Offline Handwritten Arabic Character Segmentation with Probabilistic Model[*]

Pingping Xiu, Liangrui Peng, Xiaoqing Ding, and Hua Wang

Dept. of Electronic Engineering, Tsinghua University,
State Key Laboratory of Intelligent Technology and Systems,
100084 Beijing, China
{xpp, plr, dxq, wangh}@ocrserv.ee.tsinghua.edu.cn

Abstract. The research on offline handwritten Arabic character recognition has received more and more attention in recent years, because of the increasing needs of Arabic document digitization. The variation in Arabic handwriting brings great difficulty in character segmentation and recognition, eg., the subparts (diacritics) of the Arabic character may shift away from the main part. In this paper, a new probabilistic segmentation model is proposed. First, a contour-based over-segmentation method is conducted, cutting the word image into graphemes. The graphemes are sorted into 3 queues, which are character main parts, sub-parts (diacritics) above or below main parts respectively. The confidence for each character is calculated by the probabilistic model, taking into account both of the recognizer output and the geometric confidence besides with logical constraint. Then, the global optimization is conducted to find optimal cutting path, taking weighted average of character confidences as objective function. Experiments on handwritten Arabic documents with various writing styles show the proposed method is effective.

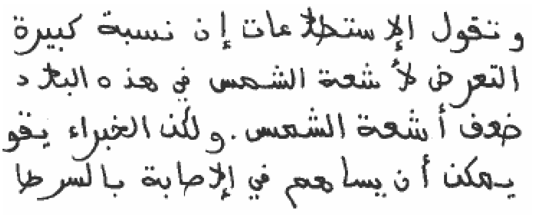
1 Introduction

The technique of Optical Character Recognition (OCR) has been the subject of intensive research for decades. In recent years, research on Arabic OCR achieves more and more attentions due to the increasing interaction between western world and Arabic world. As the unique characteristics of Arabic script, it is hard to implement existing frame of segmentation algorithm of other languages. Several most prominent features that are closely relevant to the designing of OCR system are described as follows:

1. There are 28 basic characters in Arabic Script, most of which have four different forms depending on their position in a word.
2. The Arabic characters of a word are connected along a baseline, no matter printed case or handwritten case. This inherent characteristic of connectivity is a crucial challenge to the segmentation algorithm designing. (Fig 1).
3. Many Arabic characters have sub parts (diacritics), which are positioned above or below the main parts of the character. In handwritten script, the relative positions between them vary a lot.

[*] This paper is supported by National Natural Science Foundation of China (project 60472002).

H. Bunke and A.L. Spitz (Eds.): DAS 2006, LNCS 3872, pp. 402 – 412, 2006.



وتقول الإستطلاعات إن نسبة كبيرة
التعرض لأشعة الشمس في هذه البلاد
ضعف أشعة الشمس. ولكن الخبراء يقو
يمكن أن يساهم في الإصابة بالسرطا

Fig. 1. Arabic Script Sample

A typical Arabic recognition system consists of five stages: pre-processing, segmentation, feature extraction, classification and post-processing, among which the specific segmentation module is the most challenging step. For the word segmentation, the analytical approaches that segment the words into individual characters [2-7] are suitable for handling a large vocabulary of words. In this paper, we adopt this kind of approach.

As [11] points out, there are 2 fundamental strategies in analytical segmentation, which are segmentation-then-recognition and segmentation-based strategy. The latter outweighs the former in integrating the extra recognition information, so it is adopted more widely. Within the segmentation-based strategy, there are also 2 different approaches, which are explicit segmentation and implicit segmentation. Usually, the implicit approach is sensitive to the variation of fonts and is characterized by a large computation complexity. Thus, in this paper, we adopt the explicit segmentation.

Explicit segmentation usually consists of two steps, which are over-segmentation and searching. Over-segmentation tries to segment a given word into smaller entities (called graphemes) that are ideally parts of integral characters; based on these graphemes, the searching step finds the best cutting path to obtain the integral characters.

For the over-segmentation step, numerous methods have been proposed in bibliography. The vertical projection histogram is the most direct feature to get the candidate cutting columns [2, 12, 13], however, these systems can not deal with the cases where characters are too close to each other. Also, The morphological stroke extraction [14] and the skeleton analysis [3] are both frequently used, however, none of them shows robust to the handwritten case. Contour analysis [4, 5, 15-18] shows more robust, perhaps because of the simple form of Arabic characters on top contour. In this paper, a contour-based algorithm is presented, which shows robustness through experiments. For the searching step, we propose a model to evaluate candidate solutions with an objective function. The confidence of individual character is integrated with 3 types of information, which are recognition information, segmentation information and logical constraint.

