A NEW SEGMENTATION TECHNIQUE FOR MULTI FONT FARSI/ARABIC TEXTS

M. Omidyeganeh[•], *K. Nayebi*[•], *R. Azmi*[••] *and A. Javadtalab*[•••]

[•] Department of Electrical Engineering, Sharif University of Technology, Tehran, Iran. momid@mehr.sharif.edu , knayebi@sharif.edu.

[••]Department of Computer Engineering, Azzahra University,Tehran,Iran, razmi@alzahra.ac.ir.

[•••] Department of computer engineering Sharif University of Technology, Tehran, Iran. javadtalab@ce.sharif.edu

ABSTRACT

Segmentation is a very important stage of Farsi/Arabic character recognition systems. A new segmentation algorithm -for multi font Farsi/Arabic texts- based on the conditional labeling of the up contour and down contour is presented. A pre-processing technique is used to adjust the local base line for each subword. This algorithm uses adaptive base line for each subword to improve the segmentation results. This segmentation algorithm, in addition to up and down contours, takes advantage of their curvatures also. The algorithm was tested on a data set of printed Farsi texts, containing 22236 characters, in 18 different fonts. 97% of characters were correctly segmented.

1. INTRODUCTION

Optical character recognition is an attractive branch of image processing with many applications in man-machine interface and document processing. Intensive research in this area has also resulted in commercial systems [13]. However, Farsi/Arabic texts have some properties that make them difficult to recognize. Farsi/Arabic texts are cursive and are written from right to left .A Farsi/Arabic character might have several shapes -from 1 to 4 shapes- depending on its relative position in the word. In addition, some Farsi/Arabic characters have the same shape and differ from each other only in some dots or zigzag bars. Each word, machine-printed or handwritten, may consist of several separated subwords. A subword is either a single character or a set of connected characters. Although, seven Farsi characters out of 32 do not join to their left neighbors, others join to the neighboring characters to make a word or a subword. The neighboring characters, separated or connected, may overlap vertically. Some of these characteristics of Farsi/Arabic script are shown in Figure 1.

There are several papers published on the recognition of Arabic and Farsi texts e.g. [1,2,3,4,5,7,10,13,14]. The main problem is Farsi/Arabic text is its segmentation. There are two main approaches to word recognition: segmentation-based and segmentation-free [8,9,11]. Because of the mentioned characteristics of Farsi/Arabic text, a hybrid approach for Farsi text recognition seems more promising [2]. This paper concerns the first approach, where each word or subword is first split into a set of single characters. The word is then recognized by the sequence of its characters.

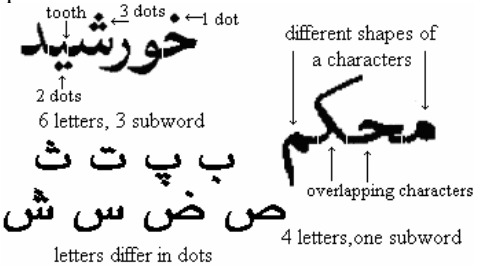

Figure 1.Some characteristics of Farsi/Arabic script

Different character segmentation techniques for the printed Farsi/Arabic words have been proposed [14, 8, 10 and 3]. At 2001, R. Azmi proposed a new technique for omnifont Farsi text segmentation [5]. His segmentation algorithm was based on the conditional labeling of the up contour. In this paper, we present a new algorithm for segmentation of multi font Farsi/Arabic texts, which uses the idea of applying conditional labeling rules similar to R. Azmi, on both the up contour and the down contour of the subword. This segmentation algorithm also uses up contour curvature and adaptive base line for each subword. This technique is not sensitive to overlapping characters and slant. The paper is organized in five sections; Section 2 describes the pre-processing stage, including the base line detection and its local adjustment. In Section 3, the proposed segmentation algorithm is explained. The experimental results are presented in Section 4. Finally, the conclusion is given in Section 5.

2. PRE-PROCESSING

We prepare a document for 6 fonts. The documents are scanned with a resolution of 300 dpi and are stored as binary images. In this resolution, the pen size for a normal printed text with sizes between 14 to 16 is about 4 to 6 pixels depending on fonts. The text lines and their words/subwords are segmented by finding the valleys of the horizontal and vertical projection profiles. To

calculate the pen size, a text line is scanned column by column. The most frequent pen thickness in these columns is adopted as the pen size, ***w***. At the next step, the global base line is defined as a horizontal line, all across a text line whose width is equal to ***w***, and covers the maximum number of black pixels in that text line. Each subword is combination of some regions. One or more of these regions are bodies; others are points, zigzag bars, etc.

