ELSEVIER

Pattern Recognition Letters 22 (2001) 97–104

Pattern Recognition Letters

www.elsevier.nl/locate/patrec

A new segmentation technique for omnifont Farsi text ☆

R. Azmi [1], E. Kabir *

Department of Electrical Engineering, Tarbiat Modarres University, P.O. Box 14115-143, Tehran, Iran

Received 20 January 1999; received in revised form 24 January 2000

Abstract

A new segmentation algorithm based on the conditional labeling of the upper contour is presented. A pre-processing technique is proposed that adjusts the local base line for each subword. The algorithm was tested on a data set of printed Farsi texts in 20 fonts. 98.5% of the connected characters were correctly segmented. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Farsi text; Omnifont; Character segmentation; Contour tracing; Chain code

1. Introduction

Optical character recognition is an attractive branch of pattern recognition with many applications in man–machine interface and document processing. Intensive research has been done and commercial systems are now available (Mori et al., 1992). However, there are a few works reported on the recognition of Arabic and Farsi texts (e.g., Amin, 1998; Parhami and Taraghi, 1981; Massruri and Kabir, 1995; Azmi and Kabir, 1996).

Farsi is written from left to right. 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. Each character may take up to four different shapes, depending on its position in the subword. The neighboring characters, separated or connected, may overlap vertically. There are similar characters that only differ in their dots. These characteristics of Farsi script are shown in Fig. 1.

There are two main approaches to word recognition: segmentation-based or bottom–up and segmentation-free or top–down (e.g., Lu and Shridhar, 1996; Hull and Srihari, 1986). Due to the above mentioned characteristics, a hybrid approach for Farsi text recognition seems more promising (Azmi, 1999). This paper concerns the first approach, where each word or subword is first split into a set of single characters and then is recognized by its individual characters.

Different character segmentation techniques for the printed Arabic or Farsi words have been proposed; the basic ideas behind them are addressed here.

☆ This work was supported in part by the Scientific Research Council of Iran.

* Corresponding author. Fax: +98-21-800-5040.

E-mail address: kabir@modares.ac.ir (E. Kabir).

[1] Present address: Department of Computer Engineering, Azzahra University, Tehran, Iran.

PII: S0167-8655(00)00086-6

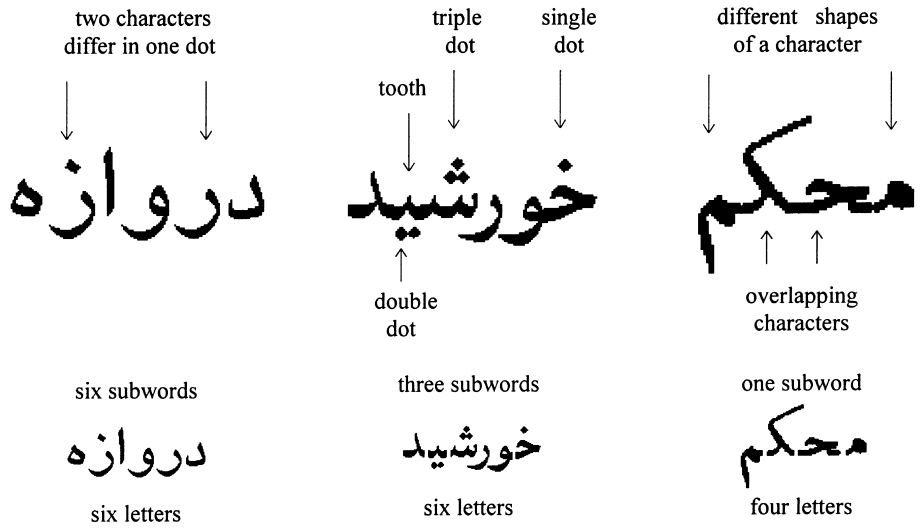


Fig. 1. Some features of Farsi script.

In the very first published work on Farsi OCR (Parhami and Taraghi, 1981), the basic rule for character segmentation is as follows. The binary image of the subword is searched column by column. If a column has a single segment of black pixels and the size of that segment is close to the pen size, that column is a potential segmentation column.

In another work (El-Sheikh and Guindi, 1988), the outer contour of the subword, excluding its dots, if any, is used for the segmentation. The contour height for each column is defined as the vertical distance between the two extreme points of the contour in that column. A window is moved horizontally across the contour image of the subword and the average height within that window is calculated. As a basic rule, an average height less than a preset threshold shows a potential segmentation point.

