Research Journal of Applied Sciences Engineering and Technology 2(5): 428-435, 2010
ISSN: 2040-7467

Submitted Date: May 05, 2010 Accepted Date: June 14, 2010 Published Date: August 01, 2010

On Multiple Typeface Arabic Script Recognition

Abdelmalek Zidouri
Department of Electrical Engineering, King Fahd University of Petroleum and Minerals,
KFUPM - 1360, Dhahran 31261, Saudi Arabia

Abstract: In this study, we propose a new sub-word segmentation and recognition scheme, which is independent of font size and font type. Different ways of recognition are attempted namely Neural Net, template matching and principal component analysis. Results show that the real problem in Arabic character recognition remains the challenging separation of sub-words into characters. The system is realized in a modularized way. The combination of the different modules forms the basis of a complete Arabic OCR system. A successful preprocessing stage is reported. Unlike Latin based languages, recognition of printed Arabic characters remains an open field of research.

Key words: Arabic character recognition, cursive script, multi-font, segmentation

INTRODUCTION

Character recognition is one of the oldest fields of research. It is the Art of automating both the process of reading and keyboard input of text in documents. A major part of information in documents is in the form of alphanumeric text. However characters' automatic recognition is not an easy problem, especially for languages using the Arabic script. Arabic is written cursively (connected) even when machine printed or typed. It is becoming increasingly important to have information available for editing, examination and manipulation in a format, which is recognized by computers such as ASCII characters or UNICODE characters. A large quantity of Arabic documents exists in image format, which cannot be edited by computers. Similarly searching and indexing cannot be easily provided as compared with textual data. Compared to research in the area for other languages, the publications on Arabic character recognition are relatively scarce, Khorsheed (2007, 2002), Hamami *et al.* (2002), Amin (1998), Zidouri *et al*. (1995) and Mahmoud (1994).

In this study, we have proposed an offline Arabic OCR system, which converts printed Arabic document images into textual form. Thus it provides a means of data entry for computers in which the user needs virtually no training. This allows for storage of larger quantities of data, improved access to the data required, easy handling of data and reduction of costs.

BACKGROUND

Arabic language is one of the most ancient languages and spoken by many people in areas around the globe. The Arabic script and language have resisted any major change for centuries now. Text written or words used more than 1000 years ago are still being used and understood by schoolboys around the Arab world. Nevertheless, with the advent of computer age and information technology, efforts have been directed to adapt the Arabic script for ease of use with the new tools. One such effort has been concerned with automating of the handling of Arabic characters and text. This effort is faced with the usual problems of character recognition in general in addition to problems that are specific to Arabic language only. Arabic presents some specific characteristics that are worth noting for the English reader.

- Arabic is written from right to left
- It is composed of 28 characters
- The characters change shape depending on their position in a word
- They can be grouped in 100 character shapes
- They present a lot of similarities and composed of many loops and cusps, Table 1.
- Characters are connected even when typed or printed
- Two kinds of spaces, between words and within a word introduced by characters that have no middle shape MF

Table 1 shows the complete set of the Arabic Alphabet characters in their different shapes: Isolated or form (IF), at the beginning of a word or (BF), in the middle of a word or (MF), and at the end of a word or (EF).

MATERIALS AND METHODS

In this study, we have proceeded to the development of an Arabic OCR system, in a modularized way. Our system composed of several stages, and within each stage we tackle one specific problem. A block diagram of the system is shown in Fig. 1.

Table 1: 28 Arabic characters and their forms

IF	BF	MF	EF	IF	BF	MF	EF
أ			ـا	ض	ضـ	ضـ	ض
ب	بـ	بـ	ب	ط	طـ	طـ	ط
ت	تـ	تـ	ت	ظ	ظـ	ظـ	ظ
ث	ثـ	ثـ	ث	ع	عـ	عـ	ع
ج	جـ	جـ	ج	غ	غـ	غـ	غ
ح	حـ	حـ	ح	ف	فـ	فـ	ف
خ	خـ	خـ	خ	ق	قـ	قـ	ق
د			د	ك	كـ	كـ	ك
ذ			ذ	ل	لـ	لـ	ل
ر			ر	م	مـ	مـ	م
ز			ز	ن	نـ	نـ	ن
س	سـ	سـ	س	ه	هـ	هـ	ه
ش	شـ	شـ	ش	و			و
ص	صـ	صـ	ص	ي	يـ	يـ	ي

