A New Approach for Arabic Offline Handwriting Recognition

Mohammad Adnan Al-Alaoui, Mohammad Amin Abou Harb, Zeid Abou Chahine, and Elias Yaacoub

Abstract—A new approach for Arabic handwriting recognition is proposed. The proposed method is part of a larger software framework to teach Arabic reading and writing to illiterates. The method is customized to each letter of the Arabic alphabet. The characteristics of each letter are analyzed and the appropriate detection scheme for that letter is then determined. This allows the method to provide feedback to the user on the correctness of the character written. Furthermore, in case of incorrect writing, the method indicates what part of the letter was erroneously written. This feedback feature allows the user to enhance his handwriting the next time he writes the same letter. The target is to combat adult illiteracy in the Arab world by using Information Technology.

Index Terms—Adult illiteracy, Arabic language, Image processing, Offline handwriting recognition.

I. INTRODUCTION

The use of information and communications technology, ICT or IT for brevity, to combat illiteracy and to move illiterates directly from illiteracy to computer literacy was proposed in [1, 2]. Such an ambitious project necessitates the use of handwriting recognition for teaching illiterates basic writing skills.

Due to the complexity of the Arabic language, finding an accurate handwriting recognition approach that could be used for teaching illiterates and providing them with useful feedback to correct their mistakes is not an easy task. The Arabic language is composed of 28 main characters, each written in four versions: beginning, median, termination, and separated types. In addition to that, there are several letters in the Arabic alphabet that contain one, two, or three points [3, 4]. A successful handwriting recognition method should take into account all the possible locations of the point(s) and recognize the appropriate letter within an appropriate margin of error. Moreover, the fact that the user is an illiterate means that he does not control completely the movement of the pen, and thus, his handwriting. Therefore, the handwriting recognition tool should be able to recognize a letter of any size, not completely centered in the space assigned, and with some imperfections (a reasonable margin of error) [5].

To reach this end, we propose a method based on offline recognition, but that can be easily extended to online. Traditional techniques relied on neural networks (e.g. [3]) for recognition. In an educational system, the user, or “learner”, is asked to write a specific letter. Hence, the correct letter is known to the teaching software. Consequently, the recognition software does not need to recognize the typed word or letter from a set of letters or words, but rather indicate to the user if his input was correct or not, and if not, give useful feedback in order for the illiterate user to correct his mistakes. The novelty of the proposed approach resides mainly in this “feedback” characteristic, in addition to its versatility and ease of expandability to the case of general handwriting recognition (i.e. where the software needs to recognize the typed letter from a set of letters and is not aware of the correct result).

The paper is organized as follows. A review of the relevant literature is presented in Section II. Section III presents the key steps used in the proposed approach (definitions, detection of the characteristics of the letters, approximating the letters with lines, etc.) and groups these steps to build the proposed algorithm. In Section IV, some results showing the efficiency of the proposed method are displayed, and its advantages and disadvantages are discussed. Finally, conclusions are drawn in Section V and ideas for future work are presented.

II. LITERATURE REVIEW

The two main methods used for character recognition are offline and online handwriting recognition. Mostly they have been dedicated for the use of stroke-like languages such as English or Chinese, but very few have been applied to cursive language such as cursive Arabic or English characters [6]. They are mainly composed of a preprocessing phase followed by a recognition phase.

In online recognition system, the writing process is directly converted into a signal as a function of time. As the user writes on a graphic tablet, the signal is traced into coordinates in function of time: x(t) and y(t) [7]. The words are easily segmented into strokes. The strokes are then given as input to a recognition process that combines the different strokes into a coherent word based on a set of rules [6].
In offline character recognition, after scanning, the word is converted to a binary image or an iconic picture, with each set of pixels representing a certain feature (loop, line, etc.). Features could be further assigned characters. Processing techniques used for analysis can be found, for example, in [8-10]. A detailed overview of how characters are analyzed can be read in [7].

Traditional techniques relied on neural networks (e.g. [3], [11]) for recognition. Other techniques include principal component analysis (PCA) and Fisher linear discriminant (FLD) along with Bayesian classification, as used for example in [12] in order to detect handwritten digits.

