Journal of Computer Science 3 (7): 549-555, 2007
ISSN 1549-3636
© 2007 Science Publications

Optical Character Recognition System for Arabic Text Using Cursive Multi-Directional Approach

[1]Mansoor Al-A'ali and [2]Jamil Ahmad
[1]Computer Science Department, College of Information Technology, University of Bahrain, P. O. Box 32038, Sakheer Campus, Kingdom of Bahrain
[2]Dean, IQRA University, Pakistan

Abstract: This paper presents a novel new technique based on feature extraction and on dynamic cursor sizing for the recognition of Arabic Text. The most challenging area in Arabic OCR (AOCR) research is the segmentation of words into their sub-words and their individual characters. Several rules are defined that govern the size and movement of the cursor through each segment. The features obtained from each segment are termed strokes and each segment is defined by a number of strokes where each stroke is defined mainly in terms of a sequence of directions. The basic concept followed here is a logical, dynamically sized cursor that is used to "travel" through a text image of one word at a time while extracting features of strokes. The strokes obtained are then "pieced" back together to be classified into character classes based on a knowledge base and eventual recognition of characters is achieved. The results demonstrate that the technique is successful.

Key words: Arabic, OCR, features, strokes, segments

INTRODUCTION

OCR is the process of converting a raster image representation of a document into a format that a computer can process. Thus, it may involve many sub disciplines of computer science including image processing, pattern recognition, natural language processing, artificial intelligence, and database systems [1].

Despite intensive investigation, the ultimate goal of developing an optical character recognition (OCR) system with the same reading capabilities as humans still remains unachieved and more so in the case of Arabic language.

OCR has attracted researchers interest not only because of the very challenging nature of this problem to shorten the reading capabilities gap between machines and humans but also because it improves human machine interaction in many applications. Example applications include office automation, cheque verification, and a large variety of banking, business and data entry applications.

Most commercially available OCR products are for typed English text because English text characters are separated from one another with spaces and do not have all the extra complexities associated with Arabic letters. This is the reason why English OCR techniques and systems are easier and well developed.

Arabic is a popular script. It is estimated that there are more than one billion Arabic script users in the world. If OCR systems are available for Arabic characters, they will have a great commercial value. However, due to the cursive nature of Arabic script, the development of Arabic OCR systems involves many technical problems, especially in the segmentation stage. Although many researchers are investigating solutions to solve the problems very little progress has been made. Some researchers focused on segmentation, thinning and pattern matching whilst others focused on using neural networks for training the OCR system2,3 [2,3]. Arabic is a cursive-type language, which is written from right to left and therefore recognition should occur in this way. There are 28 characters in the Arabic alphabet. Each character has two to four different forms, which depends on its position in the word or sub-word. As a result, there are 100 classes to be recognized. Fig. 1 shows the basic isolated individual letters.

An Arabic word can itself consist of one or more sub-words (**مكتبة**). Most characters have dot(s), zigzag(s), madda, kashida, etc., associated with the character and this can be above, below, or inside the character. Many characters have a similar shape. The position or number of these secondary strokes and dots makes the only difference, for example, the following word has two letters which are identical but for the dots above one and below the other.

Corresponding Author: Mansoor Al-A'ali, Department of computer science, college of information technology, university of Bahrain, PO Box 32038, Kingdom of Bahrain

ا	ب	ت	ث	ج	ح	خ	د	ذ	ر
Alef	Bah	Tah	Thah	Geam	Hah	Khah	Dal	Thal	Rah
ع	غ	ظ	ط	س	ش	ص	ض	ز	
Zean	Seen	Sheen	Sad	Dad	Tah	Zah	Ayn	Ghayn	
ف	ق	ك	ل	م	ن	ه	و	ى	
Pha	K'aaf	Kaaf	Lam	Meem	Noon	Ha	Waw	Yah	

Fig. 1: Basic Isolated Arabic characters

Arabic words may horizontally overlap and characters may stack on others. These introduce problems for both the word and the character segmentations. At this stage, it is not hard to understand that segmentation is a crucial step in the development of an Arabic OCR system.

