Optimizing Feature Selection for Recognizing Handwritten Arabic Characters

Mohammed Z. Khedher, Gheith A. Abandah, and Ahmed M. Al-Khawaldeh

***Abstract*—**Recognition of characters greatly depends upon the features used. Several features of the handwritten Arabic characters are selected and discussed. An off-line recognition system based on the selected features was built. The system was trained and tested with realistic samples of handwritten Arabic characters. Evaluation of the importance and accuracy of the selected features is made. The recognition based on the selected features give average accuracies of 88% and 70% for the numbers and letters, respectively. Further improvements are achieved by using feature weights based on insights gained from the accuracies of individual features.

***Keywords*—**Arabic handwritten characters, Feature extraction, Off-line recognition, Optical character recognition.

I. Introduction

Arabic writing system differs from European systems by some differences [1]. It is written from right to left and it is always cursive whether handwritten or printed. It contains 28 characters. Six of them can be connected only from the righthand side and have two shapes (connected and standalone). The rest can be connected from either or both sides. Hence, there are four shapes for these 22 characters according to the location of the character in the word (start, middle, end, or standalone). Some of the characters have secondary objects like a dot or combination of dots (one, two, or three). In fact, some characters can only be distinguished from others only by these secondary features. A comparison of various characteristics of Arabic, Latin and other languages are discussed in many references [2].

Many off-line Arabic character recognition techniques for printed text were published [3]. The state of the art of this type of recognition achieves good accuracy. There are even some successful commercial products for this application [4]. However, the off-line recognition of handwritten Arabic characters is more difficult and users are still waiting for reliable and accurate solutions.

Characters are recognized by human eyes via recognition of features associated with the characters. Human eyes are trained to the recognition of these features and associate features to corresponding characters. Hence as a result of this association, the human brain uses the feature information to recognize the character, word, sentence, and consequently the meaning may be understood. Features used for this recognition is language dependant. Hence, the purpose of this paper is to discuss features suitable for handwritten Arabic character recognition and to evaluate their usefulness and effectiveness in performing the recognition task.

M. Z. Khedher is with the University of Jordan, Faculty of Engineering, Electrical Engineering Dept., Amman 11942, Jordan, (phone: +962-6-535-5000/2813; e-mail: khedher@ju.edu.jo).

G. A. Abandah is with the University of Jordan, Computer Engineering Dept., (e-mail: abandah@ju.edu.jo).

A. M. Al-Khawaldeh has a BSc in Electrical Engineering from the University of Jordan, (e-mail: ahmed@hayyan.net).

II. Approach

To conduct our experiments, we used a database of handwritten samples collected from 48 different persons [5]. The 48 persons wrote the same text. Then we collected similar characters from the 48 persons into pages. Figure 1 shows the sample page for the character *Ain* (ع).

Figure 1: Forty-Eight-Sample Page for the Letter *Ain*

We conducted our experiments on a PC and we developed an application to host the feature extraction and recognition algorithms: Handwritten Arabic Character Recognition (HACR). The user interface for this experimental application is shown in Figure 2 and it is available on [6]. The processing and recognition of characters on HACR is done on three main stages: preprocessing, feature extraction, and recognition.

Images for handwritten text are first **preprocessed**, so that it is presented to later stages in a manageable form. Through this process, variations which do not affect the identity of the character of the image are removed.

Feature Extraction is the next stage. Extracted features should contain the useful information carried by the character image. The complexity of the handwritten cursive Arabic text requires using many features to make recognition possible.

Recognition is the process of evaluating the extracted features of an unknown character and comparing them with the features of the set of possible characters. The character

that is most similar to the unknown character is reported as the hit.

The features that we have selected are sufficient to achieve good recognition rate for all 10 numbers and 28 standalone letter shapes. After gathering data about the recognition rates, we analyzed the accuracy and effectiveness of each feature in recognizing these shapes. Gained insights were then used to improve the recognition accuracy.

III. Preprocessing

The input document image may not be a properly spaced plain-text document, and may contain noise and other unrelated objects. These problems and other similar problems have to be dealt with first.

To present the image to the next recognition stages in a suitable format, the following actions are performed:

1. Grid removal: to remove vertical and horizontal lines.
2. Noise removal: to convert the colored image into a clean black and white image.
3. Splitting: to split the image into single-character images. Figure 2 shows a sample image after HACR performed this third step.
4. Thinning: to reduce the width of the character to one pixel. We used Rosenfeld thinning algorithm [7] with some modifications to decrease the number of undesired branches.

