Segmentation of Arabic Handwriting based on both Contour and Skeleton Segmentation

Safwan Wshah, Zhixin Shi and Venu Govindaraju
Department of Computer Science and Engineering
University at Buffalo, Amherst, NY, USA
{srwshah, zshi, govind}@buffalo.edu

Abstract

We propose a new algorithm for segmentation of off-line handwritten Arabic words. The algorithm segments the connected letters to smaller segments each of which contains no more than three letters. Each letter may be segmented to at most five pieces. In addition to improving the recognition of Arabic words, another potential application of the proposed segmentation method is to build lexicon of small size, consisting of no more than three letter combinations. Generally, it is very hard to generate lexicon for recognition of unconstraint handwritten Arabic documents due to the large number of words of Arabic language.

The algorithm has been tested on over 6300 words from 45 different documents written by 18 writers. The system is able to segment more than 93% of the words into segments, each containing at most one letter, 6% of the words into segments that contains two letters and 3% of the words into segments that contains three letters.

1. Introduction

Arabic is written by more than 325 million people in over 20 different countries [1]. The Arabic script evolved from a type of Aramaic, with the earliest known document dating from 512 AD [2]. The Aramaic language has fewer consonants than Arabic, so new letters were created around the 7th century by adding dots to existing letters. There are therefore several letters which differ only by a single dot [2]. Other small marks (diacritics) are used to indicate short vowels, but are often not used. Arabic text, both handwritten and printed, is cursive and the letters are joined together along a writing line [1].

Because the Arabic text has not been changed for a long time and due to the huge number of people using it, it is considered a rich language and contains huge number of words and vocabularies.

The Arabic alphabet has 28 letters (Figure 1), each with two or four shapes. A letter's shape is determined by its position: start, middle, or end of a sub-word, or alone. Arabic text is written from right to left, and adjacent letters are joined except when the sub-word ends with one of the six letters in Figure 1(b). A connected sequence of letters is called a "sub-word" [3].

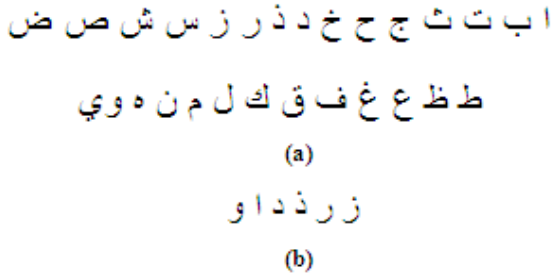

Figure 1: (a) The Arabic alphabet. Transliterated: (from right to left) alif, baa, taa, thaa, jiim, Haa', khaa, daal, Thaal, raa, zaay, siin, shiin, Saad, Daad, Taa, Thaa, ayn, ghayn, faa, qaaf, kaaf, laam, miim, nuun, haa, waaw, yaa. (b) Letters which cannot be joined to a following letter.

The segmentation problem of Arabic handwriting is considered a challengeable task due to different styles of handwriting and the connectivity of the Arabic letters [4]. Until now, there was no robust algorithm that could solve the problem, see [4] for a survey of previous work. A similar problem has been addressed for English handwritten text. Kim [5] developed an algorithm to segment the connected English text where up to four segments can be connected to form a single letter, the algorithm proved very good recognition results in terms of accuracy and speed [5].

978-0-7695-3725-2/09 $25.00 © 2009 IEEE
DOI 10.1109/ICDAR.2009.152

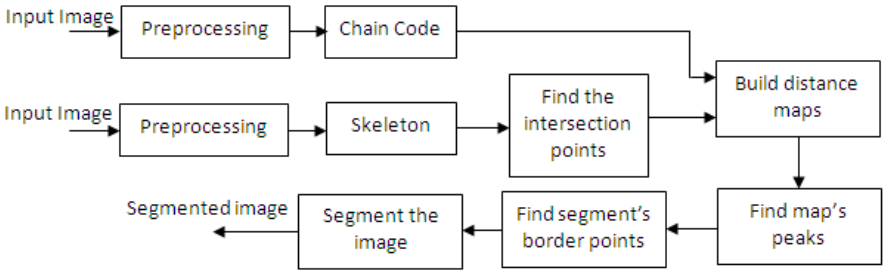

Figure 2: Word segmentation methodology

In most character recognition systems the lexicon size should be small for practical use [6], thus the need for a robust segmentation algorithm is needed to separate the connected letters and make the lexicon size smaller. In this paper we present a new and robust segmentation technique to segment Arabic words to no more than five segments per character, defined as Maximum segments per character (MSC), and any segment contains up to three letters, defined as the Maximum characters per segment (MCS). In this way, we are simplifying the problem and making the size of the lexicon smaller (up to three connected letters). Although we sometimes need to combine up to five segments to form one letter, the search time is significantly reduced by minimizing the size of the lexicon.