Arabic uses many ligatures, especially in handwritten text. Ligatures are characters that occupy a shared horizontal space creating vertically overlapping connected or disconnected letters; '*Madda*', and diacritic objects. These make the task of line separation and segmenting text more difficult. Arabic uses many types of external objects, such as dots, '*Hamza*' which are common amongst certain sets of letters. Characters can be written in many different sizes, writing instruments (varying thickness and stroke quality), and slants (causing character sharing along the horizontal axis). The shape of the letter is influenced by its position in the word. The letter is written differently if it is written at the beginning, at the end, in the middle or it is isolated. For example, the letter ك can be written in the shapes shown in Fig. 2.

isolated	ك
beginning	كـ
middle	ـكـ
end	ـك

Fig. 2: Shapes of an Arabic letter 'ك'

Fifteen letters of the Arabic alphabet have one, two or three "dots" placed above or below the character body, for example:

ت ب ق ث ف ي

Four letters have "hamza", each in a different place, for example:

أ ىء ء ؤ

Character segmentation can be performed by either the dissection or recognition-based techniques. Dissection means the decomposition of the image into a sequence of sub-images using general features. It involves analysis of the image into its sub-image segmentation paths. Each sub-image is treated as a character for recognition. It is worth mentioning that classification of characters is carried out at a later stage. Projection analysis, connected component processing, and white space and pitch ending are some of the common dissection techniques used by OCR systems. These techniques are suitable for scripts which have spaces between characters. If a dissection technique is used for cursive scripts, a more intelligent and specific analysis technique for the particular script is needed. However, there is still no guarantee that high segmentation accuracy can be achieved. The basic principle of recognition-based character segmentation is to use a mobile window of variable width to provide the tentative segmentations which are confirmed (or not) by the classification. Characters are by-products of the character recognition for systems using such a principle to perform character separation. The main advantage of this technique is that it bypasses serious character separation problems. In principle, no specific segmentation algorithm for the specific script is needed and recognition errors are mainly due to failures during the classification stage. For these reasons, more and more cursive script OCR systems use this technique.

Despite the research work done so far on the recognition of handwritten Latin and Asian languages text and the excellent results obtained in Latin text, a few research papers and reports have been published for handwritten Arabic text recognition [1].

Research in Arabic OCR has been gaining momentum and some attempts have been reported. Most published research so far does not deal with all issues of Arabic OCR, typed or handwritten, but is rather focused on specific aspects. Kavianifar etal [4] report on a feature extraction for a multi-font OCR. The approach works on the recognition of sub-words of all words and then the global features for each word are

extracted. A concept of Contour tracing is used in the feature extraction phase.

Alma'adeed etal[2] use the Hidden Markov models (HMM) with some success in recognizing certain types of printed Arabic words. They present unconstrained Arabic handwritten word recognition using a model which is based HMM. The system first attempts to remove some of the variation in the images that do not affect the identity of the handwritten word. Next, the system codes the skeleton and edge of the word so that feature information about the lines in the skeleton is extracted. Then a classification process based on the HMM approach is used.

Other researchers have combined OCR with post processing to correct the results by utilizing the Arabic morphology[5]. The technique corrects substitution and rejection errors. However, such approach will face problems since post OCS correction cannot possibly be done for most Arabic words since many two, three or four letter words can be difficult to precisely decide.

Sari etal[6] present an Arabic character segmentation algorithm of Arabic scripts. The developed segmentation algorithm yields on the segmentation of isolated handwritten words in perfectly separated characters only. It is based on morphological rules, which are constructed at the feature extraction phase. Finally, the Arabic character segmentation algorithm is combined with an existing handwritten Arabic character recognition system.

