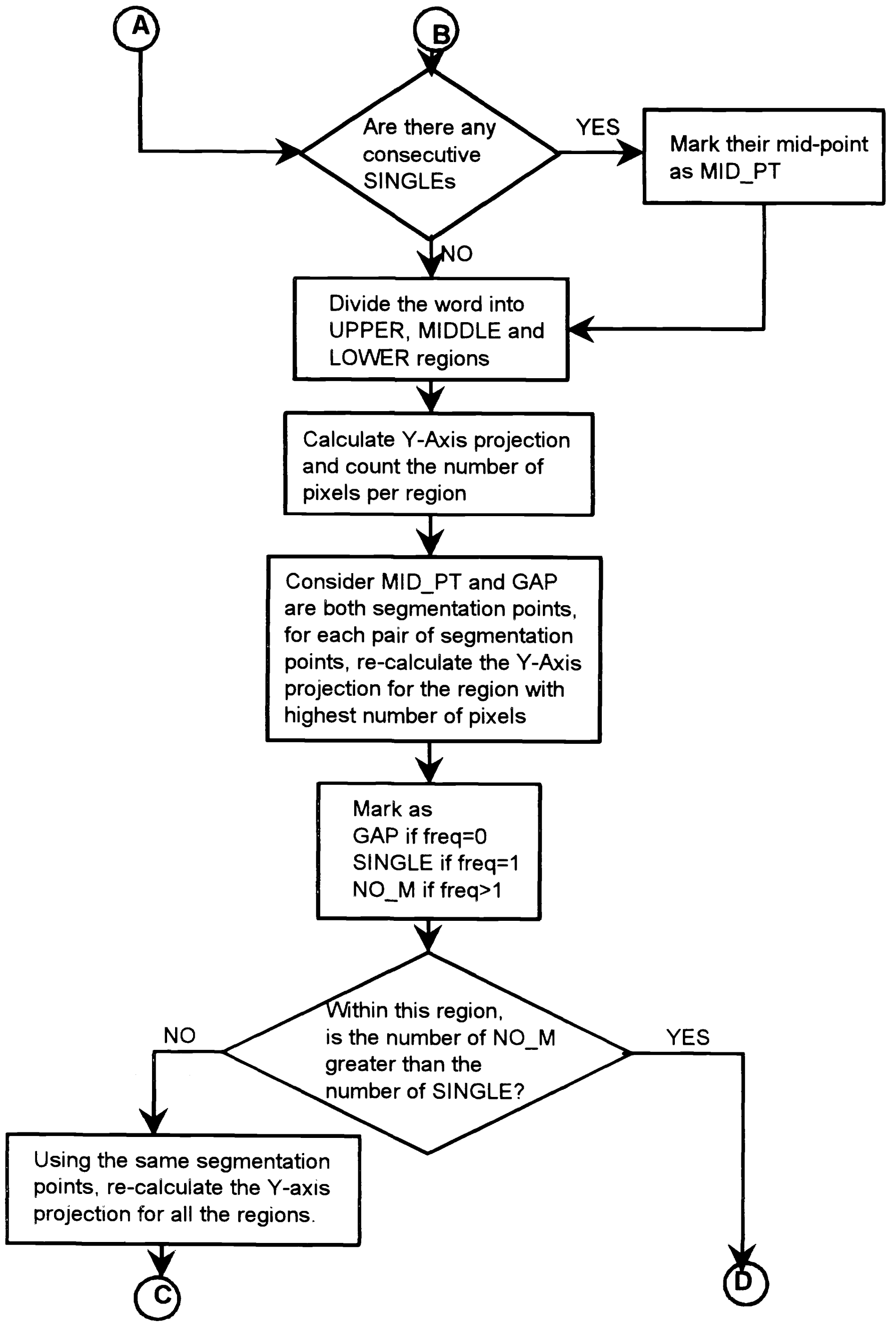

A
B
Are there any consecutive SINGLEs
YES
Mark their mid-point as MID_PT
NO
Divide the word into UPPER, MIDDLE and LOWER regions
Calculate Y-Axis projection and count the number of pixels per region
Consider MID_PT and GAP are both segmentation points, for each pair of segmentation points, re-calculate the Y-Axis projection for the region with highest number of pixels
Mark as GAP if freq=0 SINGLE if freq=1 NO_M if freq>1
Within this region, is the number of NO_M greater than the number of SINGLE?
NO
YES
Using the same segmentation points, re-calculate the Y-axis projection for all the regions.
C
D