Specific to offline handwritten Arabic application, there remains a problem that has not received enough attention in bibliography, that there are frequently sub parts below or above the main parts, and moreover, the relative position between them often varies a lot in handwriting case (Fig 2), for which we must find the correct association between sub and main parts. Unfortunately, few systems investigated are trying to solve this problem. Most systems simply assume that the correct cutting columns on main parts also cut sub parts correctly [5-7]. In our algorithm, we consider all the possible combination of sub parts and main parts, and propose a 3-queue model to generate the candidate segmentation solutions. This feature shows its advantage when dealing with the cases like Fig 2.

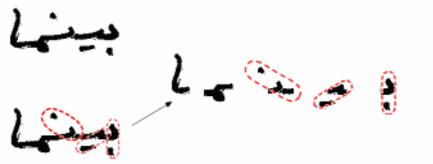


Fig. 2. The cases that sub parts are hard to be associated to their mother main parts. The main parts of characters are too close to each other. As a result, the associating cannot be simply executed through attributing the sub parts to the most near main part.

The rest of the paper is organized as follows: Section 2 introduces our probabilistic segmentation model, and the 3-queue segmentation candidate technique is described in this section. Section 3 illustrates the workflow of the algorithm. Section 4 describes the contour-based over-segmentation step, which generates original segmentation graphemes. Section 5 discusses the experiments and results. The conclusion is in Section 6.

2 The Probabilistic Segmentation Model

In this paper, we propose a new segmentation model for generating segmentation candidates. Fig 3 illustrates that any segmentation candidate can be represented by the corresponding state path in 3-dimentional space. Denote e_i ($0 < i \le N$) as the sequence of graphemes segmented from the main parts, e_i^u ($0 < i \le N^u$) as the sequence of sub parts above the main part, and e_i^d ($0 < i \le N^d$) as that below the main part, all of which follow the left-to-right order. We make the assumption that any valid integral character consists of three sequences of graphemes, each sequence being a consecutive subsequence from e_i ($0 < i \le N$), e_i^d ($0 < i \le N^d$) and e_i^u ($0 < i \le N^u$) respectively. With this assumption, any character sequence $\mathbf{c}_k$ ($0 < i \le N$) (sorted from left to right) can be segmented through cutting vector sequences $\mathbf{x}_k = (x_k, x_k^u, x_k^d)$ ($k = 0, 1, \ldots N$) (Fig 3),

$$\begin{cases} \mathbf{c}_{k+1} = \left\{ e_i \middle| x_k < i \le x_{k+1} \right\} \cup \left\{ e_i^u \middle| x_k^u < i \le x_{k+1}^u \right\} \cup \left\{ e_i^d \middle| x_k^d < i \le x_{k+1}^d \right\} \\ \qquad = c_{k+1} \cup c_{k+1}^u \cup c_{k+1}^d, \quad 0 \le k < N \\ 0 = x_0 \le x_1 \le \cdots \le x_N = N \\ 0 = x_0^u \le x_1^u \le \cdots \le x_N^u = N^u \\ 0 = x_0^d \le x_1^d \le \cdots \le x_N^d = N^d \end{cases} \tag{1}$$

For $\mathbf{x}_k$ ($k = 0, 1, \ldots N$), there is

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{u}_k \quad (\mathbf{x}_k \in X, \mathbf{u}_k \in U, 0 \le k < N) \tag{2}$$

where $X = \left\{ (x, x^u, x^d) \middle| 0 \le x \le N, 0 \le x^u \le N^u, 0 \le x^d \le N^d \right\}$, $\mathbf{u}_k$ is restricted in the space $U = \left\{ (u, u^u, u^d) \middle| u \ge 1, u^u \ge 0, u^d \ge 0 \right\} \cup \left\{ (0,0,0) \right\}$. The cutting path is represented by $\left\{ \mathbf{x}_k \right\}$ with the initial and final states constrained to $\mathbf{x}_0 = \left(0,0,0\right)$ and $\mathbf{x}_N = \left(N, N^u, N^d\right)$.