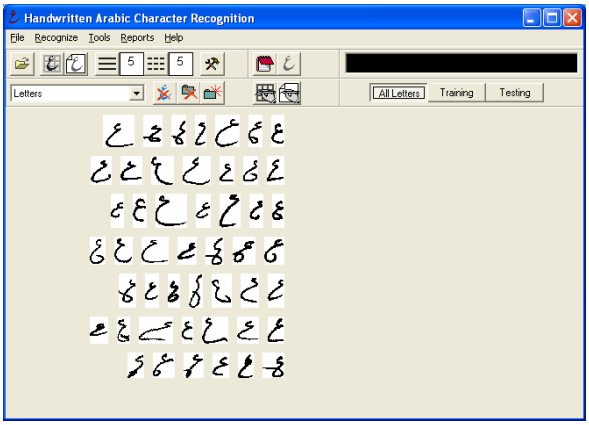

Figure 2: HACR showing the sample page after preprocessing.

IV. Feature Extraction

Character features are characteristics that distinguish one character from another. Humans use these features to recognize characters and text. These features may be classified into quantitative and qualitative features. Quantitative features include dot numbers, character height, character width, character area, and character weight above and below the baseline. Qualitative features include loops, branches, topological descriptions, topological relations, dot position, connection points, and junction points. The following is a list of the features we used in this study.

A. Width, Height, and Aspect Ratio

Since different writers write same characters in different sizes, the absolute width and height are not reliable features for recognizing handwritten characters. However, some Arabic characters are wider than others. Therefore, the aspect ratio (height/width ratio) is a useful feature.

B. Density

The character area is the total number of white and black pixels of the clipped character image. The density is the number of black pixels to character area ratio.

C. Secondary Objects

The Arabic writing system has five secondary objects: one dot, two dots, three dots, *Hamzah* (ء), *Maddah* (ˆ). If the vertical line of the characters *Tah* (ط) and *Zah* (ظ) is written disconnected from the main object of the character, then it is considered the sixth secondary. The location of the secondary object can be above, below, or within the main object, as in *Thal* (ذ), *Beh* (ب), and *Jeem* (ج), respectively. The image of the character is divided into main object and secondary objects. Then the type of each secondary and its location relative to the main object is found. In the Arabic language writing system, the location of the secondary object is very important. Some characters have the same main object shape, but differ in the location of the secondary object, e.g., *Jeem* (ج) and *Khah* (خ).

D. Closed Loops

There are two numbers (5 and 9) and nine Arabic characters (ص, ض, ط, ظ, ف, ق, م, ه, and و) that have closed loops. Moreover, there are six other characters that some writers write with closed loops (ج, ح, خ, ع, غ, and ك). This makes the closed loop feature an important feature in recognizing Arabic characters. We use two attributes related to the closed loops: the number of loops and the size of the biggest loop.

E. Filled Circles

Some of the small closed loops are sometimes written as filled circles (ف, ق, م, and و). This creates a problem in recognizing these characters because the thinning process eliminates these filled circles. Therefore, after thinning, a letter *Waw* (و) with a filled circle is undistinguishable from the letter *Reh* (ر). To solve this problem, we scan the character before thinning and find the size of the biggest filled circle. The filled circle is a useful feature.

F. Concavities

Arabic characters have strokes concaved to different directions, e.g., up (ب), right (ع), up-left (ر), etc. The size and direction of such concavities are important in recognizing Arabic characters.

G. Stroke Sequence

The stroke sequence method is used in online handwritten recognition [8]. The *stroke* is defined as the direction of the pen movement from one pixel to the next. The *stroke*

sequence is the sequence of directions that describe the character. This feature converts the two-dimensional image of the thinned character into a one-dimensional string of strokes.

For the off-line recognition of handwritten Arabic characters, the stroke sequence method need to be adapted for this purpose. Our method extracts one string of strokes from the thinned main object of the character.

This extraction is done by starting from a start point (usually an end point) and tracing the directions from one pixel to the next adjacent pixel. We use eight directions with codes 0 through 7. Direction 0 is East, Direction 1 is North-East, etc. A *joint* point is given the code 9*x*9 and a *cross* point is given the code 8*x*8; where *x* is the sequence number of that joint or cross. Finally, a *termination* point of a branch is given the code 909, as shown in the example of Figure 3.

