Efficient Segmentation of Arabic Handwritten Characters Using Structural Features

Mazen Bahashwan, Syed Abu-Bakar, and Usman Sheikh
Department of Electronics and Computer Engineering, Universiti Teknologi Malaysia, Malaysia

Abstract: *Handwriting recognition is an important field as it has many practical applications such as for bank cheque processing, post office address processing and zip code recognition. Most applications are developed exclusively for Latin characters. However, despite tremendous effort by researchers in the past three decades, Arabic handwriting recognition accuracy remains low because of low efficiency in determining the correct segmentation points. This paper presents an approach for character segmentation of unconstrained handwritten Arabic words. First, we seek all possible character segmentation points based on structural features. Next, we develop a novel technique to create several paths for each possible segmentation point. These paths are used in differentiating between different types of segmentation points. Finally, we use heuristic rules and neural networks, utilizing the information related to segmentation points, to select the correct segmentation points. For comparison, we applied our method on IESK-arDB and IFN/ENIT databases, in which we achieved a success rate of 91.6% and 90.5% respectively.*

Keywords: *Arabic handwriting, character segmentation and structural features.*

Received December 23, 2014; accepted August 26, 2015

1. Introduction

Automatic handwriting recognition has many applications, such as for bank cheque processing, address and zip code recognition on envelopes, and handwriting analysis [9, 13, 25]. As a result, studies had been conducted on handwriting recognition for many languages such as English, Chinese, Japanese, and Arabic, among others [4, 21]. This paper focuses on Arabic text recognition. The Arabic language is an official language in over 25 countries and is spoken by approximately 234 million people [22]. Arabic characters are similar to characters in other languages such as Jawi, Farsi, Urdu, and Kardi [8, 24]. Approximately 7 to 10 million manuscripts were written in Arabic script between the seventh and fourteenth centuries [17]. Hence, a high-performance offline Arabic script recognition is needed for certain tasks, such as preservation of old manuscripts.

A large gap exists between the research on Latin script and the research on Arabic script. Among the reasons for this gap are lack of adequate support in terms of financial funding, coordination, and other utilities, such as comprehensive Arabic text databases and dictionaries [14]. This situation could also be attributed to difficulties associated with characteristics of Arabic script, which will be described in Section 3.

The recognition of unconstrained cursive Arabic handwriting is still low because of poor character segmentation. This finding is an indication that segmentation plays a vital role in the character recognition process [2, 18].

Segmentation approaches can be divided into two categories: holistic approach and analytical approach [27]. Systems that are based on the holistic approach (also called as global approach) try to recognize entire words without splitting them into individual characters. The disadvantage of the holistic approach is that it needs a large dictionary (lexicon), which, in turn, makes the searching process costly [1]. On the other hand, systems based on the analytical approach try to segment a word into characters or graphemes (part of the character) and does not require the use of a large dictionary. In this work, the analytical approach is adopted. The technique consists of three stages: preprocessing, generating candidate segmentation points, refining and verifying all candidate points.

This paper is organized into six sections. Section 2 explains some works related to Arabic character segmentation. Section 3 illustrates the characteristics of Arabic language scripts. Section 4 describes our proposed method to generate the segmentation points, segmentation paths and the process of refining and verifying the segmentation points. Section 5 presents the results and discussion. Finally, we conclude our paper in Section 6 and suggest several directions for future works.

2. Related Works

For the last few decades, an increasing number of empirical studies have been conducted on handwriting recognition. However, findings on recognizing unconstrained cursive handwriting remain limited. One of the major reasons for this lack of findings can be attributed to poor character segmentation [18]. The suitable features for character segmentation can be in the form of segmentation points such as local minima [5], branch points, cross points, loop points [11], or pen thickness [20]. These features may not be detected in all segmentation points because some of them might be lost during the writing or the preprocessing stage.

Bouafif *et al.* [7] used Harris corner method to detect possible segmentation points in words, and only corners that lie between 5 pixels above and below the baseline were taken as valid segmentation points. Obviously, many valid segmentation points that were outside this range were ignored. The main disadvantage of this technique is its high dependency on the baseline.

Elnagar and Bentrcia [10] used six agents and a baseline for detecting segmentation points. They mentioned some limitations of their algorithm, such as missing segmentation points due to weakness in agents and dependency on the endpoints, branch points, and cross points as features. Moreover, the detection of the agents was error prone because of its high dependency on the accuracy of the baseline detection.

Al-Hamad introduced an algorithm for segmentation and validation of Arabic handwritten words [3]. His method involved three major steps. First, segmentation points are obtained from a modified vertical histogram of a thinned word–image. Then, the initial segmentation points are validated by using a neural-based segmentation point validation scheme. Finally, the fusion confidence value is obtained to validate segmentation points. According to Al-Hamad [2], the modified vertical histogram has a limited ability to identify some characters, such as the character baa (ب) and similar character shapes. Moreover, such characters are not detected as characters because they look like ligatures in the histogram. The main limitation is the presence of numerous incorrect local minima and maxima, which often result in a large number of incorrect ligatures.