The objective of the previous techniques is the correct detection of a character from a set of given characters. However, in an educational system, the user, or “learner”, is asked to write a specific letter. Hence, the correct letter is known to the teaching software. Consequently, the recognition software does not need to recognize the typed word or letter from a set of letters or words, but rather indicate to the user if his input was correct or not, and if not, give useful feedback in order for the illiterate user to correct his mistakes. We propose a method based on offline recognition, but that can be easily extended to online. The novelty of the proposed approach resides mainly in this “feedback” characteristic, in addition to its versatility and ease of expandability to the case of general handwriting recognition (i.e. where the software needs to recognize the typed letter from a set of letters and is not aware of the correct result), and to online handwriting recognition. The proposed method is based on the characteristics of each letter which include: loops, bulks, lines, and their relative positions and proportions. The Arabic alphabet with the different forms of each letter is presented in Fig. 1. The details of the proposed approach are detailed in the next section.

### III. Proposed Method

#### III.1 Definitions and Characteristics

In this section we define some of the concepts we will use throughout this paper, and we describe the procedures used to identify the characteristics of a letter.

**A. Pixel connectivity:**

This is a way to determine which pixels in the surrounding 8 pixels are turned on. This is done by assigning to each neighboring pixel a bit and if that bit is one then the corresponding pixel is on; otherwise, it is off. Fig. 2 explains the described scenario.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>Current Pixel</td>
</tr>
<tr>
<td>128</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
</tr>
</tbody>
</table>

In Fig. 2, if the pixel above the current pixel is on we add 16 to current pixel and if the bottom right pixel is on we add 128 and so on. An example making use of the values displayed in Fig. 2 is shown in Fig. 3. Note that in Fig. 3, only three pixels are connected to the current pixel of interest: the above pixel, the pixel to the right, and the pixel located at the bottom-left of the studied pixel. It should be mentioned that, in all the figures, the values on the horizontal and vertical axes represent the numbering of the image pixels, starting from the top-left corner. Note that in Fig. 3, index [value] = 82 = 2[corresponding to bottom-left] + 16[corresponding to above] + 64[corresponding to right] (see Fig. 2 for correspondence between direction and numbers added).

### Fig. 1. Arabic alphabet with the different forms of the letters.

### Fig. 2. Pixel connectivity.

### Fig. 3. Pixel Connectivity example.
B. Pixel connectivity map:

The pixel connectivity map is an indication of how many pixels of the surrounding eight pixels are turned on. An example of pixel connectivity map is displayed in Fig. 4, where the pixel of interest is connected to three neighboring pixels: bottom-right, above and left.

In Fig. 4, the value of the selected pixel is 3 because it is connected to three of its neighboring pixels [bottom-right, above and left]. It should be noted that this is a characteristic of a bulk (bulks will be defined next). We could also note that less bright pixels correspond to pixels with connectivity=2 and their index=2.

C. Bulks:

A bulk is a set of connected pixels having more than two connections (e.g. crossroad).

Finding Bulks: To find the bulks we first find the pixel connectivity map then retain all pixels with connectivity greater than or equal to 3. Now we simply number these different regions to obtain the bulks.

Bulks characteristics: Bulks basically have one characteristic that is the number of connections to that bulk and the actual (x,y) coordinate of the pixels connected to the bulk.

D. Loops:

A loop is an enclosed area of the image (e.g. the letter "O" has 1 loop).

Finding Loops: To find the loops we first find distinct regions in the picture using 4-connectivity map (i.e. a pixel is only considered connected to four of its eight neighbors namely up, below, left and right of it). This pinpoints all areas in the picture including the background; now all what is left to do is remove the background by removing the part that has pixels on the edge or the biggest area part. Fig. 5 shows an example where the first detected loop is the background and the second loop is the one of interest.

Finding the loop boundary: This is done by first isolating the loop then dilating it with a 3x3 structuring element, then subtracting the dilated image from the original image. Fig. 6 shows an example of a detected loop after isolating it from a correctly written letter "ط".

Loops characteristics: Loops are characterized by their convexity, angle of inclination (especially helpful in "ط"), area, aspect ratio, and number of bulks on the loop. Loops can be further subdivided into an upper part and a lower part, or left part and right part. This subdivision is useful for checking the connections made with the loop if any.

E. Asayeh:

This term is not related to the method by itself. It is only a notation used in upcoming examples to denote the vertical line (similar to a stick) drawn over the loop of the letter "ط" of the Arabic language. In fact, Asayeh is the Arabic translation of stick.
III.2 The Use of Moments

This Section describes the use of moments to find the inclination of a certain object (e.g., loop). By inclination we mean the angle that the line of minimum inertia makes with the horizontal. And that angle is given by [8]:

\[ \theta = \tan^{-1}\left( \frac{2U_{xy}}{U_{xx} - U_{yy} + \sqrt{(U_{xx} - U_{yy})^2 + 4U_{xy}^2}} \right) \]  

In (1), \( U_{xy} \) is the 2nd central moment in x and y, \( U_{xx} \) is the 2nd central moment in x, and \( U_{yy} \) is the 2nd central moment in y.