Figure 5 (continued)

Based on this knowledge of regions, each pair of adjacent GAP or MID_PT is considered from left to right across the word. This consideration involves re-calculating the Y-axis projection for the region with the highest number of pixels, assign the GAP, SINGLE and NO_M classes for the Y-axis projection. If the number of NO_M is greater than the number of SINGLE, this implies the variation of strokes in the chosen region is not monotonic. In other words, the pair of adjacent GAP or MID_PT under consideration are valid segmentation points and rename them as CHECK. If the stroke variation within the region is monotonic, there is a high possibility that the stroke is a connecting segment between two characters, except in some special cases such as 't', 'l', 'c', 'I', 'j', where the region chosen appears to have a monotonic behavior but the segment is actually part of a character. To accommodate these special cases, the algorithm counts the number of SINGLE and NO_M as well. If the total number of SINGLE and NO_M is greater than the number of GAP, then treat the pair of adjacent GAP or MID_PT as valid segmentation points and rename them as CHECK. The reason for this is that if the stroke is a connecting segment between two characters, there will be a large number of GAP in relation to the number of NO_M and SINGLE. Otherwise, the number of NO_M and SINGLE will be in majority.

For MID_PT that do not satisfy any of the above criteria, change them into NO_M except the case of three consecutive MID_PT. In this case, leave the center MID_PT unchanged and convert the other two to NO_M. This center MID_PT will be left as a possible segmentation point and will be used at a later stage if the character is not recognized correctly. The segmentation process completes when all the MID_PT are checked. The final list of segmentation points for the word consists of locations of GAP and CHECK points with additional MID_PT that may be considered as possible segmentation points. The signal flow of the algorithm is depicted in figure 5.