Elzobi *et al.* [11] used a histogram to detect possible segmentation points and applied heuristic rules to refine the result. They reported the occurrence of missed segmentation points because of the overlapping characters and over-segmentation points in characters such as seen (س) and sheen (ش). In addition, the rules cause missed and over-segmentation points because of the dependency on cross points, branches, and loop points.

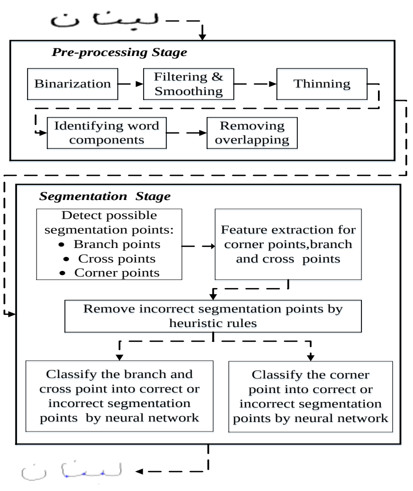
Thus, we propose a new segmentation method based on using a corner detector, branch points, and cross points. We also propose a novel verification technique of the segmentation points using heuristic rules and neural network.

3. Characteristics of Arabic script

The Arabic script consists of 28 characters and is written from right to left. Each character has at least two to four shapes that depend on the position of the character within the word. In addition, a single word may consist of one or more than one sub words. This is because the following characters cannot be joined (ا,د,ذ,ر,ز,و) from the left side. More than half of the characters contain one to three dots. These dots may be at the top, middle, or bottom of the character. These dots distinguish between characters that may otherwise have the same shape. Usually, the characters are connected horizontally, but in some cases, characters may be connected vertically. Overlapping normally occurs between the sub-words or between the sub-word and characters in the same word, which are mostly found in the vertical direction.

4. Proposed Method

The general block diagram of the proposed method is shown in Figure 1. Initially, an image will be loaded into the system followed by some operations at the preprocessing stage before the segmentation stage is initiated.



Pre-processing Stage
Binarization
Filtering & Smoothing
Thinning
Identifying word components
Removing overlapping
Segmentation Stage
Detect possible segmentation points:
• Branch points
• Cross points
• Corner points
Feature extraction for corner points,branch and cross points
Remove incorrect segmentation points by heuristic rules
Classify the branch and cross point into correct or incorrect segmentation points by neural network
Classify the corner point into correct or incorrect segmentation points by neural network

Figure 1. Block diagram of character segmentation.

4.1. Preprocessing

The operations applied in the preprocessing stage are for general images. However, all the images in the database we used, i.e., the IESK-arDB and IFN/ENIT datasets, underwent binarization, smoothing, dilation operation.

Generally, an Arabic word contains one or more main components. The main component contains a single character or some connected characters. More than half of the characters in the Arabic script have a secondary component such as a dot and hamza, and the size of the secondary component is usually very small compared with the main component and is quite far from the center of the main component, as shown in Figure 2. In this figure, the main component is represented by a rectangular box, while the secondary component is represented by an ellipse.

Baseline detection is a method that is used to identify word components. The baseline corresponds to a simple horizontal projection that contains the maximum number of foreground pixel count. A box is first drawn in each connected component, and then the main connected component is intersected with the baseline (as depicted in Figure 2).

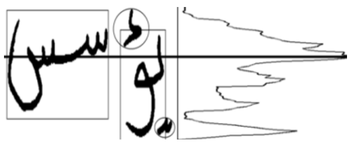


Figure 2. Arabic word that contains two main components and two secondary components with a baseline drawn.

Each secondary component is assigned to one main component. If the secondary components are inside, above, or below the main component without overlapping, then they will be assigned to that main component. In case of overlapping between two main components, the distance will be measured depending on the location of the secondary components. According to this distance, the secondary components are assigned to the nearest main components [11].

Finally, we apply thinning operation based on the approach used by Zhang and Suen [28]. This method has been widely used in previous researches [15, 16, 19]. Figure 3 shows an example of the thinning operation.

a) original binary image. b) result after the thinning process.

Figure 3. An example of Zhang and Suen thinning algorithm.

4.2. Candidate Segmentation Points

Usually, the connection point between two characters is considered a branch point or a cross point. However, the connection point (branch point or cross point) might not be present due to the writing style or as a result of the preprocessing operation. The missed connection point can still be captured using a corner detector as shown in Figure 4. Therefore, to avoid missing the connection point, our approach uses the corner points as the candidate segmentation points.

a) due to preprocessing stage. b) due to writing style; these missing points are indicated with small squares.

Figure 4. Missing segmentation points.

In our work, we adopted the contour-based detector method with the Chord-to-Point Distance Accumulation (CPDA), which is found in [6] to detect the corners as candidate segmentation points. CPDA discrete curvature estimation is less sensitive to local variations and noise on the curve and it does not use any derivatives. We generate short curves by removing branch points and the loop points from the thinned word. We then apply the corner detector to each curve separately.