Cowell etal[7,8] used normalized isolated characters for size and extracts an image signature based on the number of black pixels in the rows and columns of the character and compares these values to a set of signatures for typical characters of the set. This technique identifies the closet match and gives the closeness of match to all other characters in the set, which is expressed in a triangular confusion matrix. The problem with this approach is that the thinning process and the length of the characters would eventually produce many fuzzy situations and hence imprecise decisions would appear.

Cheung etal[9] introduced Arabic word segmentation algorithm to separate horizontally overlapping Arabic words/sub-words. There is also a feedback loop to control the combination of character fragments for recognition. The authors claim a 90% recognition accuracy with a 20 chars/s recognition rate.

Al-Ohali et al [10] describe an the development of an Arabic cheque recognition system based n the use of a database of letter recognition information databases for the recognition of hand-written Arabic cheques. The databse contains a databases of real-life Arabic legal amounts, Arabic sub-words, courtesy amounts, Indian digits, and Arabic cheques. They describe a validation procedure which includes grammars as well as some algorithms to verify the correctness of the tagging process. This approach is limited since it deals with a limited set of words representing the words representing numbers and needs a more elaborate testing of the handwritten words.

The rest of the paper is organized as follows: Section 2 presents our proposed model and the segmentation technique which is a further elaboration of the proposed approach, section 3 presents the implementation of the approach, section 4 presents the method for character classification, and section 5 presents the results of the implementation of the approach.

THE PROPOSED MODEL AND SEGMENTATION

Fig. 5 shows diagrammatically the overall mechanism of the proposed system where initially text is scanned and converted into a binary file. The image is then analyzed and classified using the techniques developed in this paper (which will be discussed in the following sections). The resulting AOCR system is for off line recognition of Arabic text. Initial tests were performed on computer typed text and they do not appear to be any visible problems to extend the system to work on handwritten text.

In the early stages of development, thinning of the image before extraction of the strokes was considered, but this idea was abandoned. The reason was that the text image was not being thinned accurately which would cause the stroke segmentation algorithm to crash. The decision was then taken to implement a logical, dynamically sized cursor that would take care of thinning and stroke determination simultaneously.

The segmentation of the image (Arabic text) is done at two levels. First, the text in the image matrix is split into lines of text using the horizontal projection technique and into segments using the vertical projection technique (i.e. location of horizontal lines of zero density of pixels, given the line of text from the horizontal projection technique, indicates the beginning of a segment and the subsequent location of another zero density line of pixels indicates the end of a segment, thus an entire segment is located). The segments obtained here can be entire words, part of a word, an isolated character, a diacritic, or a dot(s). For example the line of text (I study in the university of Bahrain) would be segmented as indicated by the enclosing boxes as shown in Fig. 3. Each word is segmented into its sub-words.

Arabic text sentence: Each segment would be identified by 2 sets of coordinate points - one for the bottom left point of the enclosing rectangle and another for the top right point of the enclosing rectangle.

أنا		أ د ر س		في		جا معة		ا لبحر ين	
ا	نا	ا	د	ر	س	جا	معة	ا	لبحر	ين

Fig.. 3: Arabic Text

Other preprocessing functions: Each segment is then marked with horizontal, vertical or intersection markings which is done using the middle of the distance method as follows:

1. Start from the left most X boundary of the segment.
2. Find the first 'on' (i.e. black) pixel while changing the X-axis value in the segment and record the location of this pixel as 'first'
3. Find the first 'off' (i.e. white) pixel while still changing X and record the previous most pixels 'last'
4. Calculate the midpoint as follows: midpoint = (first + last) div 2
5. Mark the pixel at the midpoint as horizontal
6. Repeat steps (2) to (5) till X reaches the end of the segment
7. Now start from the top leftmost Y boundary of the segment
8. Repeat steps (2) to (6) exactly except change Y instead of X. Also if a pixel has been marked as horizontal and has been calculated as a vertical marking also, it is marked as a intersection.