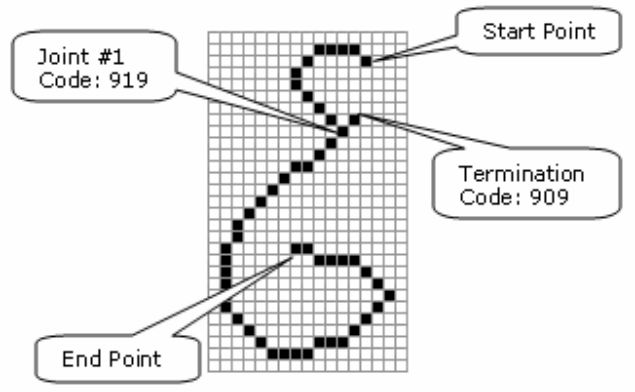

Figure 3: The stroke sequence string for the character *Ain* is 3444556777791919099195554555556566666777700010011113334443 4.

The stroke sequence string is useful to find other character features. We use it to count the number of joint, cross, and termination points in the primary object of the character. Therefore, our work is similar to other research ([9] and [10]) that uses features extracted from the skeleton to recognize handwritten Arabic characters.

V. Recognition Based on Features Matching

We used a basic character recognition approach that depends on feature matching between the unknown character and a database of character features. This database is built using the first 24 samples of each 48-sample page. The last 24 samples are used as the unknown characters in the recognition experiments to evaluate used features and recognition algorithms.

To reduce execution time, we first use some features to narrow down the number of *possible* characters. Consequently, we only compare the unknown character with a subset of the database characters.

A. Possible Characters

Narrowing down the search to a subset of possible characters is a critical step. If this is not done accurately, the excluded characters would include the correct match and the correct match will not be found. We limited the number of features used to narrow down the search to three features: secondary location, closed loop, and vertical secondary. These features were selected because they are accurately extracted.

Secondary Location: Depending on the location of the secondary object, the search is narrowed down to the group of characters that has no secondary objects (ا, ح, د, ر, س, ص, ط, ع, ل, م, ه, and و), a secondary above (ت, ث, خ, ذ, ز, ش, ض, ظ, غ, ف, ق, and ن), below (ب and ي), or within (ج and ك).

Closed Loops: Depending on the existence of the closed loop, the search is narrowed down to the group of characters that has or does not have a closed loop.

Vertical Line: The existence of this secondary object narrows the search to the letters *Tah* (ط) and *Zah* (ظ).

B. Feature Matching

The types of the various values of the features can be Boolean value, integer, or ratio. The unknown character's features are compared with the respective values of the possible characters. To facilitate the recognition process, the result of each comparison is normalized to real number values in the range 0 and 1. The normalized matching value is then multiplied by the weight factor of the respective feature. HACR enables the user to select these weight factors.

The sum of all feature weighted comparisons is found for each possible character. The character that has the greatest sum is recognized as the hit character.

C. Stroke Sequence Matching

The stroke sequence string is an important feature. It contains a lot of information about the character. We compare the string of the unknown character with the strings of the possible characters after scaling these strings. The purpose of scaling is to have strings of same length so that they can be compared. Scaling is done by periodically repeating some digits in the shorter string of the compared string pair.

The string comparison is done by finding the sum of the adjusted differences of the respective digits in the two strings. As the strokes have only 8 directions, the adjusted difference is found by the formula:

$$Difference = |Input\ Stroke - Possible\ Stroke| \quad (1)$$

$$\text{If } (Difference > 4) \text{ then } Difference = 8 - Difference \quad (2)$$

VI. Results and Discussion

HACR was first trained for the 10 numbers and 28 letters. Through this training, HACR built its database of character features using the first 24 samples of each 48-sample pages. The HACR was then used to recognize the last 24 samples of each page. This procedure was applied on numbers and letters separately.

Using equal weight factors for 18 features (not including the three features used to find possible characters), we got the recognition accuracies shown in the second column of Table 1. The shown percentages are averages of the recognition accuracies of the 24 samples of the 10 numbers and 28 letters. These recognition accuracies are smaller than 100% because of the high variations in the writers' writing styles. The

accuracy is better for numbers because they are less complicated than the letters.

TABLE 1
RECOGNITION ACCURACY

Category	Equal Weights	Proportional Weights	Categorized Weights
Numbers	87.6%	90.4%	90.0%
Letters	69.8%	71.7%	73.4%

Figure 4 shows samples of the variations noticed in the writing styles of different users. These samples show that loops are sometime introduced or filled and secondary objects are written in multiple styles.