2. Our Approach

The objective is to develop a robust segmentation algorithm to be used for systems of handwritten word recognition for real-time applications. Figure 2 illustrates the methodology that has been developed.

The input image goes through steps of pre-processing, chain code generation, and skeleton. The pre-processing step includes noise removal, smoothing [7, 8], and special types of minimum and median filters. Smoothing removes jaggedness of the contours [13], and minimum and median filters are applied with a condition of not removing any pixel considered as the only way to connect two or more blobs. In this manner, we prevent any loss of original data.

The chain code generation step converts the binary image input into a chain code representation by coding the boundary contours of components in the image while preserving the positional and directional information of adjacent pixels [9]. An array is defined for efficient representation and manipulation of data. Single pixel components (speckle noise) are detected and removed. For the Arabic segmentation problem we found that only the exterior contours have useful information. Figure 3 shows examples of chain code applied to handwritten Arabic documents.

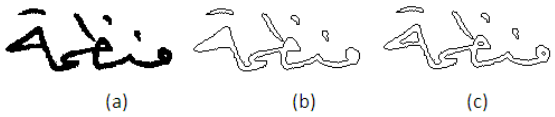

Figure 3: (a) original image (b) chain code exterior contours only (c) chain code exterior and interior contours.

The skeleton algorithms have been proven effective in a broad range of image-processing applications including character recognition [10]. The skeleton algorithm will find one pixel thick representation showing the centerlines of the text. Generally, for a skeletonization algorithm to be effective, it should ideally compress data and retain the significant features of the pattern. The result of an algorithm is dependent on the definition of connectivity [11], in our case we use the algorithm in [12]. For the case of handwritten Arabic it is hard to find a robust and useful skeleton algorithm that retains the significant feature of the pattern due to the variety of handwritten Arabic writing styles. Figure 4 shows examples of skeletonised images from different Arabic patterns.

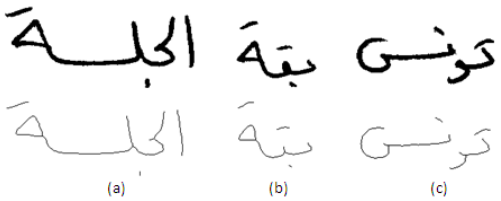

Figure 4: The upper line for orignal image and the second for skeletonized image: (a) Skeleton sample where no data lost; (b) The data lost where the middle letter 'q f' now looks as 't '' in the skeleton image; (c) Another sample of skeleton data lost where the second letter 'w w' now looks as 'r '' in the skeleton image.

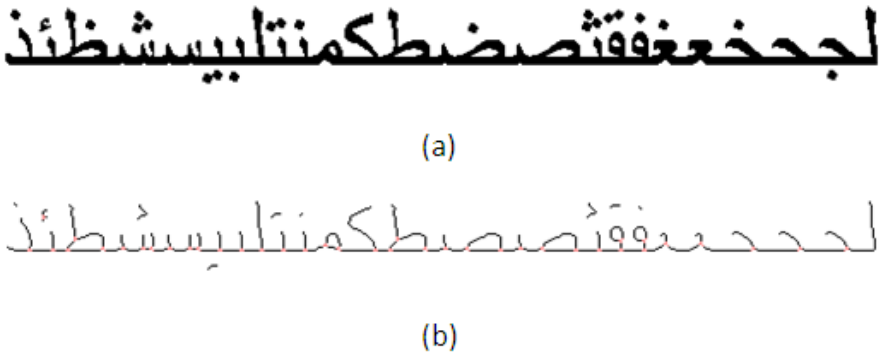

Figure 5: (a) All connected letters combined in one word together; (b) the skeleton algorithm applied to the original image and it shows that for each connected letter it has one or more intersection points that separate it from the previous letter.

The algorithm is based on the notion that every connected character, especially the middle character which represents the connected letters inside the sub-word, has intersection points in its skeleton at the beginning and end of the letter, as shown in Figure 5. The intersection point is defined as the center point of the 3x3 mask with two perpendicular lines passing through it where the length of the first line should be three pixels and the length of the other lines have at least two pixels. This property not only appears in the printed text but also appears in most cases of the handwritten Arabic text.

Because it is hard to find a skeleton algorithm that keeps the features of the handwritten Arabic text (as we shown above), skeleton results cannot be used for recognition. It is necessary to apply the segmentation algorithm on the original image. In order to do this our segmentation algorithm is specially designed to apply both chain code and skeleton results.