PATTERN AND FEATURE EXTRACTION RECOGNITION SYSTEM

For each of the segments the following subsections apply:

Feature Extraction, Definition and Representation: A horizontal and vertical intersection is located in the segment and this serves as the starting point of the logical, dynamically sized cursor. The cursor starts with one pixel, which gives it a size zero. A cursor size of one is indicated by including all 8 immediate neighboring pixels in the zero sized cursor. Further increases are indicated by increasing the outer layers of the cursor as shown in Fig. 4. The relationship between the size of the cursor and the number of pixels it occupies is: Number of pixels = ((size * 2) + 1) ^ 2

The cursor is first increased to an optimal size so as to cover less than or equal to a certain constant called *'PercentForOptimalSizeCursor'* which is a percentage of (0...1). All except stroked pixels should be included to calculate the percentage of pixels on in a cursor. The movement of the cursor is governed by the following rules. That is, a cursor can move in any of the eight directions by its whole size provided:

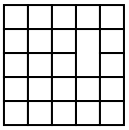
No. of pixels	Cursor appearance	SIZE
1		0
9		1
25		2

Fig. 4: Cursor appearance & Size

1. All pixels in the new cursor position are unvisited and not already part of another stroke
2. All the pixels in the new cursor position are at least a certain percentage ON. This percentage is indicated by a constant called 'PercentOn' (0..1)
3. If the cursor can move in more than a certain no. of directions, indicated by a constant called 'OptionLimit' (it can be a value between 1..8 but is usually set at 3), the cursor has to be resized, increased in this case, so as to reduce the no. of directions that the cursor can move in to meet the OptionLimit value.
4. If the cursor cannot move in any direction, it has to be resized, reduced in size in this case, so as to increase the no. of directions it can move in. But if by reduction, the cursor is able to move in more than the OptionLimit, then the previous size is taken.
5. If after resizing, the cursor can still not move in any direction, this means that

either a start point of a stroke has been found or if a start point was earlier located, an end point of a stroke has been determined

6. After the possible directions that a cursor can move in have been found, they are sorted according the most natural flow of direction. That is, the first choice for movement of the cursor will be so as to make the minimum change in direction from the previous direction, see Figure 6.

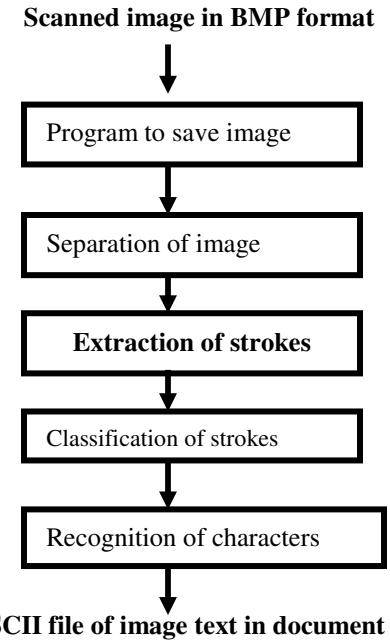


Fig. 5: The overall model of the proposed system.

If the cursor previously moved towards the North (NN) and the directions available for movement are: NN, NW, SE, SW then, after sorting the list of options from most natural flow of direction to least, the list would look like this:

Rank	Direction
1	NN
2	NW
3	SE. SW

SE and SW are ranked the same because they represent the same amount of change in direction.