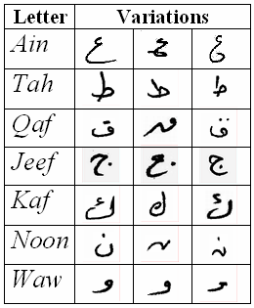

Figure 4: Variations in the Writing Styles of Different Writers

We noticed that some features are more accurate than other features. In other words, some features are less affected by variations in the writing style than other features. Table 2 shows the accuracy of the 18 used features. These accuracies were found by averaging the accuracy of each feature in matching the correct character over the 24 samples.

TABLE 2
ACCURACIES OF USED FEATURES

No	Feature	Numbers	Letters
1	Height/width ratio	78.6	75.3
2	Density	81.8	79.5
3	No of dots	100.0[a]	91.0
4	Hamzah existence	100.0[a]	98.4
5	Closed loops	96.4	80.1
6	Closed loop size	94.0	74.6
7	Filled circle size	69.3	68.0
8	Up concavity	83.8	65.4
9	Down concavity	88.4	84.7
10	Left concavity	83.3	71.6
11	Right concavity	69.7	80.5
12	Up-left concavity	94.2	83.9
13	Up-right concavity	96.1	76.8
14	Down-left concavity	98.8	95.6
15	Down-right concavity	87.7	96.8
16	Stroke sequence	75.4	73.8
17	Joint points	65.7	62.8
18	Cross points	100.0[a]	98.7

[a] These features do not exist in any number.

This insight into the features' accuracies motivated us to use unequal weight factors for the 18 features. We have experimented with many weight values. The detailed experimental results are on the HACR web page [6]. The third column of Table 1 shows the overall recognition accuracies for weights proportional to the accuracies of the used features. These weights improved the recognition accuracy. However, we were able to get even better accuracy for recognizing letters when we used categorized weights (as shown in the fourth column of Table 1). The used categorized weights are 5, 10, 20, 40, and 60 for feature accuracy ranges of 55-65, 66-75, 76-85, 86-95, and 96-100, respectively.

VII. CONCLUSION

Selecting proper features for recognizing handwritten Arabic characters can give good recognition accuracies. Further recognition improvements can be achieved by using insights from evaluating the importance and accuracies of selected features. We intend to expand our study to include connected character shapes and further optimize the feature set, weight factors, and matching order. As we use normalized values for the features, we can use them as input to fuzzy logic or neural network systems to get better results.

The stroke sequence is a powerful feature that can be further utilized to derive additional features like inflections. Better results can be achieved by improving the algorithms used in recognizing the secondary objects.

The large variations in the writing styles of some characters suggest that we should treat each variation separately. Therefore, the recognition system must be trained for the different variations separately so that it can identify the various variations accurately.

REFERENCES

[1] G. Abandah and F. Khundakjie, "Issues concerning code system for Arabic letters," *Dirasat Engineering Sciences J.*, vol. 31, no. 1, pp 165-177, April 2004.

[2] A. Amin, "Arabic character recognition," *Handbook of Character Recognition and Document Image Analysis*, pp 397-420, World Scientific, 1997.

[3] Y. Al-Ohali, "Development and evaluation environment for typewritten Arabic character recognition," M.Sc. thesis, King Saud University, Saudi Arabia, 1995.

[4] Sakhr Software OCR, http://www.sakhr.com.

[5] G. Abandah and M. Khedher, "Printed and handwritten Arabic optical character recognition – initial study," A report on research supported by The Higher Council of Science and Technology, Jordan, August 2004. Available: http://www.abandah.com/gheith.

[6] Handwritten Arabic Character Recognition, http://hacr.hayyan.net.

[7] R.Stefanelli and A.Rosenfeld, "Some parallel thinning algorithms for digital pictures," *J. of ACM*, vol.18, no.2, pp. 255-264, 1971.

[8] S. Cha, Y. Shin and S. Srihari, "Approximate stroke sequence string matching algorithm for character recognition and analysis," *Pattern Recognition and String Matching*, vol. 13, 2003.

[9] A. Saleh, "A method of coding hand-written Arabic characters and its application to context-free grammar," *Pattern Recognition Letters*, vol. 15, pp. 1265-1271, Dec. 1994.

[10] A. Amin, H. Al-sadouni, and S. Fischer, "Hand-printed Arabic character recognition system using an artificial network," *Pattern Recognition*, vol.29, no.4, pp. 663-675, April 1996.