To detect the corner in the curve using the CPDA [6], the curve is first smoothened, and then three chord-lengths ($L_1=10$, $L_2=20$, $L_3=30$) are used to estimate the curvature value at each point n in the curve. The CPDA curvatures ($h_1(k)$, $h_2(k)$,$h_3(k)$) that correspond to the three chord lengths are then computed by using Equation (1). The CPDA discrete curvature $h_L(k)$ at the point K with the chord length L is calculated by taking the summation of all perpendicular distances from the point K to the chord length at all possible chord length locations.

$$h_L(k)=\sum_{j=k-L+1}^{k-1}\left(d_{k,j}\right) \qquad (1)$$

Then, the CPDA discrete curvature for each chord length is normalized by using Equation (2)

$$h'_j(k)=\frac{h_j(k)}{\max(h_j)},\ for\ 1<k<n\ \&1\leq j\leq 3 \qquad (2)$$

All these curvatures are then multiplied to produce a single value, which is called the curvature product, $H(k)$, as given in Equation (3).

$$H(k)=h'_j(k).h'_j(k).h'_j(k),\ \ for\ \ 1\leq k\leq n \qquad (3)$$

The curvature product curve is then smoothened, and the candidate corners are located by detecting the maxima $H(k)$. Finally, our proposed method adopts the algorithm in [12] to measure the angle at each candidate corner point by using two tangent lines.

4.3. Construction of the Segmentation Path

In this study, we propose a segmentation path based method to extract features which are related to segmentation points. The proposed segmentation path is a curve that starts from a point on the thinned word to one of the three ends: either to the top boundary of the image, bottom boundary of the image, or back to the start point for a closed-loop path. Before creating a path, a preprocessing step is applied on the background of the image. This preprocessing step will change the value of the background pixels based on some conditions. We use four different values of base 2 (2^n), where n is an integer, to assign the background values. These values have the following characteristics: 1) summation of any two values from a set of four values will always be different from the sum of the remaining values; 2) summation of any three out of a set of four values will always be different from the last remaining value. These characteristics will ensure the creation of a unique track path.

These four numbers are arranged in decreasing order from the highest to the lowest with the following labels: *Pt*, *Pb*, *Pr*, and *Pl*. The descending arrangement from highest to lowest value is used to determine the track path of either as top, bottom, right, or left, respectively.

We then scan the image column by column from top to bottom. During the scanning process, all background pixels (255)[1] are replaced by *Pt* until foreground pixel (0) is reached or until the bottom of the image is reached. This condition can be formulated as follows;

Let A_{MxN} be an image, and $a_{ij} \in A$ denotes the pixel value of row i and column j. The image is first scanned from top to bottom in each column separately.

$$a_{ij} = \begin{cases} Pt & \text{if } \left(a_{ij} = 255 \text{ and } a_{i-1,j} = Pt\right) \text{ or } a_{1,j} = 255 \\ 0 & \text{if } a_{ij} = 0 \\ 255 & \text{if } a_{ij} = 255 \text{ and } \left(a_{i-1,j} = 0 \text{ or } a_{i-1,j} = 255\right) \end{cases} \tag{4}$$

Then, the output image from the previous step is scanned from the bottom to the top, and all pixels with the value of 255 are replaced with *Pb*. The scan continues until the foreground pixel (0) or *Pt* in the image is reached. This can be described as follows:

$$a_{ij} = \begin{cases} Pb & \text{if } \left(a_{ij} = 255 \ \& \ a_{i+1,j} = Pb\right) \text{ or } a_{i=m,j} = 255 \\ 0 & \text{if } a_{ij} = 0 \\ Pt & \text{if } a_{ij} = Pt \\ 255 & \text{if } a_{ij} = 255 \ \& \ \left(a_{i+1,j} = 0 \text{ or } a_{i+1,j} = 255\right) \end{cases} \tag{5}$$

The procedure is repeated from right to left, with pixels with a value of 255 being replaced with *Pr*. This condition is described as follows:

$$a_{ij} = \begin{cases} Pb & \text{if } a_{ij} = Pb \\ 0 & \text{if } a_{ij} = 0 \\ Pt & \text{if } a_{ij} = Pt \\ Pr & \text{if } a_{ij} = 255 \ \& \ \left(a_{i,j+1} = Pt \text{ or } a_{i,j+1} = Pb \text{ or } a_{i,j+1} = Pr\right) \\ 255 & \text{if } a_{ij} = 255 \ \& \ \left(a_{i,j+1} = 0 \text{ or } a_{i,j+1} = 255\right) \end{cases} \tag{6}$$

Finally, the image is scanned from left to right, with pixels with a value of 255 being replaced with *Pl* until the foreground pixel (0) is reached. This step is expressed as follows:

$$a_{ij} = \begin{cases} Pb & \text{if } a_{ij} = Pb \\ 0 & \text{if } a_{ij} = 0 \\ Pt & \text{if } a_{ij} = Pt \\ Pr & \text{if } a_{ij} = Pr \\ Pl & \text{if } a_{ij} = 255 \ \& \ \left(a_{i,j-1} = Pt \text{ or } a_{i,j-1} = Pb \text{ or } a_{i,j-1} = Pl\right) \\ 255 & \text{if } a_{ij} = 255 \ \& \ \left(a_{i,j-1} = 0 \text{ or } a_{i,j-1} = 255\right) \end{cases} \tag{7}$$

For easy explanation, the converted background pixels are referred to as the "guiding pixels".
Once the scanning procedures are done, the background pixels will have either one of these guiding pixel values (255, *Pt*, *Pb*, *Pr*, *Pl*), and they are grouped into five classes that define their respective direction path, as shown in Table 1.