Next, we define main body/bodies of each subword. If a region overlaps with the base line in some pixels, it is considered to be the body (Fig. 2). To find the pen size more precisely, we calculate pen size just for bodies of each subword again. The line of bodies is scanned column by column. In this case, the most frequent thickness of the black-pixels in these columns is adopted as the new pen size, ***w.*** Contour of each subword is extracted using a convolution kernel with Lapacian edge detection method. By moving from right top black pixel to left down black pixel clockwise through the contour, up contour is extracted. Down contour is extracted by moving from left down black pixel to right up black one clockwise through contour. For locating the base line accurately, a technique is used that locally adjusts the base line. If line is completely written (for A4 page); the line is segmented in to five sections, otherwise we will use suitable sections of line. The up and down contours of subwords of the determined length of line, traced in CCW, are represented by the 8-directional Freeman code. Within a distance of ***w/2*** around the upper edge of the global base line, the row of the up contour image having the maximum instances of the code 4, say ***n4***, is considered as the upper bound of the local base line, ***iup***. The lower bound, ***idown***, is found in a similar way, searching for a row with maximum instances of the code 0 in the image of down contour image, say ***n0***. If the width of the resulting local base line is greater than ***1.25w***, then if ***n4 > n0***, the ***iup*** is retained and the ***idown*** is shifted upward, so that the width of the base line becomes ***w***. Otherwise, if ***n4 <=n0***, the ***iup*** is shifted downward in the same way.

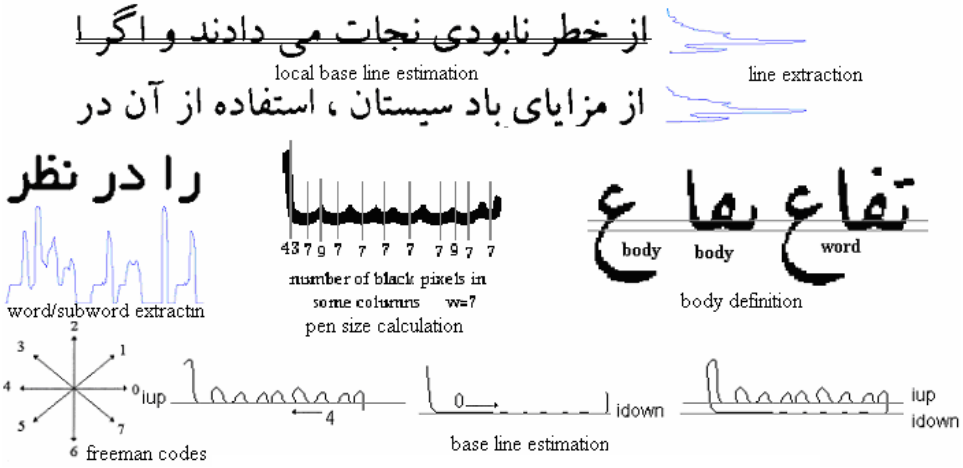

Figure 2.Pre-processing

3. SEGMENTATION ALGORITHM

3.1. Contour labeling

This step of the segmentation technique is based on the conditional labeling of the up contour and down contour of each subword (Fig. 3). Tracing the up contour from right to left in CCW direction, each point is labeled depending on its distance from the base line and the label of its preceding point. These labels are 1, 0 and -1 standing for up, middle and down, respectively. The labeling process is shown in Fig. 5(a) in the form of a state diagram. The label of the first point of a contour is always up. Figure 4(a) shows a sample word and it's labeled up contour. The neighboring points having the same label make a path. If a path is shorter than ***(w/2+1)***, it is linked to the preceding path. Since in some cases the curves and bends are only in up contour or down contour of subwords, in our algorithm, we also label down contours. Labeling procedure for down contours is the same as that of the up contours. The state diagram of this procedure is shown in fig. 5(b). In this way, as shown in figure 4(b), the up contour and down contour are represented by strings of the labeled paths –both from right to left.