The goal of segmentation is to optimize the objective function

$$\text{conf} = \frac{\sum_i n_i \cdot \text{conf}_i}{\sum_i n_i} \tag{3}$$

which is the weighted average of the individual character confidence. n_i is the weight of the corresponding character, which is defined as the number of graphemes in the character.

conf_i, the confidence of i-th character, can be defined as

$$\text{conf}_i = \max_{\text{code}i} \log P\left(\text{code}_i, \text{img}_i\right) \tag{4}$$

where code_i is the hypothesis code of i-th character, and img_i is i-th character's image.

$$P\left(\text{code}, \text{img}\right) = P\left(\text{code} \middle| \text{img}\right) \cdot P\left(\text{img}\right) \tag{5}$$

$P\left(\text{img}\right)$ is related to the probability of the geometric configuration, defined as

$$P\left(\text{img}\right) = P\left(\text{rect}\left(c\right)\right) \cdot P\left(\text{rect}\left(c^u\right)\right) \cdot P\left(\text{rect}\left(c^d\right)\right) \cdot P\left(\text{pos}\left(c^u, c\right)\right) \cdot P\left(\text{pos}\left(c^d, c\right)\right) \tag{6}$$

$$P\left(\text{rect}\left(\varnothing\right)\right) = P\left(\text{pos}\left(\varnothing, \cdot\right)\right) = 1 \tag{7}$$

where c, c^u and c^d are the main, upper part, lower part of img respectively.

$\text{rect}\left(\cdot\right)$ represents the pair of $\left(width, height\right)$, the parameters of the bounding box of the grapheme, and $\text{pos}\left(\cdot\right)$ represents the relative position $\left(Xpos, Ypos\right)$ of the two graphemes (taking the centroid of the latter one as reference point). $\text{rect}\left(c\right)$, $\text{rect}\left(c^u\right)$, $\text{rect}\left(c^d\right)$, $\text{pos}\left(c^u, c\right)$ and $\text{pos}\left(c^d, c\right)$ are all taken as 2 dimensional random variable, which can be assumed as normal distribution. The parameters of the normal distribution can be estimated based on large samples with the EM algorithm.

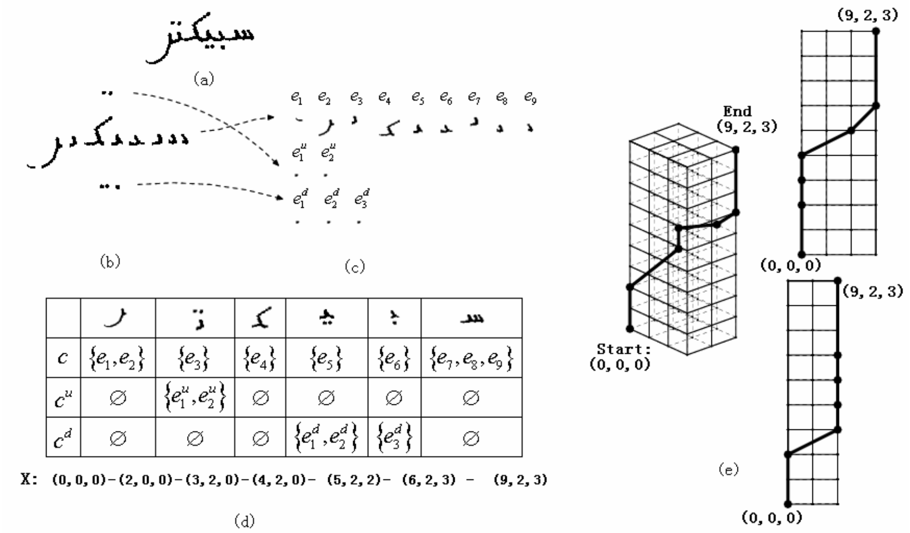

(a)
(b)
(c)
e_1 e_2 e_3 e_4 e_5 e_6 e_7 e_8 e_9
e_1^u e_2^u
e_1^d e_2^d e_3^d
c: $\{e_1,e_2\}$ $\{e_3\}$ $\{e_4\}$ $\{e_5\}$ $\{e_6\}$ $\{e_7,e_8,e_9\}$
c^u: $\varnothing$ $\{e_1^u,e_2^u\}$ $\varnothing$ $\varnothing$ $\varnothing$ $\varnothing$
c^d: $\varnothing$ $\varnothing$ $\varnothing$ $\{e_1^d,e_2^d\}$ $\{e_3^d\}$ $\varnothing$
X: (0, 0, 0)-(2, 0, 0)-(3, 2, 0)-(4, 2, 0)- (5, 2, 2)- (6, 2, 3) - (9, 2, 3)
(d)
End
(9, 2, 3)
Start:
(0, 0, 0)
(9, 2, 3)
(0, 0, 0)
(9, 2, 3)
(0, 0, 0)
(e)