Table 1. Path direction based on guiding pixel value

Background pixel value	**Path direction**
Pt	Top-bound
Pb	Bottom-bound
Pr	Right-bound
Pl	Left-bound
255	Move with reference point

To find the starting point for each path, a 3x3 window is used with the candidate segmentation point placed at its center. An anticlockwise scanning mechanism is performed starting from the upper right corner of the window. The scanning will continue until all the guiding pixels that surround the foreground pixels are grouped together. Depending on the type of the segmentation point (i.e., corner point, branch point, or cross point), two, three, or four groups can exist, as shown in Figure 5.

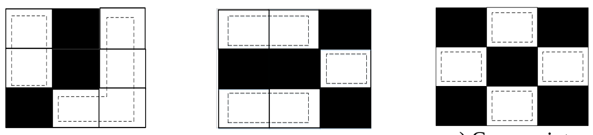
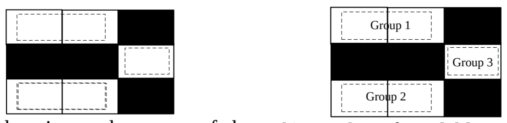
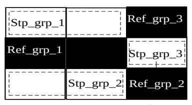

a) Corner point. b) Branch point. c) Cross point.

Figure 5. Dashed boxes represent groups in (a) corner point-2 groups, (b) branch point-3 groups and (c) cross point-4 groups.

The starting point of the path for each group (Stp_grp) will be the last guiding pixel before the foreground pixel (reference point), with respect to the anticlockwise scanning mechanism. Initially, the reference point (Ref_grp) refers to the first foreground pixel after the starting point in the anticlockwise direction. An example for the branch point case is shown in Figure 6.



a) Branch point–at the center of the window. b) Number of available groups.



c) Location of the respective start and reference points for tracing each path. Stp_grp is the start point and Ref_grp is the reference point in the group respectively.

Figure 6. An example of a branch point with its related reference points and starting points.

We begin tracing in each path from the start point. The next move is determined by checking the value of the current pixel. If it is equal to one of these guiding pixel values (*Pt*, *Pb*, *Pr*, *Pl*), then the path moves according to the direction specified by these values (refer to Table 1).Otherwise, if it is equal to 255, then the path will move according to the reference point, and the current reference point will be changed. The

[1] In this implementation, we use the value of 255 instead of 1 for the binary image.

reference point has four possible positions with respect to the current path point, as shown in Figure 7. To determine the next step for the path, the location of the next reference point needs to be determined. First, the starting scanning point (marked as x in Figure 7) needs to be located. Equation 8 is used to determine the initial scanning.

$$\begin{aligned} x_{sc} &= x_{cp} - \left(y_{cp} - y_{ref}\right) \\ y_{sc} &= y_{cp} + \left(x_{cp} - x_{ref}\right) \end{aligned} \quad (8)$$

where x_{sc} and y_{sc} are the coordinates for the initial scanning point, x_{cp} and y_{cp} are the coordinates of the current path point, and x_{ref} and y_{ref} are the coordinates of the current reference point. Here, we follow the convention of the right-hand rule for the coordinates system. Anticlockwise scanning will then begin from this initial scanning point. During the scanning, several cases may occur as follows:

- *Case 1*. If the next pixel encountered is one of the 4 guiding pixels, then the path will proceed as indicated in Table 1, where it will eventually reach either the top or the bottom of the image. In this case, the path will be completed.
- *Case 2*. If the next pixel encountered is 255, then the scan will proceed until it reaches the next black pixel. In this case, this black pixel will become the next reference point, and the pixel just before this black pixel (in the scanning direction) becomes the new current path point.

This process of searching for the current path point and the current reference point will continue until either case 1 above is met or the path returns to the initial scanning path (i.e., making a closed loop).

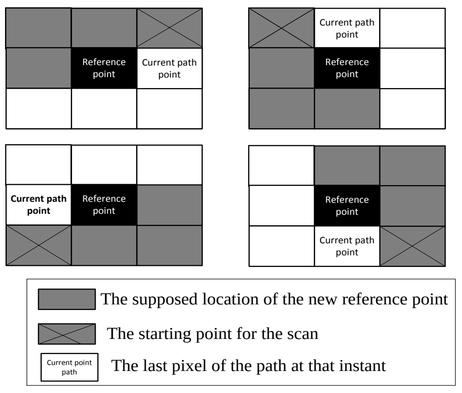

Reference point
Current path point
The supposed location of the new reference point
The starting point for the scan
Current point path
The last pixel of the path at that instant

Figure 7. An example of four possible locations of the reference point with the current path point.

4.4. Feature Extraction

The segmentation paths, corner detection (CPDA), binarized and thinned images produce useful information which can be used as features for the refinement and verification of candidate segmentation points.