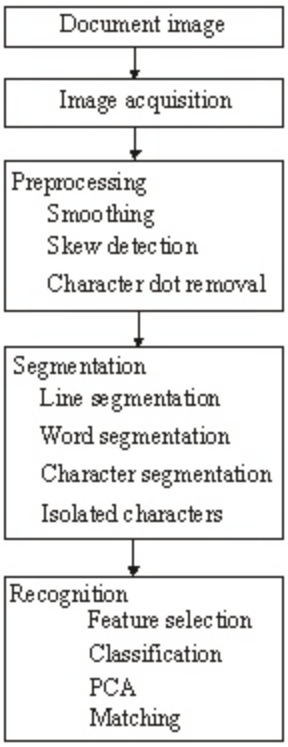


Fig. 1: System block diagram

Preprocessing: We have assumed that document images will be provided to the system. A general-purpose scanner will perform the digitization of the document image. After digitization, a gray level document image is obtained. For ease of processing we convert it to a binary image.

Binarization: In order to process it effectively we need to binarize the image. There are many techniques proposed in the literature for binarization or automatic thresholding. Most of the techniques utilize the histogram of image. The intensity value, which has the maximum variance is selected as a threshold value. If there is not much difference in terms of variance among intensity levels then this technique suffers from under thresholding. This shortcoming is addressed by Abutaleb (1989). It performs two level thresholding. At first level the thresholding is performed using Otsu method, Otsu (1979). After thresholding, connected components algorithm is applied and second level thresholding is performed on each location of each connected components in original gray level image. One of the main drawbacks of this scheme is the computational complexity.

Noise removal: Character dots in Arabic carry important significance in discrimination of similar shape characters. More than half of Arabic characters of the Alphabet carry one, two or three dots each. On the other hand existence of "dots-like" pattern, on any scanned document image, is almost unavoidable. We used a knowledge-based threshold to remove isolated pixels or components having very small number of pixels in the image so that, it would not be confused to be character dots. This was chosen carefully not to remove any genuine character dots in the process.

Skew estimation: Skew Estimation is one of the mandatory preprocessing steps in OCR systems. We addressed this issue successfully in, Sarfraz *et al*. (2005a, b). There, a novice approach to skew estimation is introduced where multiscale properties of an image are utilized together with Principal Component Analysis (PCA) to estimate the orientation of Principal axis of clustered data. Here the image is decomposed into detail sub-bands. The energy distribution of wavelet-transformed signal is then estimated by using PCA.

Segmentation: Segmentation is the most important phase in Arabic OCR systems. The better the segmentation the better will be the recognition and finally better results, Zidouri *et al*. (2005). In our approach the segmentation phase starts with line segmentation, then word segmentation and finally character segmentation.

In line segmentation the input image of the document is segmented into lines of text. This is achieved by reading the horizontal projection of pixels in the document. If the projection becomes zero this means that a line of text is finished and subsequently the line of text is segmented out of the image.

In word segmentation 8-neighbor connected component algorithm is used for segmenting sub-words from a single line of text. In this algorithm the 8 groups of neighboring pixels of every component of image is evaluated to check if a word is ending. If a group of black pixels is surrounded by white pixels from its 8 neighbors then the set of pixels is segmented as a sub-word.

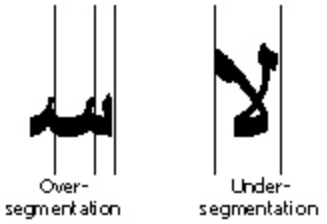


Fig. 2: Common problems in Arabic character segmentation

Arabic characters are connected on the baseline. In order to dissect sub-word into characters, we exploit the fact that, junction point between connected characters lies at baseline. But in certain cases like case of a character "س", it will suffer from "over segmentation", in other cases, some words may have overlapping characters such as "لا"and thus suffer from "under segmentation" as in Fig. 2. Even some segmentation techniques, might work for one font but fail to segment words if font type or size is changed. We have developed a general technique, which is independent of font size and font type.