Fig. 3. The cutting mechanism in 3-queue model. (a) is the Arabic word to be cut, (b) is the segmented grapheme sequence of the main part, (c) is the sorted 3- queue grapheme sequences for main part, over-baseline and under-baseline. (d) is the representation of segmented integral characters, each one is combined with 3 components of c, c^u, c^d. (e) the 3-dimensional state space for $\mathbf{x}_k$, the cutting path starts with state (0,0,0) and ends with state (9,2,3).

$P(\text{code}|\text{img})$ is related to the recognition output of the input image i. In our system, we calculate $P(v|i)$ by calculating $P(v_c|c)$, $P(v_u|c^u)$ and $P(v_d|c^d)$ separately, where v_c is the hypothesis code for main part c, and v_u, v_d for c^u and c^d respectively. We define

$$\begin{aligned} P(\text{code}|\text{img}) &= P(\text{code}, v_c(\text{code}), v_u(\text{code}), v_d(\text{code})|\text{img}) \\ &= \max_{v_c, v_u, v_d} P(\text{code}|\text{img}, v_c, v_u, v_d)\cdot P(v_c|c)\cdot P(v_u|c^u)\cdot P(v_d|c^d) \end{aligned} \tag{8}$$

$P(v_c|c)$, $P(v_u|c^u)$ and $P(v_d|c^d)$ are estimated by the statistical recognition module provided in [21]. $P(\text{code}|\text{img}, v_c, v_u, v_d)$ is defined as:

$$P(\text{code}|\text{img}, v_c, v_u, v_d) = \begin{cases} 1, & v_c, v_u, v_d \text{ can combine to form 'code'} \\ \beta, & v_c, v_u, v_d \text{ cannot combine to form 'code'} \end{cases} \tag{9}$$

The parameter β determines how strong the logical rule imposed. The logical rule will affect the segmentation more sensitively when β is getting closer to 0.

3 The Workflow of the System

The workflow of our system is presented by Fig 4. The preprocessing phase consists of page decomposition, line cutting, word cutting and normalization. The first 3 steps follow the approach presented in [6], and the normalization step makes the average character height as the unit for all geometric measures. In the segmentation phase, there is a sorting step that executes: 1. splitting the sub parts into two queues: over-baseline and under-baseline; 2. sorting the three queue into left-to-right order. In this step, e_i $(0<i\le N)$, e_i^d $(0<i\le N^d)$ and e_i^u $(0<i\le N^u)$ are obtained. The pre-recognition step recognizes all the possible c, c^u and c^d, storing the recognition information into a database, which will be repeatedly used in searching step. Then, the searching task is facilitated with Dynamic Programming algorithm, which can find global optimal solution, that is, the solution with lowest objective function value conf^*, with low computation cost. In this step, the recognition results can be directly obtained at the same time that we get the segmentation solution.

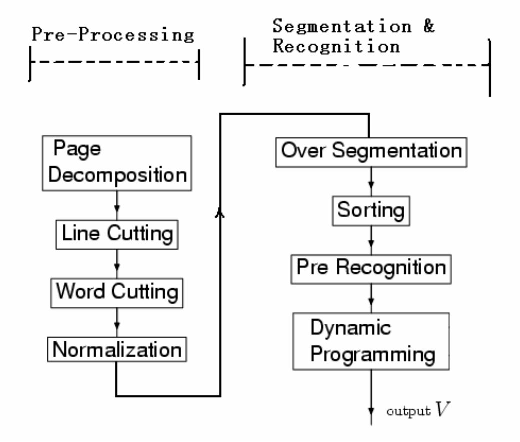


Fig. 4. The overview of the recognition system

4 The Step of Over-Segmentation

Over-segmentation is the step prior to the step of sorting and segmentation (Fig 4), which partitions the connected component of the main part into graphemes. In this step, the contour-based analysis is applied, which can efficiently find the candidate cutting positions with robustness. The rule for choosing candidate cutting positions is listed as following:

- **Rule1** all the local minima of the top contour are treated as candidate cutting points.
- **Rule2** all the points on top contour which have a distance to bottom contour smaller than 4/3 stroke width are treated as candidate cutting points. The stroke width has been calculated through histogram of vertical run-length.