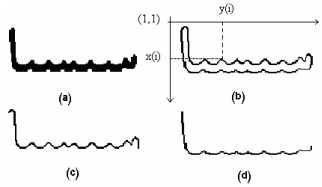

Figure 3.(a) Body of word (b) its contour (c) up contour (d) down contour

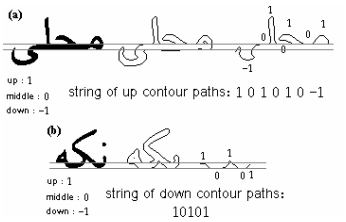

Figure 4. (a)A word, its contour, and its labeled up contour . (b) A word, its contour and labeled down contour

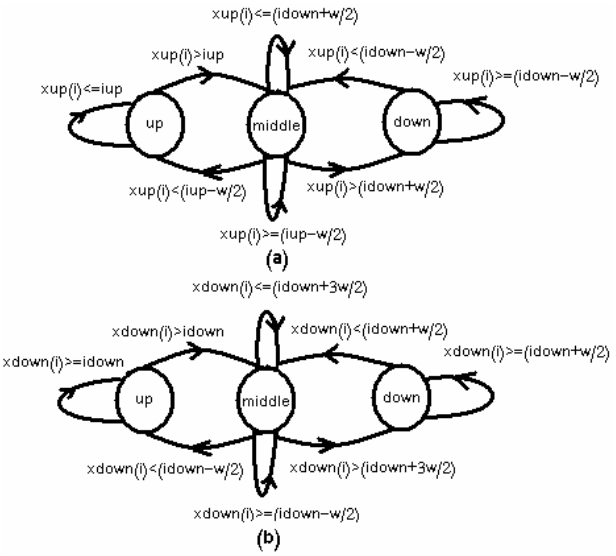

Figure 5.State diagram of (a)up contour (b) down contour labeling process

3.2. Contour curvature grouping

Using contour curvature of subwords will improve the segmentation results. Specifically soft bends in subwords are hard to determine with labels introduced in 3.1. So we add another step to our algorithm. This step works like in 3.1. Up contour and down contour of the subword, traced CCW, are represented by the eight-directional Freeman code. Numbers from 0 to 7 are the names of these kinds of grouping. The neighboring points having the same number make a group. If the length of a group is shorter than ***w/2***, it is linked to the preceding path. So the codes are smoothed somehow.

3.3. Character segmentation

A potential segmentation point is defined as follows:

1. All segmentation points must be ***1.5w*** apart from left and ***2w*** apart from right end of the subword.
2. All segmentation points must be around base line.(within w/2 of it)

3.3.1. For contour groups 0-7 (freeman code)

Here we use the difference between two adjacent groups and the paths (up, median, down) to determine the segmentation point. If the previous path is a 1 (up) path longer than ***w***, and the point in up contour is in a group with number 2, 3 or 4; and the point in down contour with the same column, is in a group with number 6 or 7, the point is segmentation point.

3.3.2.For contour labeled by up, median and down labels

For both the up contour and the down contour, If the 0 path (median) is longer than ***w*** and:

The previous path and the next path is 1 path and the next path is longer than ***1.5w***; or the next path is -1 and its length is more than ***2.5w***; or the next path is -1 path, longer than ***4w*** and the last path.

3.3. Adaptive local base line

Thereafter, we divide the length of subword by the number of segmentation points, and compare the result ***r*** with a threshold ***t***. If ***r*** is less than ***t*** the local line will vary and the procedure will repeat. Local line will go up and down for definite times depending on ***w***. This step is useful, especially when our base line is not accurate. Here we will take care of wrong segmentation points so that they do not affect our results. The threshold is determined statistically. It is worth mentioning that this segmentation algorithm is not sensitive to slant and overlapping characters. The segmentation algorithms that are based on the vertical histogram or upper profile of the words have severe problems with overlapping characters. This algorithm uses the contours and therefore tolerates any degree of overlapping between the characters. An example to show the strength of this technique, compared with the other techniques, is shown in Fig. 6.

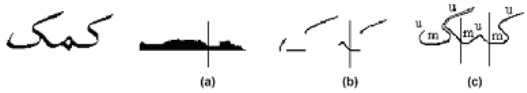

Figure 6.The effect of slant and overlapping characters in: (a) histogram-based methods; (b) profile-based methods d (c) contour-based method.[5]

3.4. Post-processing

The segmentation algorithm tends to over-segment the characters, i.e. certain characters are split into sub-characters. There are three main reasons for this. First, the algorithm is tuned as not to miss any segmentation point, if possible. Second, there are certain simple shapes in the body of some characters that resemble other characters. Third, our algorithm is for multi font cases and in fact, it is very difficult to find a procedure suitable for all fonts. It is possible to leave these errors to be resolved in the recognition stage. However, the errors that occur frequently can be detected and corrected, as described here. Sometimes more than one segmentation point is determined instead of one. In this case points nearer than ***w/2*** is gathered to a one point. Some

characters, when occurring at the end of a subword, may have a u path that causes a false segment. Some other characters have a similar u path that produces a correct segment. The second group of character is detectable by their height or loop. Therefore, the false segment is recognized and connected to its right neighbor (fig.7). Using dots and their information such as position, number, etc will be useful, too. [1]