As shown in Figure 7b the target is to find the closest path from the chain code contour points to the skeleton intersection point that finally forms the segmentation paths. The developed algorithm is described in Figure 6. Some intersection points will generate two paths and the other will generate three pathse as shown in Figure 7c. To automatically detect the number of segmentation paths, the algorithm described in Figure 6 detects the closest three peaks to the intersection point. If the third peak distance point is close enough to the intersection point, then it is considered the third segment point if, and only if, it is applied to the condition:

$$D_{3,j} < \frac{3 * (D_{1,j} + D_{2,j})}{2} \qquad (2.1)$$

Where Di,j : Distance from the peak(i) point to the intersection point.

INPUT: Skeleton image segments, chain code segments.
1. For each segment in the skeleton image: **a.** For each intersection point in the segment: i. Find the corresponding chain code contour for the current skeleton segment. ii. Build the distance map (between the intersection point and all chain code contour points) as shown in figure(8) starting from the farthest chain code point. iii. Find the first three lower peaks. iv. Apply equation 2.1, to find if the third peak applied for third segment border. v. Draw lines from the intersection point to the applied peak points. **b.** Colorize the new segments.

Figure 6: Algorithm summary of finding the peak segment borders from skeleton and chain code data.

Figure 7: (a) original Image (b) skeleton and chain code results combined together with the intersection points (c) segmented image, note the first intersection point generate two segments and the last point generate three segments.

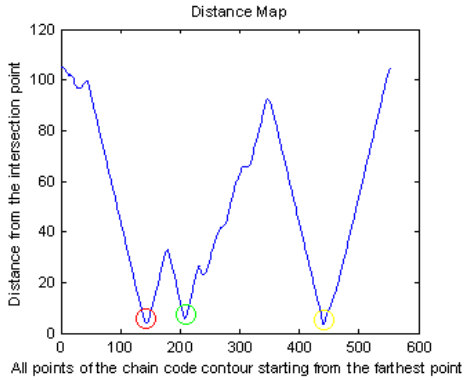

Figure 8: The distance map from the intersection to all contour points generated by the chain code, this Map sample is for the last intersection point of figure 7(b). The yellow peak is the closest peak and the red is the second peak, the green peak is considered peak because its distance apply to equation 2.1.

3. Experimental Results

The algorithm has been applied over 6300 words (28300 letters) from 45 different documents from DARPA and IFTN databases. The table of the numbers of the segments per character (NSC) shows that 21.3% of the segments contain exactly one letter. 90.4% of segments are one letter including 2.5 segments on average; also 9.6% of the segments contain more than one letter.

Based on these results, for any recognition system that uses the proposed algorithm, the best approach is to test a group of segments for a single character up to five connected segments first. If this test failed, then the 2 letters lexicon is applied and otherwise the three letters lexicon.

Figure 9, shows the result of the algorithm applied on one of the used document.

NSC	Percentage %
1/5	7.2
¼	13.6
1/3	20.3
½	28
1	21.3
2	6.3
3	3.3

Table 1.1: Number of segments per letter percentage from the total letters number.

4. Conclusion

Arabic is a rich language with vocabularies, and its letters are connected to form sub-words with a huge number of possibilities in forming these connections. In recognition systems, a lexicon for each group of letters is necessary. However, in practice, the lexicon for 4 characters and more is hard to implement. The presented algorithm segments any word to no more than three letters for any segment and no more than five segments for a single letter. Kim [5] has shown the advantages of this method on the speed and accuracy of the recognition system for the handwritten English text. In this paper we invent a new algorithm for Arabic text that has similar results and also reduces the size of lexicons. The algorithm should improve system robustness and speed due to the segmentation rate of 93% of the words to one letter including three segments on average.

The target is to implement a robust system that is able to segment words into segments where each segment contains one and only one letter of the word. To reach this target, many enhancement algorithms can be add to the implemented algorithm to minimize MSC and MCS. Also, the algorithm needs to be applied on real recognition systems to test for accuracy, speed and lexicon reduction ability.

Acknowledgments: This material is based upon work supported by the Defense Advanced Research Projects Agency DARPA/IPTO (PLATO: A System for Taming MADCAT: Multilingual Automatic Document Classification Analysis and Translation) ARPA Order No. X103 Program Code No. 7M30 Issued through a subcontract from BBN Technologies Corp. under DARPA/CMO Contract # HR0011-08-C-0004. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency or the U.S. Government.