Consider the following notation:

θ	=	Width of single dot in the document.
Ls	=	Width of smallest character
Ls'	=	Width of two smallest character if appear together
Lm	=	Maximum Width of character in isolated form
B(x, y)	=	Location of Baseline
I	=	Image of sub-word
I'	=	Image of sub-word without dots
E	=	Empty image of size I.

To improve segmentation efficiency we opted to remove stress marks like dots from characters. Their original position is remembered and reintroduced only in the recognition phase. In order to remove dots from sub-word, we have employed connected component approach with 8 neighbors.

Steps in character segmentation:

- Skeletonize I'
- Scan from right to left in row-wise fashion, to find a band of horizontal pixels having length >= Ls
- Take vertical projection on the scanned band found in step 2. If no pixel is encountered, draw a vertical guide band on E.
- Use special mark for the guide bands, which are drawn due to the scanned band (found in step 2) below the baseline B (x, y).
- Repeat the procedure, for all the rows.

After performing above-mentioned steps, an image E with several guide bands is obtained. In order to select, correct guide band for sub-word dissection, we extract several features from each guide band:

Feature	Description
F1	Width of the guide band
F2	Distance from 1st predecessor from right, zero in case of 1st guide band
F3	Distance from 2nd predecessor from right, zero in case of 1st and 2nd guide band.
F4	1 if guide band drawn due to scanned band is above baseline 0 if guide band drawn due to scanned band is below baseline
F5	Midpoint of guide band

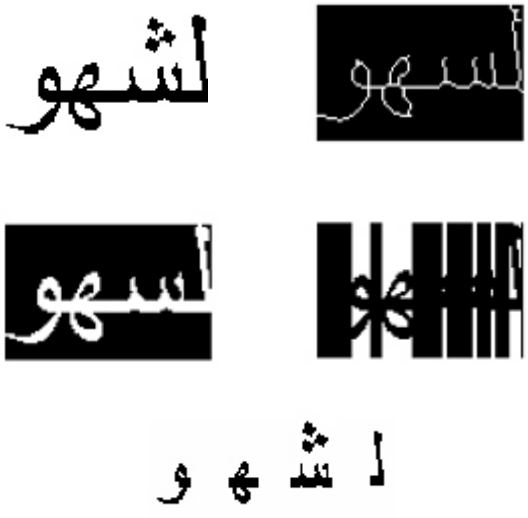
The judicious selection of guide band is driven through several rules. The feature sets {F1...F5} of each guide band are tested for each rule. If it satisfies rules then it is selected otherwise it is rejected.

Rule 1 : Choose guide band having highest relative width (F1) and F4 = 1
Rule 2 : Choose guide band if F2 > L_s and F4 = 1
Rule 3 : Choose guide band if F2 <= Ls and F3 > L_s' and guide band is not the last one.
Rule 4 : Choose guide band if F1 >= L_m and F4 = 1

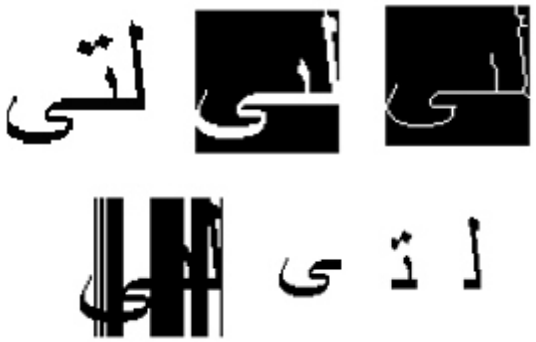
For the 1st guide band in the sets, even if it fails to qualify Rule 1-4 and the guide band next to it satisfies Rule 2 then it should be selected.

If all the guide bands fail to satisfy any rule, then we apply less constrained rule base i.e., removing F4 condition except Rule 4.

Example: Fig. 3.



Sub-word with character "ش" (3 dots above)



Sub-word with character "ت" (2 dots above)
Fig. 3: Example of Arabic characters with dots

DISCUSSION

This segmentation scheme has been implemented in a MATLAB environment. Results are quite promising. Few points need special care:

- The problem of character overlapping for some Arabic fonts causes some under segmentation of some characters like the special character "ال" composed of two characters "ل" and "ا"
- The problem of ligature for *Arabic traditional font* generates also under segmented characters.