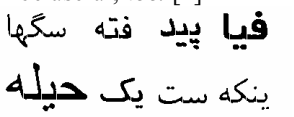

Figure 7.Segmentation examples

4. EXPERIMENTAL RESULTS

The segmentation algorithm was tested on a set of printed texts in 18 different fonts (Fig. 8). The test set includes 22236 characters. The training samples are not included in the test set. Results are shown below. (Table 1)

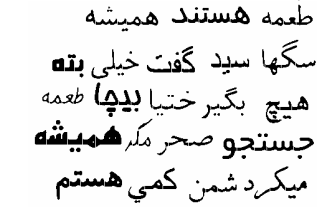

Figure 8. .Sample words of different fonts used in test set

Table 1-segmentation results

Font	Correct Segmentation Rate(%)
traffic	99.92%
Yekan	99.76%
Homa	99.68%
Roya	99.52%
Simplified Arabic	99.44%
Arabic Transparent	99.28%
Yagut	99.20%
Elham	99.03%
Times New Roman	98.80%
Mitra	97.91%
Nazanin	97.11%
Kamran	96.79%
Esfehan	96.15%
Davat	94.79%
Koodak	94.40%
Arshia	92.30%
Zar	90.62%
Lotus	90.46%
Total	**96.95%**

5. CONCLUSION

In this paper, a character segmentation algorithm was proposed for multi font Farsi/Arabic text. A correct segmentation rate of about 97% for 18 fonts was achieved. The algorithm is tolerant to slant (below .1719 degree), and to some extent to the misalignment of the local base lines. The segmentation errors were mainly due to skewed text lines. Using a pre-processing step will improve the results.

6. REFERENCES

[1]M. Altuwaijri, and M. Bayoumi, (1994), Arabic text recognition using Neural Networks, *Proc. Int. Symp. on Circuits and Systems – ISCAS*, 1994, pp. 415 – 418.

[2]H. Al-Muallim and S. Yamaguchi, "A method of recognition of Arabic Cursive Handwriting", *IEEE Trans. Pattern Anal. Mach. Intell.* PAMI – 9, 1987,pp 715-722

[3]A. Amin, "Off-line Arabic character recognition: the state of the art". *Pattern Recognition 31*, pp. 517-530. 1998.

[4]A. Amin and G. Masini, "Machine Recognition of Multifont printed Arabic Texts", *Proc. 8th Int. Conf. on Pattern Recognition*, Paris, 1986, pp 392-295.

[5]R. Azmi and E. Kabir, "A New Segmentation Technique for Omnifont Farsi Text", *Pattern Recognition Letters 22*, pp. 97-104, 2001.

[6]R. Azmi, "Recognition of omnifont printed Farsi text". *Ph.D. Thesis, Tarbiat Modarres University*, Tehran,1999.

[7]R. Azmi and E. Kabir, "A recognition algorithm for hand printed Farsi characters". *Proceedings of the International Conference on Telecommunication,* ICT '96, Istanbul, pp. 852-855, 1996.

[8]T.S. El-Sheikh and R.M. Guindi, "Computer recognition of Arabic cursive scripts." Pattern Recognition 21, pp. 293-302, 1988.

[9]J.J Hull and S.N. Srihari, "A computational approach to visual word recognition: hypothesis generation and testing", *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, CVPR'86, Washington, DC*, pp. 156-161, 1986.

[10]B.M. Kurdy and A. Joukhadar, "Multi font recognition system for Arabic characters". *Proceedings of the Third International Conference and Exhibition on Multi-Lingual Computing, Durham*, pp. 731-739, 1992.

[11]Y. Lu and M. Shridhar, "Character segmentation in handwritten words - an overview". *Pattern Recognition 29*,pp. 77-96, 1996.

[12]K. Massruri and E. Kabir, "Recognition of hand-printed Farsi characters by a Fuzzy classifier". *Proceedings of the Second Asian Conference Computer Vision, ACCV '95*, Singapore, Vol. 2, pp. 607-610, 1995.

[13] S. Mori, C.Y. Suen and K. Yamamoto, "Historical review of OCR research and development". Proc. IEEE 80, pp. 1029-1058,1992.

[14]B. Parhami and M. Taraghi, "Automatic recognition of printed Farsi text", *Pattern Recognition 14*, pp. 395-403,1981.