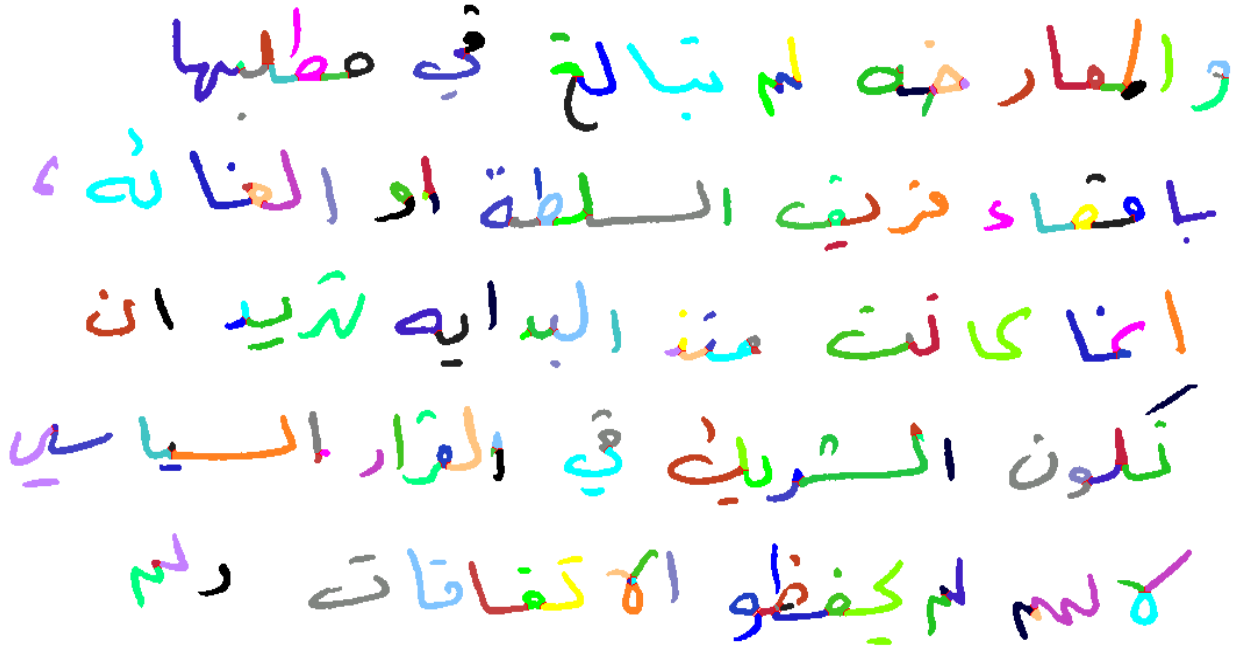

Figure 9: Sample result of a segmented document.

References

[1] Ethnologue: Languages of the World, 14th ed: SIL International, 2000.

[2] S. Hussain, N. Durrani, S. Gul, " Survey of Language Computing in Asia," Center for Research in Urdu Language Processing, National University of Computer and Emerging Sciences, 2005-02, 2005

[3] L. Lorigo and V. Govindaraju, "Segmentation and pre-recognition of Arabic handwriting," proc. Int'l conf. Document Analysis and Recognition, vol. 2, pp. 605-609, 2005.

[4] L. Lorigo and V. Govindaraju, "Off-line Arabic Handwriting Recognition: A Survey," Department of Computer Science and Engineering, University at Buffalo Technical Report 2005-02, 2005.

[5] G. Kim and V. Govindaraju, "A Lexicon Driven Approach to Handwritten Word Recognition for Real Time Applications," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 19, no. 4, pp. 366-379, Apr. 1997.

[6] T. Caesar, J. Gloger, E. Mandler "Using lexical knowledge for the recognition of poorly written words.," ICDAR 1995 915-918.

[7] M.K. Brown and S. Ganapathy, "Preprocessing Techniques for Cursive Word Recognition," Pattern Recognition, vol. 16, no. 5, pp. 447–458, 1983.

[8] J. Schürmann, N. Bartneck, T. Bayer, J. Franke, E. Mandler, and M. Oberlander, "Document Analysis—From Pixels to Contents," Proc. IEEE, vol. 80, no. 7, pp. 1,101–1,119, 1992.

[9] H. Freeman, "Techniques for the Digital Computer Analysis of Chain-Encoded Arbitrary Plane Curves," Proc. Nat. Electronics Conf., vol. 17, pp. 412–432, 1961.

[10] Lam, L., Seong-Whan Lee, and Ching Y. Suen, "Thinning Methodologies-A Comprehensive Survey," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 14, No. 9, September 1992, page 879, bottom of first column through top of second column.

[11] S. Mersa and A. Darwish. A new parallel thinning algorithm for gray scale images. Proceedings of the IEEE-EURASIP Workshop on Nonlinear Signal and Image Processing, pages 409.413, 1999. 2.

[12] Y. S. Chen and W. H. Hsu, "A new parallel thinning algorithm for binary image,", Taiwan, 1985, pp. 295-299.

[13] M.K. Brown and S. Ganapathy, Preprocessing Techniques for Cursive Word Recognition," Pattern Recognition, vol. 16, no. 5, pp. 447–458, 1983.