- **Rule3** similar to rule 2, that all the points on bottom contour which have a distance to top contour smaller than 4/3 stroke width are also treated as candidate cutting points.
- **Rule4** the points of top contour, which are on the base line, are treated as candidate cutting points. The base line is extracted with Hough transform [8, 15].
- **Rule5** (the rule for filtering the candidates generated from rule 1,2,3,4) the distance between any two candidate cutting points must be more than 3/2 stroke width, otherwise tick out one point.

These rules are based on the typical characteristic of both printed and handwritten Arabic, that, Arabic characters are usually connected on the baseline. Based on this hypothesis, we take the points located on baseline as the candidate cutting points. Three different contour features can help us to find the candidates, including the local minima (rule 1), the stroke segments with the shape of "bottleneck"(rule 2, 3), and the points lied on the baseline detected by Hough transform (rule 4). Though some extra candidates are introduced, the losing cases are reduced. (Fig 5)

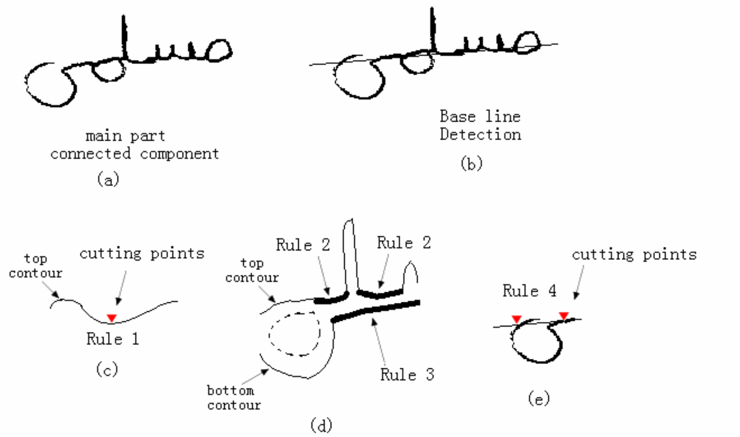


Fig. 5. The rules for finding candidate cutting points. (c) Rule 1 finds local minima on top contour; (d) Rule 2, 3 find segments on contour that are close to the contour on other side. The bold black lines represent the segments that cutting points can be located on. (e) Rule 4 finds candidate cutting point that located on the baseline.

5 Experiments and Results

5.1 Comparative Result

We have a text database of 20000 characters, including various writing styles, with which we segmented and labeled half of the text manually, training the segmentation model and recognition module separately, and chose test samples from another half. We classify them into 5 types (Fig 6). Each type of samples contains about 2000 characters.

Taking $\beta = 0$ (in equation (9)), the results on the 5 test sets are listed in Table 1, from which we can observe that our algorithm performs considerably well on all the 5 sets.

However, with the system that is presented in [6], (using the same training set) we find that the performance deteriorates significantly. (Table 2).

The segmentation rates are all fall behind that of our algorithm, which may be explained as following: this system simply associates the sub parts to the nearest mother main part character. This greatly limits the solution searching, and neglects the logical constraints.

Table 1. Performance of our algorithm

Our method	S1	S2	S3	S4	S5	Average
Total Recognition Rate (%)	69.0	59.4	57.7	54.9	54.9	59.2

Table 2. Performance of the competing method

Competing method	S1	S2	S3	S4	S5	Average
Total Recognition Rate(%)	43.1	44.1	46.5	34.6	38.2	41.3

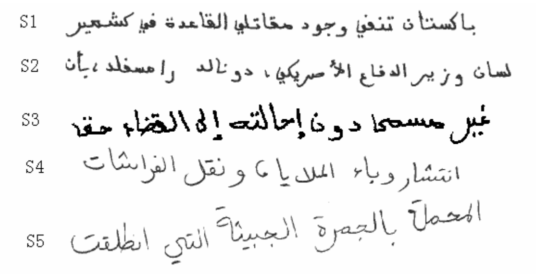


Fig. 6. The test sets for experiments. They are of various writing styles.