The features extracted from the corner points are listed below:

- The curvature product and angle of each corner is used as features.
- The thickness of the corresponding coordinate of the corner point in the image before the thinning process is also used as feature.
- For a given corner point located in between two connected points (which may be a branch, corner, cross or end point), the type of the two connected points and their respective distance to the corner point are used as features.
- The vertical distance from the baseline to a candidate segmentation point is used as feature.
- The number of the secondary components in both sides of the connected candidate segmentation point is used as features.

The features extracted from the branch and cross points are listed below:

- The number of top-bound, bottom-bound and loop paths at each candidate segmentation point are used as features.
- The measured width between the two segmentation paths moving in the same direction is used as feature.
- The number of the foreground pixels located between the two segmentation paths moving in the same direction is used as feature.
- For a given candidate segmentation point located in between two connected points (which may be a branch, corner, cross or end point), the type of the two connected points and their respective distance to a given candidate segmentation point are used as features.
- The vertical distance from the baseline to a candidate segmentation point is used as feature.
- The number of the secondary components in both sides of the connected candidate segmentation point is used as features.

4.5. Refinement and Verification

We proposed to refine and verify the candidate segmentation points by two steps; first refining candidate segmentation points by using some heuristic rules, secondly, we use neural networks to verify the rest of candidate segmentation points, as explained in the following subsections:

4.5.1. Segmentation Point Refinement by Heuristic Rules

In order to reduce the consumption time for the training process, heuristic rules are used to remove candidate segmentation points which in practice cannot be correct segmentation points. We apply five heuristic rules based on empirical studies that cover the most probable segmentation point scenarios. These rules are given as follows:

- If a branch point has three paths and two of them are closed-loop paths, then this branch point is removed from the list of segmentation point candidates.
- If a segmentation point is located five pixels from either end of the curves, then this segmentation point is removed from the list of segmentation point candidates.
- If a branch point has two bottom-bound paths and one top-bound path, located next to a branch point with a closed-loop path, and the distance between them is less than five pixels and the x-coordinate of the end point is equal to or less than the x-coordinate of the branch point, then this branch point is removed from the list of segmentation point candidates.
- If two adjacent branch points have two top-bound paths with no secondary component centroid coordinate between them, and the number of pixels between the two top-bound path points is less than 25 pixels, then the two branch points are removed from the list of segmentation point candidates.
- If a corner point has curvature product value less than 0.09 and angle value larger than 150^o, then this corner point is removed from the list of segmentation point candidates.

4.5.2. Segmentation Point Verification using Neural Networks

The nature of the corner point structure is different from the branch and cross point due to the fact that features extracted are not similar. Therefore, the proposed method uses two parallel back-propagation neural networks with log-sigmoid activation function to verify the correct and incorrect segmentation points; one neural network is used to verify the corner point and another to verify the branch and cross points. The neural networks' input layer consist of features extracted from the candidate segmentation points, while the output layer represents the classified segmentation point as correct or incorrect segmentation point.

5. Results and Discussion

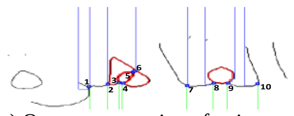
We tested our proposed segmentation approach on 1,200 word images obtained from the IESK-arDB and IFN/ENIT databases [11, 23], whereby the words were handwritten by different people. For the purpose of comparison, three criteria, i.e., correct segmentation, over-segmentation, and under-segmentation were evaluated to measure the performance of the segmentation technique.

Correct segmentation refers to points that divide the two characters correctly. Over-segmentation refers to unnecessary or excess points in segmenting two characters, while under-segmentation refers to a situation in which a missed correct segmentation point exists between two characters. However, a notable detail is that no unique position for correct segmentation exists in Arabic characters. Therefore, the results were validated by visual observation. As seen from Table 2, our method significantly reduced the under- and over-segmentation points, and the correct segmentation accuracy has been improved compared to other methods.

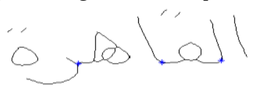
Table 2. Criteria for evaluating the segmentation rate (%).

Authors	**Over-segmentation %**	**Under-segmentation %**	**Correct segmentation %**	**No. of words**	**Database**
Elzobi *et al.* [11]	14.4	18.6	67	600	IESK-arDB
Xiu *et al.* [26]	18.8	26.6	54.6	600	IESK-arDB
Al-Hamad and Abu-Zitar [3]	17.02%	4.60%	82.98	500	local database
Elnagar and Bentrcia [10]	≈13.7%	0.3%	86%	550	IFN/ENIT
Our method	8%	0.4%	91.6%	600	IESK-arDB database
Our method	8.9%	0.6%	90.5%	600	IFN/ENIT database

The construction of the segmentation path depends on the values of the guiding pixels *Pt*, *Pb*, *Pr*, and *Pl*. In our work, we use *Pt* =16, *Pb* =8, *Pr*=4, and *Pl*=2. Figure 8-a shows the result after constructing the segmentation paths. As seen from the figure, some segmentation points have paths that go to the top boundary, bottom boundary, and a closed-loop path. For example, the difference between the two segmentation points (labels 8 and 9) is only in the location of the closed loop path, i.e., either the closed loop is located to the left or the right side of the segmentation point. Some segmentation points (labels 3, 4, and 6) have only one path and two closed-loop paths, and the difference in this case is the direction of the path (either to the top or to the bottom). Furthermore, a segmentation point (label 5) that has three closed-loop paths exists. In addition, some segmentation points (labels 7 and 10) have two top-bound paths and one bottom-bound path. Finally, a segmentation point (label 1) has two top-bound paths and one bottom-bound path. The result of the proposed method is shown in Figure 8-b.



a) Over-segmentation of points.



b) Segmentation points after applying the proposed method.