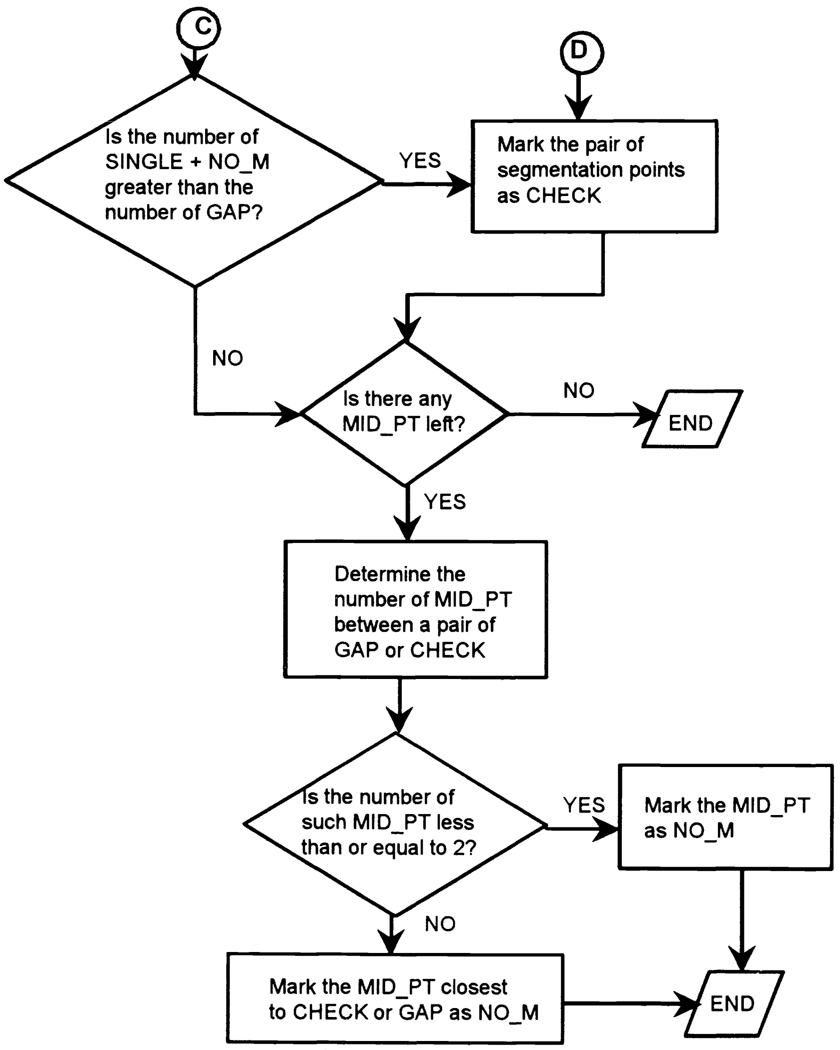

C
D
Is the number of SINGLE + NO_M greater than the number of GAP?
YES
Mark the pair of segmentation points as CHECK
NO
Is there any MID_PT left?
NO
END
YES
Determine the number of MID_PT between a pair of GAP or CHECK
Is the number of such MID_PT less than or equal to 2?
YES
Mark the MID_PT as NO_M
NO
Mark the MID_PT closest to CHECK or GAP as NO_M
END

Figure 5 (continued)

4. SEGMENTATION RESULTS

The segmentation algorithm was implemented using C++ in a MS Windows environment. The test script is a paragraph of text as depicted in figure 6. It consists of 168 characters, 33 words and 5 lines written in normal felt-tip pen. This paragraph consists of upper-case and lower-case characters without punctuation marks, and the words and characters cover most classes of what is discussed in Section 3.3. Altogether 90 samples were collected with each sample written by a different student in which one of the samples was randomly chosen for the purpose of testing the algorithm. The digitized binary image of this sample is depicted in figure 7. This sample can be considered as a good test sample as characters are reasonably upright, character and word sizes do not vary much and most characters and words are near horizontal. Most of the characters in a word are connected, while in some cases such as 'What', 'Scanner', one or two characters in the word are separated by a space. In addition, separations between words and lines are clearly identifiable and the writing style is consistent.

What you have written will be a test
Sample for the Optical Character Recognition
currently being developed and this paragraph
will be digitized by an optical scanner
and then recognized by a computer

Figure 6 : Test script

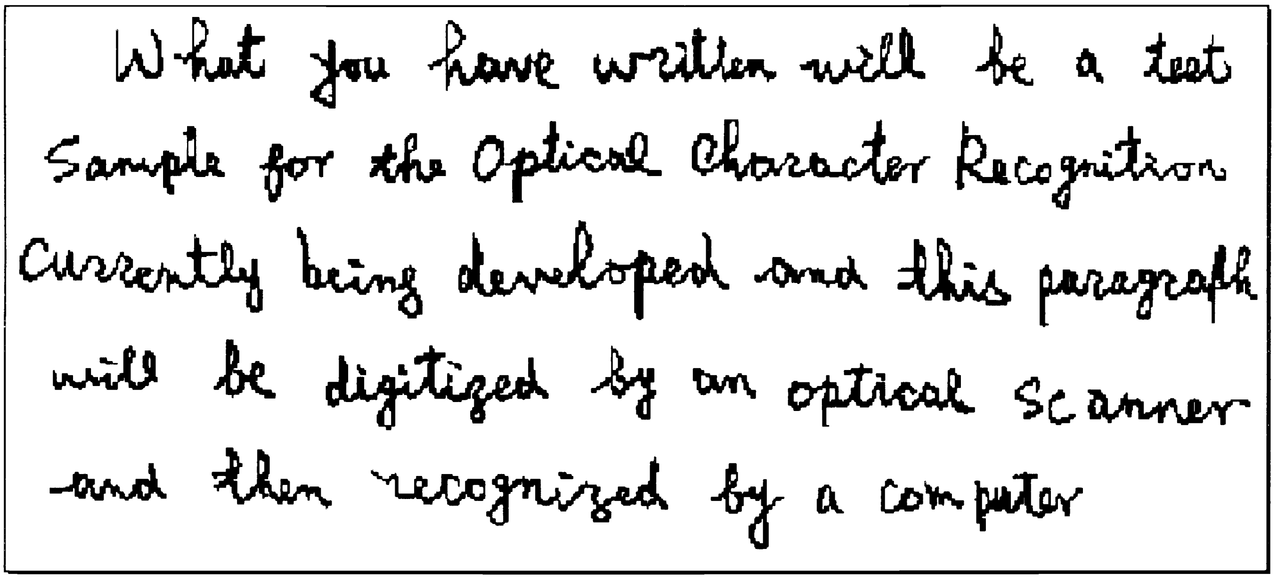

What you have written will be a test
Sample for the Optical Character Recognition
Currently being developed and this paragraph
will be digitized by an optical Scanner
and then recognized by a computer