5.2 The Significant Role of Logical Rule

The logical rule expressed in $P(\text{code}|\text{img}, v_c, v_u, v_d)$ is crucial to the performance of the algorithm. The parameter of β determines the strength $P(\text{code}|\text{img}, v_c, v_u, v_d)$ imposed on the optimizing process. In basic Arabic characters, the main parts (isolated form) consist of 16 forms including ط, ع, ا, ح, د, ر, س, ى, و, ه, ں, م, ك, ل, ق and ٮ; the upper sub parts consist of 5 forms including null, • (1-dot), •• (2-dot), ∴ (3-dot) and ء (hazma); and the lower sub parts have 3 forms including null, • and ••. The total combination possible cases would be $16 \cdot 5 \cdot 3 = 240$. However, the valid codes in alphabet include only 28 characters, 10 percent of the 240 possible combinations. It may indicate that the logical constraints can play a considerable role in selecting the correct segmentation.

To infirm this inference by experiments, we design 3 tests: A. let $\beta = 0$; B. let $\beta = 0.2$; C. choose the system presented in [6], (which has been compared with ours above).

Table 3. Comparison of experiments A,B,C

Experiment No.	S1	S2	S3	S4	S5	Average
A (%)	69.0	59.4	57.7	54.9	54.9	58.8
B (%)	49.7	40.9	30.6	47.2	47.1	43.2
C (%)	43.1	44.1	46.5	34.6	38.2	41.3

From the results, we can observe that the results of experiment B fall much behind of that of experiment A, which clearly proves that the logical rule plays a great role in performance. Experiment B's results are close to that of experiment C, which does not use the logical rule and performs directly recognition on combinations of sub and main parts. The results may also indicate that the segmentation-based scheme outweighs the segmentation-then-recognition scheme, because of the integration of the recognition result.

5.3 Result Analysis

We set $\beta = 0$, and analyze the error of the segmentation. The data is as follows:

Table 4. Error analysis (The different error types may overlap to some specific cases)

Error type	S1	S2	S3	S4	S5
Under-segmentation (%)	4.0	9.4	6.8	12.8	5.8
Over-segmentation (%)	10.6	13.2	7.5	16.4	9.7
Sub-to-main association (%)	2.3	0.8	3.4	4.3	1.6

The errors can be mainly classified into 4 categories:

– Ambiguity in recognition

Since our algorithm use the recognition output as the base for segmenting decision, the right segmentation lies greatly to the correct recognition. Sometimes it is difficult to distinguish the part of character from the integral character, for example, ـ, (the tail part of character ت), and the character ا (Alef), share the same shape of a vertical stroke, as a result, they may be difficult to classify by the recognition module. So the tail of ت is quite likely to be cut out as an integral character. In future, we should improve the performance of the recognition module to deal with this kind of problem.

– Failure in over-segmentation

Sometimes the over-segmentation encounters some irregular cases that are still unable to segment. Often it happens where the stroke is too wide, or the characters are too close to each other.

– Ligatures or unique writing conventions
In Arabic, sometimes two or more characters overlap with each other to form a new shape, and these combinations are frequently appeared. In our algorithm, we can treat some of the combinations that appear most frequently as a new character, however, sometimes the frequency of the ligature is low, and the total number of these cases is large. To improve the performance of the system, we have to study these cases more comprehensively.

– Other errors

6 Conclusion

In this paper, a 3-queue model is designed to generate the candidate segmentation solutions. It is mainly aimed to deal with the characteristic of Arabic that the sub parts are isolated from the main parts and their relative positions vary a lot. Specifically, the graphemes are lined in 3 queues, generating a state space that contains all the possible segmentations, in contrast with other "greedy approaches", which takes only the nearest main part as the mother main part of a sub part. Besides, in our model, we integrate the logical and the geometrical constraint together with recognition output in confidence calculating, using probabilistic model. The over-segmentation is designed with contour analysis, which obtains the candidate cutting points through 3 different aspects of contour features.

From our experiments, the robustness of algorithm is verified as the results turn out to be smooth on the 5 selected data sets, which are in different writing style. The comparative experiment is also conducted showing the advantage of our algorithm over traditional ones. However, we have not jet used the lexicon information in this paper, which seems to be much potential in enhancing the performance. In future, we should also try to incorporate the system with the lexicon restriction.