*	Segmentation points
🟦	Path leading to top boundary
🟩	Path leading to bottom boundary
🟥	Path going in a closed loop path
□	Candidate segmentation points

Figure 8. An example showing candidate segmentation points with its paths Over-segmentation of points and segmentation points after applying the proposed method.

In addition, our method can handle different cases, such as overlapping and cursively written words. Figure 9 illustrates some of the correct word segmentation, in which our proposed method can detect the segmentation points in cases of overlapping characters and cursively written words.

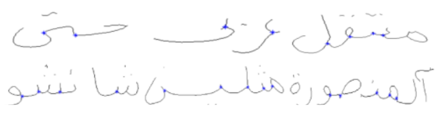


Figure 9. Segmentation of overlapping and cursively handwritten words.

Our method is also capable of correctly detecting the segmentation points for different writing styles from different writers. Figure 10 shows some of the results.

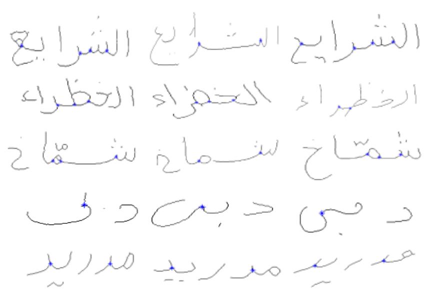
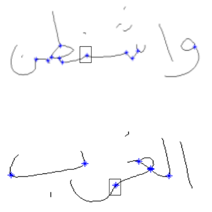
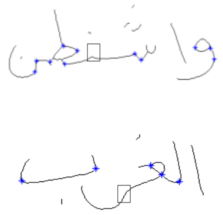


Figure 10. Segmentation of Arabic words written by five writers.

One of the strengths of our method is that it reduces under-segmentation points, because our method can detect valid segmentation points with a small curvature as illustrated in Figure 11. In this figure, the method [6] produced over-segmentation points in characters (ط ,و,ن) and missed one segmentation point between character (ش) and character (ن), because loops and the curvature product ($H(k)$) (0.07) are less than the threshold. Our method overcomes this limitation by removing loop points and modifying the curve extraction approach. Our approach manages to detect the segmentation point between character (ش) and character (ن) because the curvature product ($H(k)$) (0.98) is higher than the threshold value (0.09).



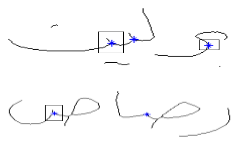
a) Corner detected by using our proposed method.



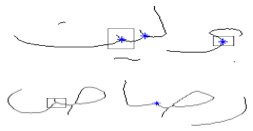
b) Corner detected by using Awrangjeb and Lu [6].

Figure 11. Results on corner point detection. The small boxes in (a) indicate that the proposed method can detect the proper corner point while the other method failed to detect them as shown in (b).

Due to a similarity to another connected character in terms of shape, over-segmentation points occur in characters like Sad (ص). The proposed method is able to remove the over-segmentation points by taking into account secondary components such as dots, as shown in Figure 12.



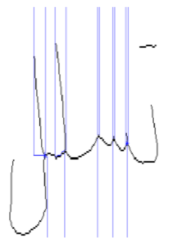
a) Detected segmentation points for the letter (ن) and (ص).



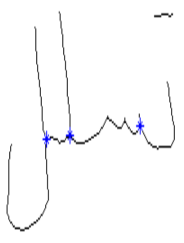
b) Result after applying the proposed method. Note that the segmentation point on the letter (ص) has been removed.

Figure 12. Results on proper detection of segmentation point.

The main difference between the character seen (ـس) and two or more connected characters lam (لل) is the length of the spur and stem, respectively. The character seen (س) consists of two or three small spurs, while the connected character lam has two or three stems, as shown in Figure 13-a. In our proposed method, we measure the length of the spur and stem before deciding the status of the segmentation points, as shown in Figure 13-b.



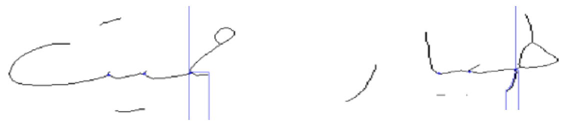
a) Candidate segmentation points with segmentation path in blue lines.



b) Result of segmentation points after applying the proposed method.

Figure 13. An example showing correct detection of segmentation points.

Our method can also distinguish between the line extension that belongs to the character and other characters such as meem (ـمـ). Figure 14-a shows the branch point with two bottom-bound paths and one top-bound path. The proposed method can detect the correct segmentation points, as shown in Figure 14-b, by determining the direction of the line and the number of pixels between them.


a) Results of detected candidate segmentation points with the segmentation path.