Conditional labeling of the upper profile of the subword is the subject of another segmentation technique (Kurdy and Joukhadar, 1992). Each pixel of the upper profile is labeled as up, middle or down, based on its distance from the base line and also the label of the previous pixel. By applying a simple set of rules on the labeled profile, the potential segmentation points are found.

In another work, the vertical density histogram of the black pixels is used (Amin, 1998). Any column that its value in the histogram is less than the average value of the histogram is a potential segmentation column.

In this paper, we present a new algorithm for the segmentation of omnifont Farsi text, which uses the idea of applying conditional labeling rules similar to Kurdy and Joukhadar (1992), but on the upper contour of the subword instead of its upper profile. Unlike the above mentioned methods, our technique is not sensitive to the overlapping characters and slant. Fig. 2 illustrates the proposed system.

The paper is organized in six 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. Section 4 describes the post-processing stage. The experimental results are presented in Section 5. Finally, the conclusion is given in Section 6.

2. Pre-processing

The document is scanned with a resolution of 200 dpi and is stored as a binary image. Having this resolution, the pen size for a normal printed text is about 5 pixels. However, for some less

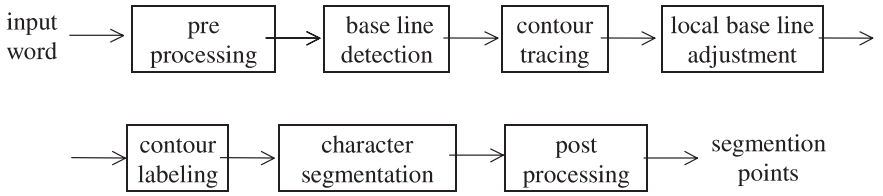


Fig. 2. The proposed character segmentation system.

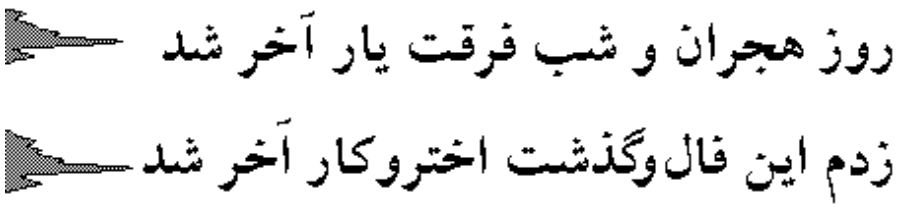
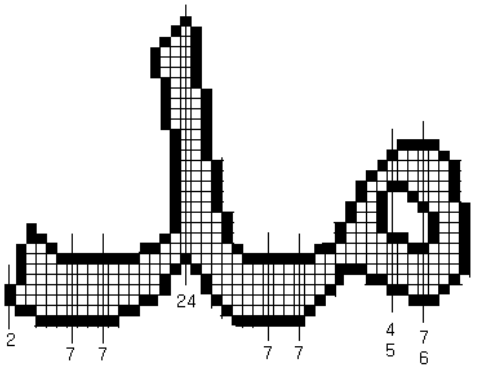


Fig. 3. A sample of Farsi text segmented into two text lines.

common fonts, it reduces to 3 pixels. The text lines are segmented by finding the valleys of the horizontal projection profile (Fig. 3).

2.1. Pen size calculation

To find the pen size, a text line is scanned column by column. The most frequent size of the black-pixel runs in these columns is adopted as the pen size, *ps* (Fig. 4).



The sizes of black-pixel runs in some columns
ps=7

Fig. 4. Computing the pen size.

2.2. Global base line detection

The global base line is defined as a horizontal line, all across a text line, that its width is equal to the *ps*, and covers the maximum number of black pixels in that text line (Fig. 5).

2.3. Local base line adjustment

Accuracy in locating the base line plays an important role in the character segmentation, in particular if the pen size is small. A novel technique is introduced here that locally adjusts the base line for each subword, as follows.

The contour of the subword, traced in CCW, is represented by the eight-directional Freeman code. Within a distance of $ps/2$ around the upper edge of the global base line, the row of the image having the maximum instances of the code 4, say n_4, is considered as the upper bound of the local base line, *ubl* (Fig. 6). The lower bound, *lbl*, is found in a similar way, searching for a row with maximum instances of the code 0, say n_0, around the lower edge of the global base line.