The terms used in (1) are defined as follows:

\[ U_{xy} = U_{11} = \sum_{x} \sum_{y} (x - \bar{x})(y - \bar{y}) \times I(x, y) \]  

\[ U_{xx} = U_{20} = \sum_{x} \sum_{y} (x - \bar{x})^2 \times I(x, y) \]  

\[ U_{yy} = U_{02} = \sum_{x} \sum_{y} (y - \bar{y})^2 \times I(x, y) \]  

The second moments or \( U_{20}, U_{02}, \) and \( U_{11} \) are a useful tool used to determine the orientation of certain parts of the letter specifically loops and lines. They are derived according to the general equation [8]:

\[ U_{mn} = \sum_{x} \sum_{y} (x - \bar{x})^m (y - \bar{y})^n \times I(x, y) \]  

Where \( I(x,y) \) is the pixel value at \((x,y)\) in binary values (i.e. 1 for white, 0 for black).

Second order moments defined in Eq. (2) to (5) make use of the Zeroth and 1st moments. The Zeroth moment or A represents the area of the object or how many on pixels it has. This is crucial in determining the relative sizes of different parts of the letter. It is defined by [8]:

\[ A = \sum_{x} \sum_{y} I(x, y) \]  

The 1st moments or \((\bar{x}, \bar{y})\) represent the position of the center of mass of the object. They constitute a useful tool to determine the relative positions of certain letter parts. They are given by [8]:

\[ \bar{x} = \frac{\sum_{x} \sum_{y} x \times I(x, y)}{A} \]  

\[ \bar{y} = \frac{\sum_{x} \sum_{y} y \times I(x, y)}{A} \]  

III.3 Line Approximation

A. Line Definition:

In the context of the proposed approach, a line is defined as an object with no bulks and two end points (note that end points are simply determined from the pixel connectivity map with value = 1 i.e. only one connection). To approximate a line object with a real line we need two things: a slope (or an angle of inclination, i.e. we need to use 2nd moments) and a center point (or a point that it passes through, i.e. we need to use 1st moments \((\bar{x}, \bar{y})\)). We also need a measure of error to know how close to a line the object really is (i.e. we need to use the sum square error) [9]:

\[ Error = \sqrt{\sum_{x} \sum_{y} dist[\text{line}(x,y)]} \]  

Where “line” is the approximated line given by slope \( \tan(\theta) \) (where \( \theta \) is defined by Eq. (1)), and center \((\bar{x}, \bar{y})\). In addition, “dist” is the function that calculates the distance between line and point.

B. Lines characteristics:

The proposed method characterizes lines by their slope, center, length, end points, error from perfect line, and their connections to other lines, bulks, or loops. Lines can be segmented by using the error function, i.e., one can start at an end point and consider that object as a line and keep adding points to that line and updating the error until a certain error margin is exceeded. At that point, a new line is started.

Fig. 7 shows a segmentation of the letter “א” or “ha” (in the form it is written with in the middle of a word). The four segments obtained are given a different color each (shades of grey), and the center of each segment is marked on the Figure.
III.4 Proposed Detection Scheme

This Section summarizes the proposed handwriting recognition approach by linking together the discussions of the previous Sections. The main steps in the algorithm are executed in the following order:
1. Determine loops and their boundaries.
2. Determine bulks and their connections.
3. Determine which bulks are on the loop(s) and which are not.
4. Approximate the rest of the image with lines.
5. Check for the correct number, position and connections of bulks.
6. Check for correct orientations and characteristics of loops and lines.
7. Check for the correct connections (and their positions) between lines and between lines and loops (through bulks).
8. Check for correct relative dimensions/area of different components of the letter.
9. Do some extra checks that could be letter specific. (Open ended). This is the most challenging part since we need to study the characteristics of each letter.
10. At each of the previous steps take a note of every error that a user might do.