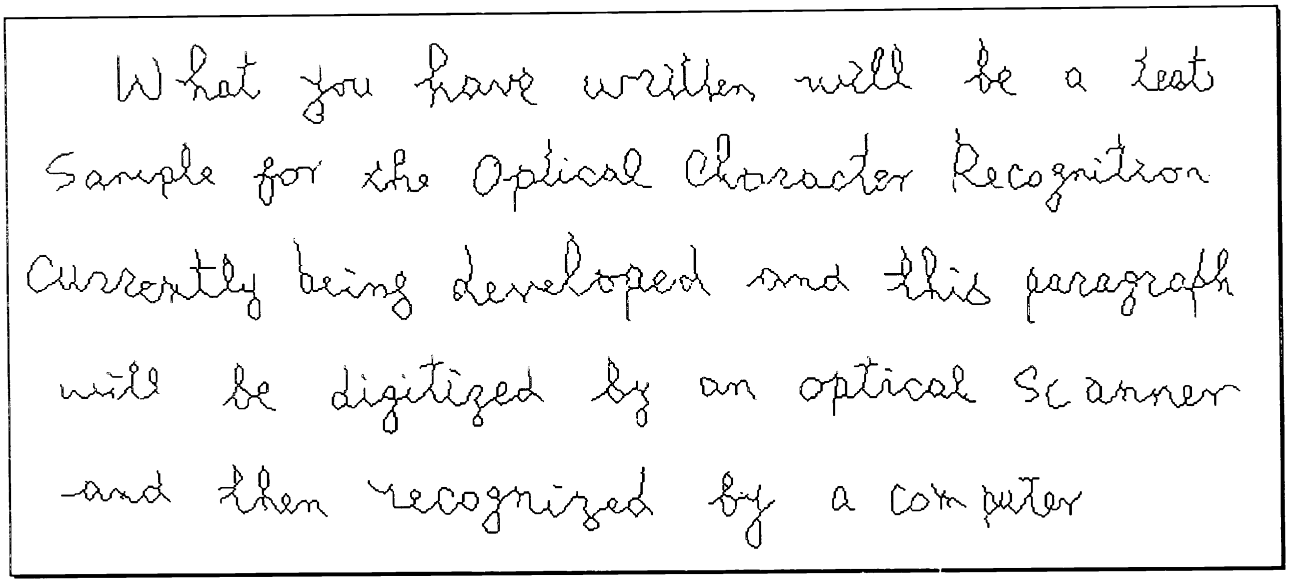
Figure 7 : Original Image

This image was scanned into a PCX format as black and white drawing, and the resolution was kept to a minimum. As can be seen from figure 7, the characters appear to be rugged and coarse. One explanation is the low resolution employed in the digitization. The other reason is that character strokes tend to diffuse slightly due to the nature of the felt-tip pen and the paper used. This effect is considered normal in a handwritten script.

The thinned image of figure 7 is depicted in figure 8. The result clearly illustrates the problem of holes and parasitic components associated with thinning. The overall thinning result is acceptable as the characteristics of words and characters are retained and certain degree of stroke fuzziness in the original image have been after the thinning process.

The segmentation result is depicted in figure 9. In this figure, the extra horizontal lines across each word denote the regional division of the word. For words such as 'Optical' and 'digitized', the division fits well with the geometric characteristics of the words, and therefore, the middle region is the one with the highest pixel concentration. However, other words such as 'What', 'Character' have divisions that are not so. The region that has the highest pixel concentration in these cases is the lower region.

The vertical lines pass through the segmentation points. It should also be noted that the left most and right most points of a word are natural segmentation points but are not shown in the figure as vertical lines. They will be considered as segmentation points in our subsequent discussion. Incorrectly segmented characters are marked with a vertical arrow and a number tag.