If the width of the resulting local base line is greater than $1.25ps$, then if $n_4 > n_0$, the *ubl* is retained and the *lbl* is shifted upward, so that the width of the base line becomes *ps*. Otherwise, if

Fig. 5. Global base line detection.

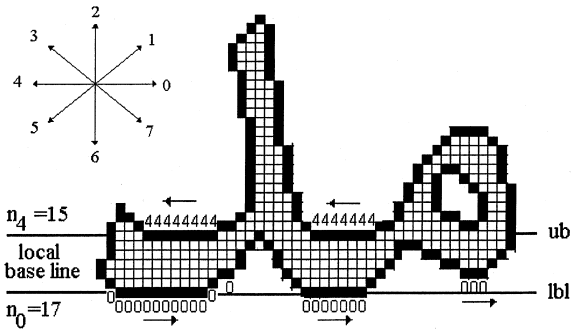


Fig. 6. Local adjustment of the base line.

$n_4 \leqslant n_0$, the *ubl* is shifted downward in the same way.

By local adjustment of the base line, the performance of the segmentation algorithm improves. This is more clear in the old printed texts where the words are not aligned properly, as shown in Fig. 7, and also in the texts with skewed lines or low resolution.

3. Segmentation algorithm

3.1. Contour labeling

The proposed segmentation technique is based on the conditional labeling of the upper contour of each subword (Fig. 8). Tracing the contour from right to left in CCW, each point is labeled depending on its distance from the base line and the label of its preceding point. These labels are *u*, *m* and *d* standing for up, middle and down, respectively. The labeling process is shown in Fig. 9 in the form of a state diagram. The label of the first point of a contour is always *u*. Fig. 10 shows a sample word and its labeled contour.

The neighboring points having the same label make a path. If a path is shorter than $ps/(2+1)$, it is linked to the preceding path. In this way, as shown in Fig. 10, the upper contour is represented by a string of the labeled paths from right to left, the direction that Farsi is written.

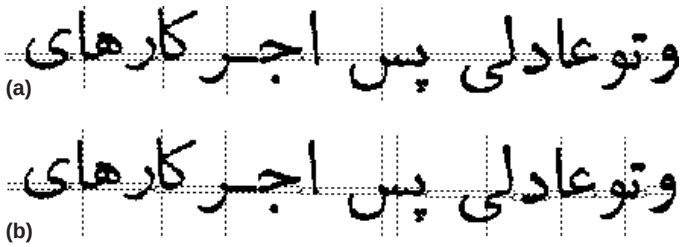


Fig. 7. Segmentation examples. (a) Using the global base line; (b) using the local base lines.

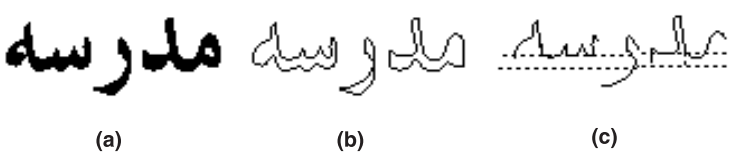


Fig. 8. (a) A word, (b) its contour and (c) upper contour.

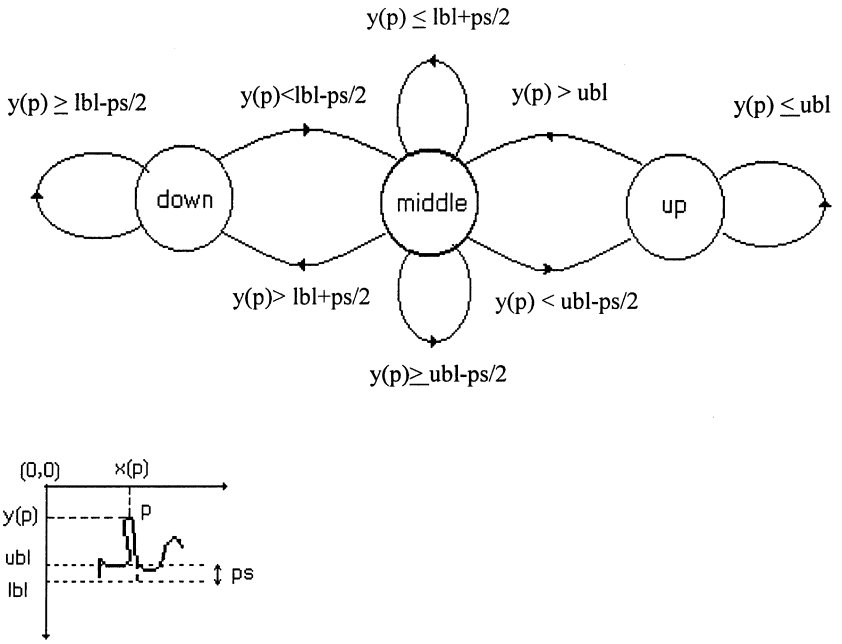
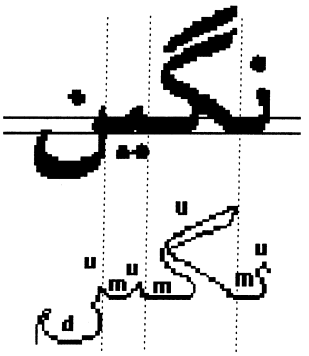