III.5 Employing Neural Networks to Check for Specific Shapes

This Section describes an additional step that is not critical to the functionality of the proposed approach and is not used in its current version, but could be used for future enhancements. It involves the use of Neural Networks to check for specific shapes. This is done because certain letters contain curved parts that need to be checked as is the case for "٤". The neural networks procedure involves the following steps:
1. Isolate the shape to be tested (for example the loop in the "٤"). This shape must not contain any bulk.
2. Find a convenient starting point e.g. the lower left corner.
3. Start moving pixel by pixel and recording the movement in a x and y movement vector, i.e., if to arrive to next pixel we moved 1 step in the x direction and one step in the y direction we record a 1 in the x vector and a 1 in the y vector. An example is illustrated in Fig. 8.
4. Elongate the x and y vectors to a convenient size keeping in mind that we need to preserve all characteristics, while at the same time providing to the neural network an input vector of constant size.
5. Pass an averaging filter to both x and y vectors to smooth them and make detection more immune to small shifts. More details about averaging filters are given in [9], and about their Matlab implementation in [10].
6. Integrate both x and y since the recorded data are steps not actual values i.e. like the derivative of the actual vector, so we integrate to compensate for that.
7. (Optional) normalize values to [0,1] range by dividing by the largest value in both x and y vectors.
8. Concatenate the y vector to the x vector to obtain a single vector, ready for the input of the neural network.

Hence, the input to the neural network consists of a single vector obtained by the concatenation of the x and y vectors after processing them according to the above steps. The neural network is trained to classify between several types of shapes including loops and lines. The neural network is also trained on a null input which basically features other none classes of wrong shapes (preferably common ones or common mistakes).

III.6 Alternate Way to Checking Curvatures

As discussed in Section III.5, neural networks are an effective way to tackle the problem of curves in letters. But our procedure discussed so far contains almost no ambiguity or statistical method. The method consists of parsing the letter into its fundamental parts using the notion of bulks, loops and lines, and then doing some deterministic checks. So the use of neural networks -though effective- breaks one of the fundamental advantages of our procedure, namely the clarity part. So to avoid using statistical tools we sought to make use of the tools we developed so far to come up with a procedure to solve the
curvature problem. The steps involved for this procedure are as follows:

1. Isolate the curved part under study (like the “ط”的 loop or the “ن”的 cuvre).
2. Use the line approximation procedure discussed earlier to approximate the curve with lines.
3. Find inclination of first and last line.
4. Find the signed angles between consecutive lines.
5. Check the angles obtained against the expected shape’s angles (for example for the “ن” the first line must be near vertical and the consecutive lines must be at a positive angle—counter clock wise—relative to each other reaching the last line at a near vertical angle).
6. Accommodate for certain anomalies in the shapes, like allowing limited violations of the rules as long as these errors are small relatively (for example, in Fig. 9 it can be seen that there is a part where two consecutive lines have negative angle between them, but in general the shape is correct).
7. Do some extra checks (open ended) that may be specific for each letter (as an example the two endpoints of the “ن” corresponding to the endpoints of the first and last line must almost line up vertically).

IV. RESULTS AND DISCUSSION

IV. 1 Results

To test the validity and accuracy of the proposed approach, some results are displayed in this section. The results test the ability of the method in detecting the different forms of errors in the write-up of a given letter and providing the correct feedback, the differences in performance while detecting a letter in its different positions, the efficiency in detecting loops and lines, and the role of the neural network in enhancing the performance. It should be noted that a user friendly graphical user interface was not created yet, since the focus so far has been on implementing the core part of the proposed approach.

Since the proposed method is designed specifically for educational purposes, we consider that the software is aware of the correct response that the illiterate user is supposed to provide; i.e. if a user is asked by the teaching software to write the letter "ط" for example, the software will compare the characteristics of the user input to those of the letter "ط", and then will decide if correct or wrong input was provided. In case of wrong input, the software will indicate to the user exactly what part of the letter was written erroneously. This approach however is easily extendable to the case of handwriting recognition in general (not only for educational purposes) where the user input is associated to the letter in the Arabic alphabet that best matches that input (i.e. the software is not aware of the correct answer). The software in this case has to compare the characteristics of the user input to all the letters of the Arabic alphabet and decide on the one that yields no errors.

Fig. 11 shows an example of a correctly written "ط". The software successfully detects that no errors were made since it returns 0 in the error counting variable “errCount” (see bottom of the Figure). Figs. 12 to 14 show examples of wrongly written letter "ط". In each case, the error counting variable “errCount” is set to 1 indicating that one error was detected. A description of the detected error is displayed as shown in the bottom of the Figures. With the other letters, the method behaves similarly to the case of Figs. 12 to 14, which were selected to present an indication of performance.