References

1. Al-Yousefi, H. and S.S. Udpa, *Recognition of Arabic characters.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 1992.
2. Amin, A. and J.F. Mari, *Machine recognition and correction of printed Arabic text.* Systems, Man and Cybernetics, IEEE Transactions on, 1989.
3. Amin, A. and H.B. Al-Sadoun. A new segmentation technique of Arabic text. in Pattern Recognition, 1992. Vol.II. Conference B: Pattern Recognition Methodology and Systems, Proceedings., 11th IAPR International Conference on. 1992.
4. Sari, T., L. Souici, and M. Sellami, *Off-line handwritten Arabic character Segmentation algorithm: ACSA.* Frontiers in Handwriting Recognition, 2002. Proceedings. Eighth International Workshop on, 2002: p. 452 - 457.
5. Olivier, C., et al. Segmentation and Coding of Arabic Handwritten Words. in 13th International Conference on Pattern Recognition (ICPR'96). 1996.
6. Jin, J., et al., *Printed Arabic document recognition system.* Vision Geometry XIII. Edited by Latecki, Longin J.; Mount, David M.; Wu, Angela Y. Proceedings of the SPIE, 2004. **5676**: p. 48-55.

7. Cheung, A., M. Bennamoun, and N.W. Bergmann. A recognition-based Arabic optical character recognition system. in Systems, Man, and Cybernetics, 1998. IEEE International Conference on.
8. Pechwitz, M. and V. Maergner. HMM based approach for handwritten Arabic word recognition using the IFN/ENIT - database. in Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on.
9. Fakir, M., M.M. Hassani, and C. Sodeyama. Recognition of Arabic characters using Karhunen-Loeve transform anddynamic programming. in Systems, Man, and Cybernetics, 1999. IEEE SMC '99 Conference Proceedings. 1999 IEEE International Conference on.
10. Dehghan, M., et al. Holistic handwritten word recognition using discrete HMM and self-organizing feature map. in PROC IEEE INT CONF SYST MAN CYBERN. 2000.
11. Bortolozzi, F., et al. Recent advances in handwriting recognition. in Proceedings of the IWDA'05. 2005.
12. Sarfraz, M., S.N. Nawaz, and A. Al-Khuraidly. Offline Arabic Text Recognition System. in 2003 International Conference on Geometric Modeling and Graphics (GMAG'03). 2003.
13. Najoua, B.A. and E. Noureddine. A robust approach for Arabic printed character segmentation. in Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on. 1995.
14. Motawa, D., A. Amin, and R. Sabourin. Segmentation of Arabic cursive script. in Document Analysis and Recognition, 1997., Proceedings of the Fourth International Conference on. 1997.
15. Bushofa, B.M.F. and M. Spann. Segmentation of Arabic characters using their contour information. in The 1997 13th International Conference on Digital Signal Processing, DSP. Part 2 (of 2).
16. Romeo-Pakker, K., H. Miled, and Y. Lecourtier. A new approach for Latin/Arabic character segmentation. in Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on. 1995.
17. Tolba, M.F. and E. Shaddad, *On the automatic reading of printed Arabic characters.* Systems, Man and Cybernetics, 1990. Conference Proceedings. IEEE International Conference on, 1990: p. 496-498.
18. Maergner, V. SARAT-a system for the recognition of Arabic printed text. In Pattern Recognition, 1992. Vol.II. Conference B: Pattern Recognition Methodology and Systems, Proceedings., 11th IAPR International Conference on.
19. Elgammal, A.M. and M.A. Ismail. A Graph-Based Segmentation and Feature-Extraction Framework for Arabic Text Recognition. in Sixth InternationalConference on Document Analysis and Recognition (ICDAR'01). 2001.
20. Lethelier, E., M. Leroux, and M.G.L. Poste. *An automatic reading system* for handwritten numeral amounts on French checks. in Proceedings of the Third International Conference on Document Analysis and Recognition. 1995
21. Wang, H., et al. New statistical method for machine-printed Arabic character recognition. in Proceedings of SPIE -- Volume 5676 Document Recognition and Retrieval XII, Elisa H. Barney Smith, Kazem Taghva, Editors. 2005.