These problems are solved by considering some group of characters that always appear together like the character "ال" above, as a separate class. Some other miss-segmentation will be dealt with in the recognition stage. The segmentation is not an aim by itself; some characters can be classified in a first run during the segmentation process by simple matching. Arabic words are sometimes composed of groups of connected and non-connected portions that we refer to as sub-words. Sub-words can be composed of one or more characters. For example the word "ابتسامة" is composed of 3 sub-words of -from right to left- 1, 4 and 2 characters each. So segmentation of sub-words is applied only to those sub-words that are composed of more than one character.

Recognition: It is well established in literature that recognition can be performed at one go, or what is known as free segmentation methods, Cheung *et al.* (2001), Al-Badr *et al.* (1995) and Zidouri *et al.* (1995). These methods usually perform well for one font or for word recognition. Eventually the segmentation problem for Arabic remains the main problem to be solved. We proposed to use for multifont, the recognition at two levels i.e. after word segmentation and after sub-word segmentation. Just after the result of segmentation, some machine learning technique for instance neural net is applied on training set. Nawaz *et al.* (2003) used 28 classes for training. He maintained 5 images per class and utilized 7 Hu's moment invariants as features, Hu (1962). Using nonlinear combinations of geometric moments he derived a set of invariant moments, which has the desirable property of being invariant under image translation, scaling and rotation. The central moments, which are invariant under any translation, are defined as

$$M_{pq} = \sum_{x=0}^{m}\sum_{y=0}^{n} (x-\bar{x})^p (y-\bar{y})^q f(x,y)\,dxdy$$

$$M_{pq} = \sum_{x=0}^{m}\sum_{y=0}^{n} x^p x^q f(x,y)\,dxdy$$

where the image size is m x n.

The set of moment invariants that have been defined by Hu (1962) are given by:

$$\phi_1 = M_{20} + M_{02} \quad (1)$$

$$\phi_2 = (M_{20} - M_{02})^2 + 4M_{11}^2 \quad (2)$$

$$\phi_3 = (M_{30} - 3M_{12})^2 + (3M_{21} - M_{03})^2 \quad (3)$$

$$\phi_4 = (M_{30} + M_{12})^2 + (M_{21} + M_{03})^2 \quad (4)$$

$$\phi_5 = (M_{30} - 3M_{12})(M_{30} + M_{12})\left[(M_{30} + M_{12})^2 - 3(M_{21} + M_{03})^2\right] + (3M_{12} - M_{03})(M_{21} + M_{03})^*\left[3(M_{30} + M_{12})^2 - (M_{21} + M_{03})^2\right] \quad (5)$$

$$\phi_6 = (M_{20} - M_{02})\left[(M_{30} + M_{12})^2 - (M_{21} + M_{03})^2\right] + 4M_{11}(M_{30} + M_{12})(M_{21} + M_{03}) \quad (6)$$

$$\phi_7 = (M_{30} - 3M_{12})(M_{30} + M_{12})\left[(M_{30} + M_{12})^2 - 3(M_{21} + M_{03})^2\right] + (3M_{12} - M_{03})(M_{21} + M_{03})^*\left[3(M_{30} + M_{12})^2 - (M_{21} + M_{03})^2\right] \quad (7)$$

To improve the accuracy of recognition, we added four more features from Hu' (1962) extended feature. The four more features are described in Hamami *et al.* (2002). The extended moments are given as follows:

$$\phi_8 = \left(M_{20}M_{02} - M_{11}^2\right) \Big/ M_{00}^4 \quad (8)$$

$$\phi_9 = \frac{\left(M_{30}^2M_{03}^2 - 6M_{30}M_{21}M_{12}M_{03} - 4M_{30}M_{12}^3 + 4M_{21}^3M_{03} - M_{21}^2M_{12}^2\right)}{M_{00}^{10}} \quad (9)$$

$$\phi_{10} = \frac{\left(M_{20}\left(M_{21}M_{03} - M_{30}^2M_{03}^2 - M_{12}^2\right) - M_{11}\left(M_{30}M_{03} - M_{21}M_{12}\right) + M_{02}\left(M_{30}M_{12} - M_{21}^2\right)\right)}{M_{00}^7} \quad (10)$$

$$\phi_{11} = \left(M_{20}^2M_{03}^2 - 6M_{20}^2M_{11}M_{12}M_{03} - 4M_{20}^2M_{02}M_{21}M_{03}\right) \quad (11)$$

Recognition using neural networks: Back propagation MLP neural net with three layers (input, hidden and output) containing 10 nodes each was implemented Fig. 4. But the result was not satisfactory and recognition rate was 75%. The syntactic approach gave more accuracy but not for all types of fonts. The matching approach was used to give high accuracy.