To test the accuracy of the method in detecting a letter in its four different forms, we consider the letter “ح” (“ha”) and write each of its forms correctly 60 times and check the number of erroneous and correct detections by the proposed approach. The results are displayed in Table I.

<table>
<thead>
<tr>
<th>TABLE I</th>
<th>CORRECT DETECTIONS OF THE LETTER “ح” (“HA”) IN ITS FOUR POSITIONS OUT OF 60 ATTEMPTS.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Beginning</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>60</td>
</tr>
</tbody>
</table>
Fig. 11 Correct detection.

Fig. 12. Wrong detection with error message saying that the Asayeh is wrong.

Fig. 13. Wrong detection with error message saying that there is no Asayeh.

Fig. 14. Wrong detection with error message saying that Asayeh is too long.
The letter “ha”, when written as the final letter at the end of a word, can be decomposed into two lines and a loop. When the 2nd line (vertical) is too short (below a certain length), the method fails to detect it and indicates an error, although the letter can be considered correct. This scenario happened only 5% of the time (95% correct detection), and similar results were obtained with the other letters. This result is not dramatic, and by increasing the letter size we can overcome this error since by doing so this line will become longer and hence detectable. In addition, an attractive feature of the proposed method is that it easily lends itself to updates and corrections. In fact, line detection is simple and no mistakes in detecting a line were reported, except in the scenario discussed above. Although loop detection is not as straightforward, no mistakes were recorded also since the detection method is deterministic and if there is a loop it will be detected unless it is not closed and the algorithm fails to close it. In other words, if one wrote a letter in a way that could be considered correct, but missed closing the loop by a certain number of pixels (this number is updatable), the software would indicate that a loop is absent. Although this does not represent a loop detection error, it could represent a false indication to the user (although indicating to the user that no loop is present in such a case could motivate him to enhance his handwriting and adjust the loop the next time he writes the letter). This problem can be solved, for example, by the neural network approach (Section III.5), since in this case the network will be trained to consider such a shape as a correct loop.

IV. 2 Advantages and Drawbacks of the Proposed Approach

The results show that the proposed approach is very efficient and has numerous advantages. The method is deterministic and deals with the characteristics of each letter. Hence, no statistical measures or training sets are involved (except for the ANN part which is not the core of this method but only provides an additional check). In addition, this approach provides valuable feedback of what is wrong in the written letter, which is especially useful in educational applications, namely fighting illiteracy in the Arab world. Furthermore, the method is immune to scaling and shifting since it deals with relative sizes. Finally, it can easily be extended to online handwriting recognition, since there are functions in the software developed so far that detect the (x,y) coordinates of the traced pixels in the order they were traced.

However, the originality of the proposed approach mandates that each letter should be carefully studied to find its key characteristics and the ways that it may be misspelled (e.g. Asayeh for “آ” is too short), in order to provide adequate feedback to the user. Furthermore, the relative dimensions of the letters’ characteristics must be investigated, e.g., knowing the dimensions of the loop of the “آ” we should determine the acceptable relative scale of its Asayeh. This relates to the reasonable margin of error that can be allowed while considering that the letter was correctly written.

V. CONCLUSIONS AND FUTURE WORK

A handwriting recognition method tailored for the Arabic alphabet was presented. The method is very suitable for teaching illiterates since it provides feedback to the user as to exactly what part of a letter was erroneously written (which pure neural network methods fail to do). This method promises extremely accurate and reliable results, and it can be easily extended to online handwriting recognition, an extension that is currently under investigation.

It should be noted that all the functionalities mentioned in this paper (from deriving connectivity maps to finding bulks and loops and their characteristics and to segmenting an image into lines using an error measure) were developed using Matlab toolboxes. We are currently using these functionalities to implement detection schemes for more Arabic letters, and we are also working on a user interface that would ultimately be embedded into a larger program that would help illiterates all over the Arab countries.

Acknowledgment

The authors greatly appreciate the support and cooperation of Ms. Amal Charara of the Lebanese National Committee for Literacy at the Lebanese Ministry of Social Affairs, and Ms. Nour Dajani of the UNESCO. Gratitude also goes to George Kadifa, President and CEO of Coreo Inc., for their support and encouragement.

The authors are grateful for the generous support provided by grants from the Rathmann Family Foundation, UNESCO, and URB (University Research Board) of the American University of Beirut.

REFERENCES