Fig. 9. Conditional labeling of the contour points.



string of labeled paths = *u m u m u m u d*

Fig. 10. A word, its labeled contour and segmentation points.

3.2. Character segmentation

A potential segmentation point is defined as the last point of an *m* path that satisfies the following conditions:

(a) The *m* path is longer than $ps/(2+1)$.
(b) The previous path is a *u* path longer than *ps*.
(c) The next path is longer than a threshold value. This threshold is 1.5*ps* for a *u* path and 4*ps* for a *d* path.

A word segmented by applying these rules, is shown in Fig. 10.

It is worth mentioning that the proposed 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 word have severe problems with overlapping characters. The algorithm proposed here, uses the upper contour and therefore tolerates any degree of overlapping between the characters. The strength of our technique, compared with the other techniques, is shown in Fig. 11.

4. Post-processing

As shown in Fig. 12, the segmentation algorithm tends to over-segment the characters, i.e., certain characters are split into sub-characters. There are two main reasons for this. First, the algorithm is so 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. Although it is possible to leave these errors to be solved in the

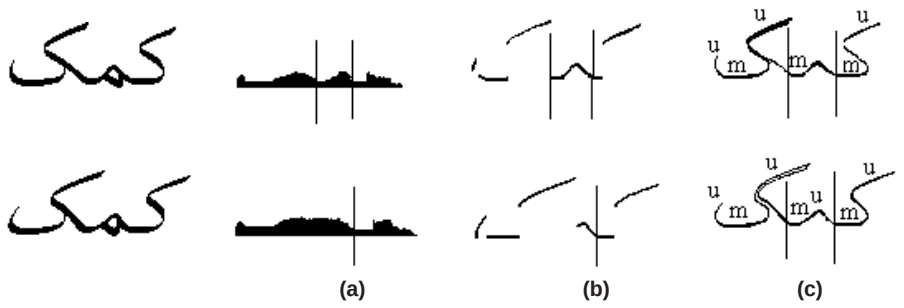


Fig. 11. The effect of slant and overlapping characters in: (a) histogram-based methods; (b) profile-based methods and (c) the proposed contour-based method.

	before post-processing	After post-processing
a	مقاومت تکنیک	مقاومت تکنیک
b	کمند	کمند
c	کم	کم
d	کشور س	کشور س
e	فصل نص	فصل نص

Fig. 12. Post-processing examples.

g1	ب پ ت ث ک گ
g2	ه ها
g3	د ذ
g4	م
g5	سـ شـ
g6	س ش
g7	بـ پـ تـ ثـ نـ یـ

Fig. 13. Character groups g1 to g7.

recognition stage, it is more desirable to correct them in a post-processing stage. In this stage, the errors that occur frequently are detected and corrected, as follows. Note that the character groups referred in this Section, g1 to g7, are shown in Fig. 13.

(a) The characters in g1, when occuring at the end of a subword, may have a *u* path that causes a false segment (Fig. 12(a)). Only the characters in g2 have a similar *u* path that produces a correct segment. The first character is detectable by its height and the second by its loop. Therefore, the false segment is recognized and connected to its right neighbor.

(b) In some fonts, a false segment is made by a *u* path at the end of the characters in g3. It is detectable by its small width (Fig. 12(b)).

(c) The character in g4 is divided into two segments in some fonts. The left segment is detectable by its label and height (Fig. 12(c)).