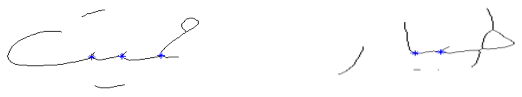

b) Results after applying the proposed method. Note that the proposed able to differentiate proper segmentation points between the letter (م) and (ط).

Figure 14. An example showing proper segmentation points for letter (م).

Nevertheless, the proposed method would miss some segmentation points in case the point does not have the features of branch, cross, and corner points, as shown in Figure 15. This issue can be solved by studying the angles of those points.

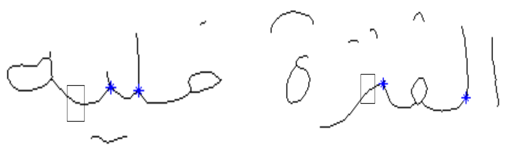


Figure 15. Arabic words with missing segmentation points (indicated by the box).

Likewise, our method still suffers from over-segmentation points because it generates many corners for cursive handwritten words, as shown in Figure 16. To solve this problem, we can either add more rules or extract new features to remove these unwanted points.

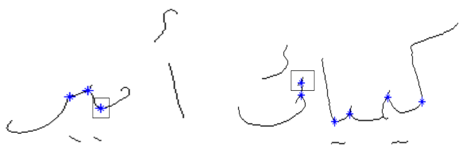


Figure 16. Arabic words with over-segmentation points due to corners.

6. Conclusions and Future Works

This paper presents an offline Arabic handwriting segmentation method based on structural techniques, in which the segmentation points are categorized into branch points, cross points, and corner points. This paper introduces a method for generating all possible segmentation points and a way of refining them. By detecting the branch points and cross points as segmentation points, the method divides the main components into small curves to detect small curvature corners as segmentation points. Finally, heuristic rules and neural networks are applied to select the correct segmentation points. The average accuracy of the proposed method is 91.05%. For future work, we suggest addressing issues such as establishing the relationship between the secondary components and segmentation points and using that in the refinement and verification step to further improve the segmentation point classification.

Acknowledgement

The authors would like to thank Universiti Teknologi Malaysia (UTM) for the support in the research and development of this work and the Ministry of Science, Technology and Innovation, Malaysia (MOSTI), (Science Fund Grant No. 01-01-06-SF1197, R.J130000.7923.4S081).

References

[1] Abandah G. and Jamour F., "Recognizing Handwritten Arabic Script through Efficient Skeleton-Based Grapheme Segmentation Algorithm," *in Proceeding of International Conference on Intelligent Systems Design and Applications*, Cairo, pp. 977-982, 2010.

[2] Al-Hamad H., "Over-Segmentation of Handwriting Arabic Scripts Using an Efficient Heuristic Technique," *in Proceeding of Wavelet Analysis and Pattern Recognition*, Xian, pp. 180-185, 2012.

[3] Al-Hamad H. and Abu-Zitar R., "Development of an Efficient Neural-Based Segmentation Technique for Arabic Handwriting Recognition," *Pattern Recognition*, vol. 43, no. 8, pp. 2773-2798, 2010.

[4] Al-Jawfi R., "Handwriting Arabic Character Recognition LeNet Using Neural Network," *The International Arab Journal of Information Technology*, vol. 6, no. 3, pp. 304-309, 2009.

[5] Alaei A., Nagabhushan P., and Pal U., "A Baseline Dependent Approach for Persian Handwritten Character Segmentation," *in Proceeding of International Conference on Pattern Recognition*, Istanbul, pp. 1977-1980, 2010.

[6] Awrangjeb M. and Lu G., "Robust Image Corner Detection Based on the Chord-To-Point Distance Accumulation Technique," *IEEE Transactions on Multimedia*, vol. 10, no. 6, pp. 1059-1072, 2008.

[7] Bouafif F., Maddouri S., and Ellouze N., "A Hybrid Method for Three Segmentation Level of Handwritten Arabic Script," *The International Arab Journal of Information Technology*, vol. 9,

no. 2, pp. 117-123, 2012.

[8] Broumandnia A. and Shanbehzadeh J., "Fast Zernike Wavelet Moments for Farsi Character Recognition," *Image and Vision Computing*, vol. 25, no. 5, pp. 717-726, 2007.

[9] Broumandnia A., Shanbehzadeh J., and Rezakhah M., "Persian/Arabic Handwritten Word Recognition Using M-Band Packet Wavelet Transform," *Image and Vision Computing*, vol. 26, no. 6, pp. 829-842, 2008.

[10] Elnagar A. and Bentrcia R., "A Multi-Agent Approach to Arabic Handwritten Text Segmentation," *Journal of Intelligent Learning Systems and Applications*, vol. 4, no. 3, pp. 207-215, 2012.

[11] Elzobi M., Al-Hamadi A., Al-Aghbari Z., and Dings L., "IESK-ArDB: a Database for Handwritten Arabic and an Optimized Topological Segmentation Approach," *International Journal on Document Analysis and Recognition*, vol. 16, no. 3, pp. 1-14, 2012.