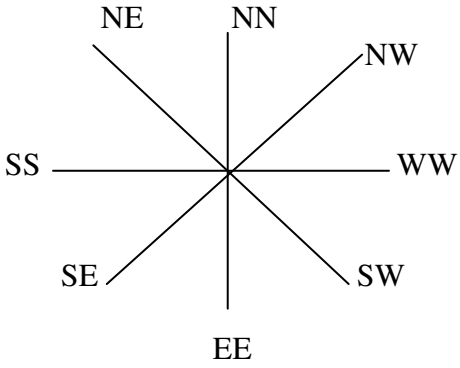


Fig. 6: Possible directions of Cursor movement

Therefore the cursor would choose the most natural flow of direction and the remaining choices would be pushed on to a stack in their sorted order, with the next most natural flow of direction at the top of the stack, see Fig. 6. The stack is emptied when the chosen direction leads to a start point after which the cursor moves as normal except that the directions and the size of the cursor at each direction is now recorded as the cursor moves. Also, the directions not taken during this stage are stored in an array as connection points between the points on the stroke and the untravelled points. After the cursor reaches an end point, a stroke has been found and recording of the directions ends. The next starting point for the cursor is obtained by popping the stack. If a stroke happens to start at the same point and the stack is empty, then either a dot has been located or an extra part of a segment not taken as part of any stroke has been found (in this case sampling will reject the extra part later on). The segment is checked for any un-stroked intersection pixels. If one is found, the entire process is repeated. If no un-stroked intersections can be found and the stack is empty, the stroke determination process ends for the current segment. Therefore, features of a segment are defined as a group of strokes and each stroke is represented by:

- A starting point
- A string of directions
- An ending point
- An array of cursor sizes that correspond to each direction in the direction string.
- An array of connection points between the present stroke and points which were not taken during the recording process and pushed on to the stack.

CHARACTER CLASSIFICATION

After the strokes of each segment have been identified and recorded, they have to be reassembled to form valid characters. Note that a stroke may be an isolated character, a character piece, two character pieces together, a diacritic sign or a dot(s). The relationship between strokes is deduced using the array of connection points for each stroke. Sampling will have to be done before classification to remove unnecessary strokes, that is, those that do not fit anywhere to form a valid character.

A knowledge base is built to contain information about character classes. For example, all characters that have a closed loop such as (ظ و ه ق م ص) could form one class and all characters that have a half open circle such as (خ ن ب ع) will form another character class and so on. Strokes will then be re-assembled to form valid characters and these will then be assigned a character class and the next step would be to use a process of elimination to deduce which character the reassembled character most closely matches. The corresponding ASCII value will then be assigned.

Since the prime consideration of the proposed algorithm is the pattern recognition system, the image processing system is kept as simple as possible. Therefore, it is assumed that there is no horizontal or vertical overlapping between characters in the text. This is done to ensure the success of the segmentation algorithm which relies on the horizontal projection technique to determine lines of text and also the vertical projection technique to segment each line of text. There should also be no slanting of text and the quality of text should be good.

RESULTS OF IMPLEMENTATION OF THE ALGORITHM

The algorithm was tested on samples of words: A scanned version of an example text is shown in Fig. 7.

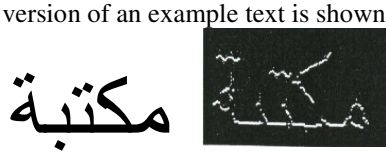


Fig. 7: Scanned image of the Arabic word

The image processing operations of binarization, segmentation and markings produced satisfactory results on the test data used. Each segment was divided up into strokes. But the types of strokes extracted varied as two constants were varied –

* PercentForOptimalSizeCursor

This constant can be varied from 0 to 1 and represents the percentage of pixels that are 'ON' in the cursor in question. It is used to set the limit for tolerance of the number of 'ON' pixels. For example, a value of 0.8 indicates a cursor may be enlarged to the point where the percentage of 'ON' pixels is 80% or less.

* PercentOn

This constant is used to set the limit for determining acceptable directions for movement. The direction is acceptable if the cursor in that position occupies at least *'PercentOn'*. For example, a value of 0.9 indicates that a cursor would have to have 90% of its pixels 'ON' to be able to move there.

Fig. 8 shows how by different combinations of the two constants, different number and types of strokes are obtained.

The reason for the large number of strokes is the combination of percentage conditions applied to the cursor. There are probably areas with uneven writing where the cursor ends up in an area, which indicates an end point, which in fact is a point where the cursor does not fit, given the rules for movement. So the cursor ends up determining a larger number of smaller strokes. But these strokes, when displayed on screen, do not give a very accurate picture of the character piece. So with certain combination of percentages, a relatively smaller number of strokes are obtained, which give a more accurate picture of the character piece. It was noted that at particular combinations of percentages, a very large number of strokes are obtained resulting in a heap overflow error. As far as a dot or dots were concerned, they were, at most levels of the two constants, recognized as a dot and in the case of two dots, as a straight line. The character classification module was not implemented due to problems in the feature extraction module.