In order to improve recognition rate, we performed character recognition at two different levels that improved recognition rate considerably.

First level recognition: There are some characters in Arabic language, which are written in isolated form within a word or sub-word according to certain rules. For instance "ا" 'alif' appears in its isolated form if preceded by any of the letters that do not connect from the left side or if it is the first letter in a word. Similarly letter "و" which is also used as a word by its own meaning "and", is written in isolated form. The others are "د", "ذ", "ر" and "ز". Instead of recognizing these isolated characters at later stage, i.e. after sub-word segmentation, we employed similarity matching after word segmentation directly.

Each Arabic character can appear in four different shapes/forms depending on the position of the word (Beginning form BF, Middle form MF, Isolated form IF and End form EF). Table 1 shows the set of 28 Arabic characters in their different forms.

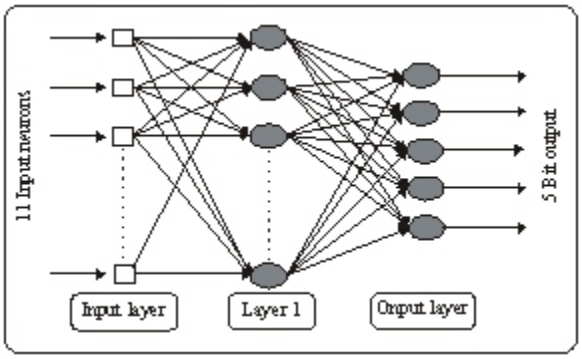


Fig. 4: Three layered radial basis function network

Instead of recognizing isolated characters at later stage i.e. after sub-word segmentation, we employed similarity matching after word segmentation. We utilized 30 isolated characters for 1st level recognition (28 listed in the Table 1 under Column IF, plus 2 other characters which are actually a combination of two characters which appear in a special connection that we prefer to deal with them as one single character).

We matched the words against a set of 30 isolated characters of Arabic. In exact template matching, query and template words are aligned based on the location of centroid. Centroid of a binary image $C(\bar{x},\bar{y})$ can be found as follows:

$$\bar{x} = \sum_{j=1}^{m}\sum_{i=1}^{n} i.I_{i,j} \Big/ \sum_{j=1}^{m}\sum_{i=1}^{n} I_{i,j}$$

$$\bar{y} = \sum_{i=1}^{n}\sum_{j=1}^{m} j.I_{i,j} \Big/ \sum_{i=1}^{n}\sum_{j=1}^{m} I_{i,j}$$

where I is the binary image of resolution *m* x *n*.

The example below illustrates the method of exact template matching. Without loss of generality assume that size of query word is greater than that of template word. Query word (on left) Fig. 5 and template word (on right) are aligned based on the location of centroid. In other word, centroid of template word is placed on the centroid of query word and "XOR" operation is performed to find number of pixels that are matched. Ratio between the number of pixels matched and total number of pixels in smaller image (template word) is calculated. This ratio is compared with a threshold value (determined experimentally) to decide match or mismatch. Although query and template word are quite different but their

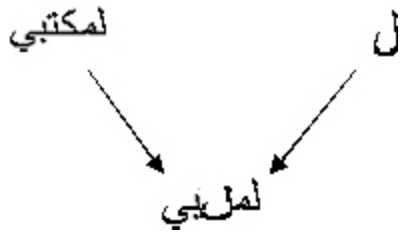


Fig. 5: Alignment based on centroid

match ratio will be close to 1 and hence it will be considered incorrectly as a match, thus this technique occasionally fail to recognize composite word.