[12] He X. and Yung N., "Curvature Scale Space Corner Detector with Adaptive Threshold and Dynamic Region of support," *in Proceeding of the 17th International Conference on Pattern Recognition*, Cambridge, pp. 791-794, 2004.

[13] Jayadevan R., Kolhe S., Patil P., and Pal U., "Automatic Processing of Handwritten Bank Cheque Images: a Survey," *International Journal on Document Analysis and Recognition*, vol. 15, no. 4, pp. 267-296, 2012.

[14] Kabbani R., "Selecting Most Efficient Arabic OCR Features Extraction Methods Using Key Performance Indicators," *in Proceeding of International Conference on Communications, Computing and Control Applications*, Marseilles, pp. 1-6, 2012.

[15] Khorsheed M., "Recognising Handwritten Arabic Manuscripts Using a Single Hidden Markov Model," *Pattern Recognition Letters*, vol. 24, no. 14, pp. 2235-2242, 2003.

[16] Lee H. and Chen B., "Recognition of Handwritten Chinese Characters via Short Line Segments," *Pattern Recognition*, vol. 25, no. 5, pp. 543-552, 1992.

[17] Leydier Y., Ouji A., LeBourgeois F., and Emptoz H., "Towards an Omnilingual Word Retrieval System for Ancient Manuscripts," *Pattern Recognition*, vol. 42, no. 5, pp. 2089-2105, 2009.

[18] Liang Y., Fairhurst M., and Guest R., "A Synthesised Word Approach to Word Retrieval In Handwritten Documents," *Pattern Recognition*, vol. 45, no. 12, pp. 4225-4236, 2012.

[19] Lu S., Ren Y., and Suen C., "Hierarchical Attributed Graph Representation and Recognition of Handwritten Chinese Characters," *Pattern Recognition*, vol. 24, no. 7, pp. 617-632, 1991.

[20] Mansour M., Benkhadda M., and Benyettou A., "Optimized Segmentation Techniques for Handwritten Arabic Word and Numbers Character Recognition," *in Proceeding of IEEE Signal-Image Technology and Internet-Based Systems*, pp. 96-101, 2005.

[21] Naz S., Hayat K., Razzak M., Anwar M., Madani S., and Khan S., "The Optical Character Recognition of Urdu-Like Cursive Scripts," *Pattern Recognition*, vol. 47, no. 3, pp. 1229-1248, 2014.

[22] Parvez M. and Mahmoud S., "Arabic Handwriting Recognition using Structural and Syntactic Pattern Attributes," *Pattern Recognition*, vol. 46, no. 1, pp. 141-154, 2013.

[23] Pechwitz M., Maddouri S., Märgner V., Ellouze N., and Amiri H., "IFN/ENIT-Database of Handwritten Arabic Words," *in Proceeding of Francophone International Conference on writing and Document*, Hammamet, pp. 127-136, 2002.

[24] Razak Z., Zulkiflee K., Noor N., Salleh R., and Yaacob M., "Off-Line Handwritten Jawi Character Segmentation Using Histogram Normalization and Sliding Window Approach for Hardware Implementation," *Malaysian Journal of Computer Science*, vol. 22, no. 1, pp. 34-43, 2009.

[25] Touj S., Ben-Amara N., and Amiri H., "Arabic Handwritten Words Recognition Based on a Planar Hidden Markov Model," *The International Arab Journal of Information Technology*, vol. 2, no. 4, pp. 318-325, 2005.

[26] Xiu P., Peng L., Ding X., and Wang H., "Offline Handwritten Arabic Character Segmentation with Probabilistic Model," *in Proceeding of the 7th international conference on Document Analysis Systems*, Nelson, pp. 402-412, 2006.

[27] Zeki A., "The Segmentation Problem in Arabic Character Recognition the State of the Art," *in Proceeding of 1st International Conference on Information and Communication Technologies*, Karachi, pp. 11-26, 2005.

[28] Zhang T. and Suen C., "A Fast Parallel Algorithm for Thinning Digital Patterns," *Communications of the ACM*, vol. 27, no. 3, pp. 236-239, 1984.

Mazen Bahashwan is currently a postgraduate student at the Computer Vision, Video and Image Processing Lab (CvviP), Faculty of Electrical Engineering, Universiti Teknologi Malaysia. His research interest is in the area of computer vision, particularly in Arabic handwriting recognition. He obtained his master degree from Universiti Kebangsaan Malaysia in 2011.

Syed Abu-Bakar received his Ph.D. degree from the University of Bradford, England in 1997. He joined Universiti Teknologi Malaysia (UTM) in 1992. Currently he is an associate professor in the department of Electronics and Computer Engineering, Faculty of Electrical Engineering. His current research interest is in image processing focusing in video security and surveillance, medical imaging, biometrics, agricultural, and industrial applications. He has published more than 150 scientific papers both at national and international levels. He is a senior member of IEEE.

Usman Sheikh received his PhD degree (2009) in image processing and computer vision from Universiti Teknologi Malaysia. His research work is mainly on computer vision and embedded systems design. He is currently a Senior Lecturer at Universiti Teknologi Malaysia, Malaysia.