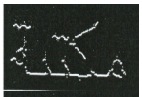
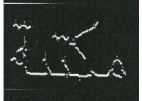
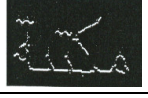
% For Optimal Cursor Size	Percent On	No of Strokes	Resulting Storks
1	1	27	
0.9	1	29	
0.8	0.9	36	

Fig. 8: Combination of Two Constant

CONCLUSION

This paper presented a new approach for Arabic character recognition based on producing a logical dynamically sized cursor to traverse the image of the Arabic word. The cursor identifies the different possible strokes and calculates some values functions. After some trials, the thinning algorithm approach was disregarded because it caused the cursor to produce un-definable stokes. The research demonstrates that the identifying the directional vectors of the strokes of the Arabic characters within the word is the best way forward for Arabic OCR's. Further work is required to recombine strokes into characters where each stroke should be added to neighboring strokes through available connection points until it fits into a character class. Building a database of character classes such as characters that have a closed loop or a half open upwards circle is another future research direction.

REFERENCES

1. Khorsheed M. S. 2002. Off-line Arabic character recognition-a review. Pattern Analysis and Applications, Vol. 5, Issue 1: 31-45
2. Alma'adeed S., Higgens C., Elliman D., Kasturi R., Laurendeau D., and Suen C. 2002. Recognition of off-line handwritten Arabic words using hidden Markov model approach, Proceedings 16th International Conference on Pattern Recognition. IEEE Comput. Soc, Los Alamitos, CA, USA. Vol. 3: 481-4
3. Menhaj M. B. and Adab M. 2002. Simultaneous segmentation and recognition of Farsi/Latin printed texts with MLP. Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290). IEEE, Piscataway, NJ, USA. vol.2: pp: 1534-9
4. Kavianifar M. and Amin A. 1999. Preprocessing and structural feature extraction for a multi-fonts Arabic/Persian OCR. Proceedings of the Fifth International Conference on Document Analysis and Recognition. ICDAR '99 (Cat. No.PR00318). IEEE Comput. Soc, Los Alamitos, CA, USA. pp: 213-16
5. Sari T. and Sellami M., 2002. MOrpho-LEXical analysis for correcting OCR-generated Arabic words (MOLEX)". Proceedings Eighth International Workshop on Frontiers in Handwriting Recognition. IEEE Comput. Soc, Los Alamitos, CA, USA. pp: 461-6
6. Sari T., Souici L. and Sellami M. 2002. Off-line handwritten Arabic character segmentation algorithm. ACSA, Proceedings Eighth International Workshop on Frontiers in Handwriting Recognition. IEEE Comput. Soc, Los Alamitos, CA, USA, pp: 452-7
7. Cowell J., Hussain, F., Hamza M. H., and Sarfraz M. 2001. Extracting features from Arabic characters, Proceedings of the IASTED International Conference Computer Graphics and Imaging. ACTA Press, Anaheim, CA, USA, pp: 201-6
8. Cowell J. and Hussain F., 2002. A fast recognition system for isolated arabic characters", Proceedings Sixth International Conference on Information Visualisation. IEEE Comput. Soc, Los Alamitos, CA, USA, pp:650-4
9. Cheung M. A., Bennamoun M. and Bergmann N. W., 2001. An Arabic optical character recognition system using recognition-based segmentation, Pattern Recognition. Vol. 34, Issue 2: 215-33
10. Al-Ohali Yousef, Cheriet Mohamed and Suen Ching 2004. Databases for recognition of handwritten Arabic cheques, Pattern Recognition, Volume 36, Issue 1: 111-121.