To overcome the above-mentioned difficulty we employed similarity match. A set of features are extracted from character in question and matched against feature set of pre-stored template images. If the similarity match is less than a fixed threshold then it is considered to be a match otherwise we discard it and it is passed to sub-word segmentation stage. Similarly each and every character is tested. In order to re-generate the document, we also stored the spatial location of isolated characters based on page number, line number and word number. Our feature set includes these measures:

F_1 = White pixel ratio = Number of white pixels / Size of image
F_2 = Black pixel ratio = 1 - F1
F_3 = Orientation of white pixels (in radians)
F_4 = Aspect ratio = Height / width

Euclidean distance between feature vectors of query and template word is calculated as follows:

$$D(Q,T) = \sqrt{\sum_{i=1}^{4} \left|F_i(Q) - F_i(T)\right|^2}$$

Where F(Q) and F(T) are feature vectors of query and template word respectively. Similarly in this way Euclidean distance of query word is calculated with all the template word. Let $D(Q,T_n)$ be the Euclidean distance between query word and its nearest neighbor template word. If $D(Q,T_n)$ is less than a fixed threshold then it is considered to be a match otherwise we pass it to sub-word segmentation stage. Similarly each and every character is tested. In order to regenerate the document, we also store the spatial location of isolated characters based on page number, line number and word number.

Second level recognition: As mentioned in the previous section, the words which are not recognized at 1st level recognition are passed to sub-word segmentation module.

a b c d

Fig. 6: Separation of Dots from the character body

Segmented characters are recognized at this level. In order to improve recognition rate, we proposed a new set of classes. Instead of recognizing characters at one go, we categorized similar shaped characters in one class. A connected "ٮ", "ٹ", "ٹ" "ٹ", "ٮ" are put together in same group as a matter of fact they only differ in number and/or position of dots. In order to recognize a character shape, we removed dots from it whether it is above, below and in between. Dots depict very important information and if we change the position or number of dots then semantic of character will change completely. Thus we need to recognize number of dots and their position. For instance, consider the following sets of characters:

{ ح , ج , خ }, { ڈ , ڈ , ٮ , ٮ , ٹ , ٹ }, { ڈ , ڈ } { ڑ , ڑ }

In each set of characters, the shape and structure of characters are same. The only difference is the position and number of dots, which can be considered as a local feature. In order to increase the recognition rate considerably, we proposed to recognize structure of character (utilizing global feature) and then apply local feature (position and number of dots) to recognize the complete character.

Connected Component Algorithm is applied on segmented characters. We extract necessary information to distinguish 'dot' from 'structural part'. Consider the Fig. 6a, which is the character to be recognized. This image undergoes to "Connected Component Algorithm" with 8 neighbors. As a result we obtained two non-connected components as shown in the Fig. 6b. Component with higher number of pixels will be the structural part, while the remaining components will be the stress mark or dots. The "dotless" or structural parts of segmented characters are passed to PCA based classifier.

We store necessary features, regarding the dots such as number of dots and their position to be able to associate them back to their original structures. We have identified three positions.

- Dots can be above the structure parts such as in the case of 'ت'.
- Dots can be below the structure parts such as in the case of 'ب'.
- Dots can be inside the structure parts such as in the case of 'ج'.

Recognition using PCA: We modified class structure of training set and proposed different classes, which are

based on similarity of shapes rather than different forms of same characters. Segmented characters obtained from the module need to be recognized. There have been several approaches proposed in literature and the recognition rate was not high. In order to improve recognition we employed "Principle Component Analysis" for feature extraction and "Nearest Neighbor Algorithm" for recognition of class. The dots are removed first and the dotless structure is passed to the PCA module for feature extraction.

The principal components capture the most statistical variances in the least squared sense; therefore can be used to represent the data in a lower dimension. This is a popular technique in dimension reductions. A pattern x in a test set is recognized by projecting down to the feature space, followed by nearest neighborhood classification.

We implemented our PCA based classifier with a data set, which includes 20 classes with 10 samples each. So our database comprises 200 binary images of 32x32 resolution. We need to up-sample the data so as to enlarge it. Bi-cubic interpolation was utilized to interpolate the data. We found very interesting result. When trained with 6 samples per class, the recognition rate of character shapes was 80%, which reached up to 90% when trained with 7 samples per class and tested with 3 character images. Some misclassification occurred due to segmentation error.