What you have written will be a test
Sample for the Optical Character Recognition
Currently being developed and this paragraph
will be digitized by an optical Scanner
and then recognized by a computer

Figure 8 : Thinned Image

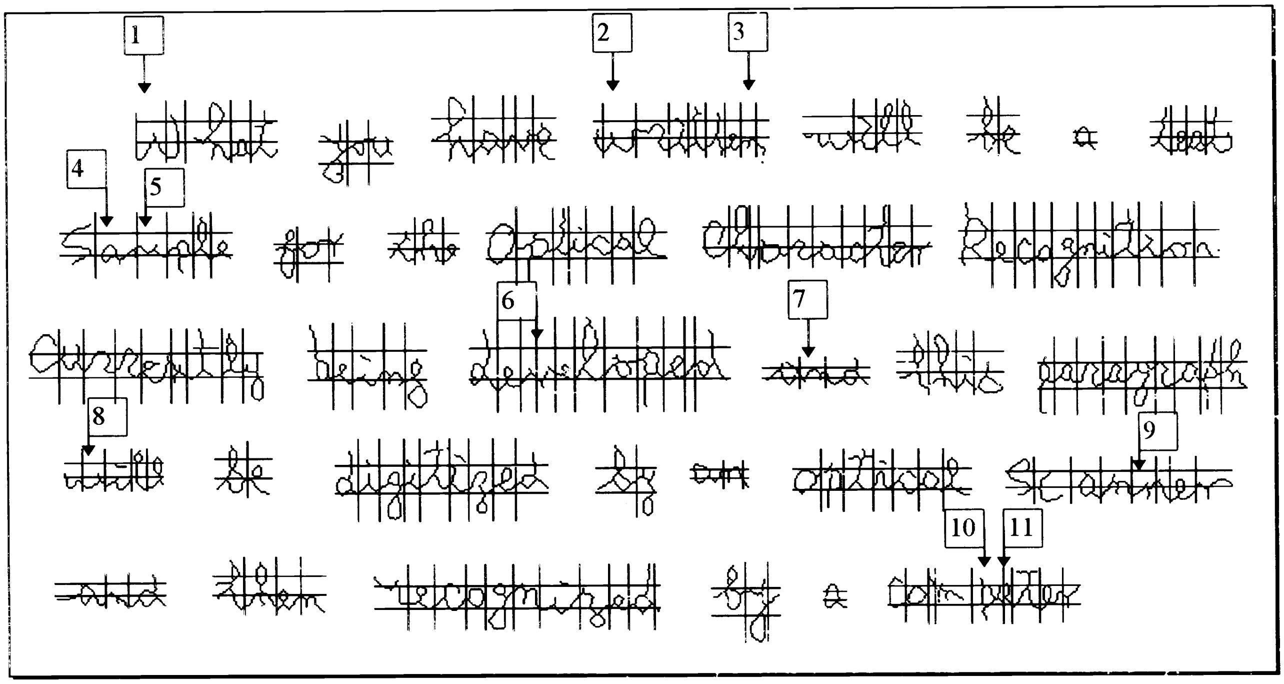


Figure 9 : Segmented Image

When considering the characters between two adjacent segmentation points in figure 9, we can observe three possible cases: correctly segmented characters, wrongly segmented characters and connected segments. The correctly segmented characters are those that the character between the two adjacent segmentation points are correct characters with minimum width. There are 157 of them all together, which gives a segmentation success rate of 93.5%. The connected segments are neither correct nor wrong. They are the result of two possible scenarios. The first is that a segments exists between two correctly segmented characters, for example, the word 'Character' has one such segment between 'h' and 'a'. As the segmentation algorithm aims to segment character with a minimum width out of a word, such segments may arise if the connecting stroke between two characters are too long. The second is that a segment exists between a correct character and a wrong character. For example, the word 'developed' has a segment between 'e' and 'v' where 'v' is segmented incorrectly here. This is due to the incorrect segmentation of 'v' rather than the length of the stroke. There are 7 of these segments altogether.

The 11 characters that have been segmented wrongly are highlighted by vertical arrows with a number tag. These characters are either having one of the two adjacent points in the wrong location or an extra segmentation point dividing the character into two. For example, the word 'computer' has a segmentation point at an incorrect location between 'p' and 'u' (errors 10,11), where both characters are segmented wrongly in this case. However, in the word 'Scanner', the second 'n' is segmented incorrectly because of an extra segmentation point being placed between two correct segmentation points (error 8). There are also cases where the wrong segmentation point only caused one character error, rather than two (error 7).