(d) In some fonts, the characters in g5 are split into two or three segments. If three of the characters in g7 join together, their bodies resemble the characters in g5. To discriminate these characters it is necessary to locate the single,

Fig. 14. Sample words of different fonts.

double and triple dots and also to distinguish small teeth from noise (Fig. 1).

If h and w represent the height and width of a dot mark, the following rules are used.

- If $0.8 \leqslant h/w < 1.3$ and $0.7 \leqslant w/ps < 1.6$, then the mark is a single dot.
- If $0.3 \leqslant h/w < 0.7$ and $1.5 \leqslant w/ps < 3.5$, then the mark is a double dot.
- If $0.7 \leqslant h/w < 1.3$ and $1.5 \leqslant w/ps < 3.5$, then the mark is a triple dot.
- If the distance between two single dots is less than $ps/(2+1)$, then they make a double dot.
- If a single dot is above or below of a double dot, they make a triple dot.
- If there is a u path in the upper contour that is shorter than $1.5ps$, the lower contour is examined. The u path is taken as a tooth, if there is a notch in the lower contour right below it (Fig. 1).

After locating the dots and teeth, the segmentation errors are corrected as follows:

- If at the beginning of a subword, after a u path, there are two teeth with no dot or only a triple dot above them, the teeth are connected together to make the characters in g5. The same is true for three teeth in the middle of a subword (Fig. 12(d)).
- If at the end of a subword, there are two teeth with no dot or only a triple dot above them, followed by a u path and a d path longer than $3ps$, they are connected together to make the characters in g6 (Fig. 12(d)).
- If there is a single tooth with no dot above or below it, it is connected to its right neighbor (Fig. 12(e)).

5. Experimental results

The segmentation algorithm was tested on a set of printed texts in 20 different fonts (Fig. 14). The test set includes 11,347 characters, 8056 of them connected. The training samples are not included in the test set. Table 1 shows a summary of the

Table 1
Segmentation results

	Correct segmentation rate	
	Before post-processing (%)	After post-processing (%)
Connected characters	91	98.5
All characters	93	98.9

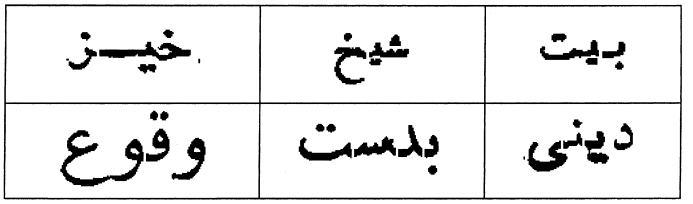


Fig. 15. Typical errors in segmentation.

results, before and after the post-processing. Some typical examples of incorrect segmentation are shown in Fig. 15.

6. Conclusion

In this paper, a character segmentation algorithm was proposed for omnifont Farsi text. A correct segmentation rate of about 99% was achieved. The algorithm is tolerant to the slant and to some extent to the misalignment of the local base lines. The segmentation errors were mainly due to the low scanning resolution and skewed text lines.

References

Amin, A., 1998. Off-line Arabic character recognition: the state of the art. Pattern Recognition 31, 517–530.

Azmi, R., Kabir, E., 1996. A recognition algorithm for hand-printed Farsi characters. In: Proceedings of the International Conference on Telecommunication, ICT '96, Istanbul, pp. 852–855.

Azmi, R., 1999. Recognition of omnifont printed Farsi text. Ph.D. Thesis, Tarbiat Modarres University, Tehran.

El-Sheikh, T.S., Guindi, R.M., 1988. Computer recognition of Arabic cursive scripts. Pattern Recognition 21, 293–302.

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

Kurdy, B.M., Joukhadar, A., 1992. Multifont recognition system for Arabic characters. In: Proceedings of the Third International Conference and Exhibition on Multi-Lingual Computing, Durham, pp. 731–739.

Lu, Y., Shridhar, M., 1996. Character segmentation in handwritten words – an overview. Pattern Recognition 29, 77–96.

Massruri, K., Kabir, E., 1995. Recognition of hand-printed Farsi characters by a Fuzzy classifier. In: Proceedings of the Second Asian Conference Computer Vision, ACCV '95, Singapore, Vol. 2, pp. 607–610.

Mori, S., Suen, C.Y., Yamamoto, K., 1992. Historical review of OCR research and development. Proc. IEEE 80, 1029–1058.

Parhami, B., Taraghi, M., 1981. Automatic recognition of printed Farsi text. Pattern Recognition 14, 395–403.