First stage recognition using weighted similarity match produced recognition of 98% and the recognition at second level for non-isolated characters results in 90% recognition where errors are mainly due to segmentation.

CONCLUSION

OCR systems are of immense importance when large amount of documents need to be edited or search operation is needed in these documents. The ultimate goal of this field of research is a complete Arabic OCR product for the end user. This requires the efforts and contributions of groups and individual researchers. This system is for Arabic documents and for more than one font. Actually we tested our system with the following fonts:

- Simplified Arabic,
- Arabic Transparent,
- Simplified Arabic Fixed and
- Arabic Matin

An efficient segmentation of printed Arabic text into characters is considered. As segmentation and recognition are closely dependent of each other we get some reasonable experimental results.

This study has also shown that Arabic OCR problems will not be solved by one method but by the combination of many methods of recognition and segmentation. Combination of the best of each method structural or statistical, local or global features extraction will yield the required target. Whether using matching or Neural Network or Moment Invariants for classification and recognition we need knowledge-based techniques to improve the recognition rate. Arabic OCR has many aspects, and a lot still needs to be done in this field.

ACKNOWLEDGMENT

The author would like to thank King Fahd University of Petroleum and Minerals for the support.

REFERENCES

Abutaleb, A.S., 1989. Automatic thresholding of gray-level pictures using two dimensional entropy. Comput. Vision Graph., 47: 22-32.

Amin, A., 1998. Off-line Arabic character recognition: the state of the art. Pattern Recogn., 31(5): 517-530.

Al-Badr, B. and R. Haralick, 1995. Segmentation-Free word recognition with application to Arabic. Proceedings of 3rd International Conference on Document Analysis and Recognition. Montreal. Aug. 14-15, pp: 355-359.

Cheung, A., M. Bennamoun and N.W. Bergmann, 2001. An Arabic optical character recognition system using recognition-based segmentation. Pattern Recogn., 34: 215-233.

Hamami, L. and D. Berkani, 2002. Recognition System for Printed Multi-Font and Multi-Size Arabic Characters. Arabian J. Sci. Eng., 27(1B): 57-72.

Hu, M.K., 1962. Visual pattern recognition by moment invariant. IRE T. Inform. Theory, 8: 179-187.

Khorsheed, M.S., 2002. Off-line arabic character recognition-a review. Pattern Anal. Appl., 5: 31-45.

Khorsheed, M.S., 2007. Offline recognition of omnifont arabic text using the HMM Tool Kit (HTK). Pattern Recogn. Lett., 28(12): 1563-1571.

Mahmoud, S.A., 1994. Arabic character recognition using fourier descriptors and character contour encoding. Pattern Recogn., 27(6): 815-824.

Nawaz, S.N., M. Sarfraz, A. Zidouri and W. Al-Khatib, 2003. An approach to offline arabic character recognition using neural networks. Tenth International Conference on Electronics, Circuits and Systems, ICECS Sharjah, UAE. Dec. 14-17.

Otsu, N., 1979. A threshold selection method from gray-level histograms. IEEE Tran. Sys. Man Cybernet., 6(1): 62-66.

Sarfraz, M., A. Zidouri and S.A. Shahab, 2005a. A novel approach for skew estimation of document images in ocr system. IEEE proceedings of the International Conference on Computer Graphics, Imaging and Vision (CGIV 2005), Beijing, China, Jul. 26-29.

Sarfraz, M., A. Zidouri and S.N. Nawaz, 2005b. On Offline Arabic Character Recognition. Computer-Aided Intelligent Recognition Techniques and Applications. Sarfraz, M. (Ed.), ISBN: 0-470-09414-1, John Wiley and Sons, Ltd., pp: 1-18.

Zidouri, A., S. Chinveeraphan and M. Sato, 1995. Recognition of machine printed arabic characters and numerals based on MCR. IEICE T. Inform. Sys., E78-D(12): 1649-1655.

Zidouri, A., M. Sarfraz, S.A. Shahab and S.M. Jafri, 2005. Adaptive dissection based subword segmentation of printed arabic text. Ninth International Conference on Information Visualisation (IV'05). London, England, Jul. 06-08, pp: 239-243.