The incorrect characters are 'W', 'w', 'n', 'a', 'm', 'v', 'n', 'w', 'n', 'p' and 'u'. Out of these characters, we discovered that the errors in both 'a' and 'p' are due to their neighboring characters ('m' & 'u') as they are correctly segmented in other words. Among the other errors, 'n' appears three times, 'w' twice and 'W', 'm', 'v' and 'u' all once. Further analysis shows that 'w' and 'W' are consistently wrong, where 'n' is 3 out of 11 occurrences are wrong, and 'm', 'v' and 'u' are all 1 out of 2 occurrences is wrong. From these results, we confirm that there are difficult characters that can easily caused segmentation errors in the process. They are 'w' and 'W', which can be segmented as 'u' & 'i', and 'V' & 'I'; 'm' being segmented as 'r' & 'n', 'n' being segmented as 'r' & 'i'; and both 'u' and 'v' being segmented as two 'i's. From these results, we would expect the difficult character list in general consists of 10 characters including 'w', 'm', 'n', 'u', 'v', 'W', 'M', 'N', 'U', and 'V'. Fortunately, the occurrences of some of these characters are not frequent, and therefore, would expect the segmentation success rate can be maintained over 90%. However, further development focusing on segmenting these characters correctly will certainly be able to bring the success rate even higher.

5. CONCLUSION

In conclusion, a new character segmentation algorithm has been successfully designed, developed, implemented and tested. For a randomly chosen sample of 168 characters written in a normal way, the algorithm is able to correctly segment 157 characters, giving a 93.5% of segmentation success rate. The algorithm is developed with clear goals in mind, and the results show that these goals are fulfilled. In addition, the algorithm has a number of merits. Firstly, it is based on three very simple ideas: geometric classes, X-axis projection and Y-axis projection. Its implementation was straight-forward. Secondly, algorithm emphasizes on segmenting the characters with minimum width rather than simply finding a separation point between two characters. In doing so, the resulting segmentation points denote minimum width characters which will make the subsequent character recognition easier and more successful. Thirdly, the algorithm is capable of segmenting most of the upper-case and lower-case characters except 10 difficult characters. As these characters such as 'w' and 'v' do not have high occurrent rate in English text, the actual rejection rate is expected to be low. Even for these difficult characters, the mistakes made by the algorithm is more likely to be 30-50% of the character's total number of occurrence, perhaps except for the cases of 'W' and 'w' where they have been segmented wrongly consistently. The major drawback of the algorithm is its inability to segment the 10 difficult characters as mention in Section 4 correctly all the times. This essentially accounts for the 6.5% of rejection in our test. If future research can be focused on tackling these difficult characters, and yet retained its ability to segment the other characters correctly, the segmentation success rate of the algorithm is expected to be even higher. At present, detailed performance analysis of the algorithm over all the 90 handwritten samples will also provide a more in-depth knowledge of the algorithm performance.

6. REFERENCES

1. E. J. Bellegarda, J. R. Bellegarda, D. Nahamoo & K. S. Nathan, "A fast statistical mixture algorithm for on-line handwriting recognition", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 16, No.12, Dec 1994, pp.1227-1233..
2. E. Cohen, J. J. Hull & S. N. Srihari, "Control structure for interpreting handwritten addresses", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 16, No.10, Oct 1994, pp.1049-1055.

3. T. S. Huang & C. Y. Suen, "Combination of multiple experts for the recognition of unconstrained handwritten numerals", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 17, No.1, Jan 1995, pp.90-94.
4. R. Sabourin, R. Plamondon & G Lorette, "Off-line identification with handwritten signature images: survey and perspectives", in Structured Document Image Analysis, ed. H. S. Baird, H. Bunke & K. Yamamoto, Springer-Verlag, 1992, pp.219-234.
5. I. S. I. Abuhaiba, S. A. Mahmoud & R. J. Green, "Recognition of handwritten cursive arabic characters", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 16, No.6, June 1994, pp.664-672.
6. P. T. Wright, "On-line recognition of handwriting", GEC Journal of Research, Vol.8, No.1, 1990, pp.42-48.
7. S. Tsujimoto & H. Asada, "Resolving ambiguity in segmenting touching characters", in Structured Document Image Analysis, ed. H. S. Baird, H. Bunke & K. Yamamoto, Springer-Verlag, 1992, pp.203-215.
8. H. S. Baird, S. Kahan & T. Pavlidis, "Components of an omnifont page reader", Proc. of 8th IEEE ICPR, 1986, pp.344-348.
9. M. Ammar, Y. Yoshida and T. Fukumura, "A new effective approach for off-line verification of signatures by using pressure features", Proc. of IEEE 8th International Conference on Pattern Recognition, 1986, pp.566-569.