Journal of Intelligent Learning Systems and Applications, 2012, 4, 207-215
http://dx.doi.org/10.4236/jilsa.2012.43021 Published Online August 2012 (http://www.SciRP.org/journal/jilsa)

A Multi-Agent Approach to Arabic Handwritten Text Segmentation

Ashraf Elnagar*, Rahima Bentrcia

Computer Science Department, College of Sciences, University of Sharjah, Sharjah, UAE.
Email: *ashraf@sharjah.ac.ae

Received November 24th, 2011; revised April 19th, 2012; accepted April 26th, 2012

ABSTRACT

The segmentation of individual words into characters is a vital process in handwritten character recognition systems. In this paper, a novel approach is proposed to segment handwritten Arabic text (words). We consider the "Naskh" font style. The segmentation algorithm employs seven agents in order to detect regions where segmentation is illegal. Feature points (end points) are extracted from the remaining regions of the word-image. Initially, the middle of every two successive end points is considered as a candidate segmentation point based on a set of rules. The experimental results are very promising as we achieved a success rate of 86%.

Keywords: Character Segmentation; Handwritten Recognition Systems; Multi-Agents; Arabic Handwriting

1. Introduction

For the past three decades, there has been increasing interest among researchers in problems related to handwritten text segmentation and recognition regardless of the language used [1]. Most of the handwriting recognition systems are based on segmentation, which is the operation that seeks to decompose a word image into a sequence of sub-images containing isolated characters. Despite of the extensive work done on the off-line recognition of handwritten Latin and Asian languages text, a small number of research papers and reports are published in the recognition of Arabic handwriting [2]. This is probably a result of a lack of adequate support in terms of funding, and other utilities, such as comprehensive and standard Arabic text databases, dictionaries, etc.; and certainly due to difficulties associated with Arabic handwritten text segmentation such as the cursive nature of Arabic handwriting where most of the characters in a single word are connected to each other. Another difficulty is the existence of overlapping characters which are not attached to each other but share horizontal space. Due to difficulties mentioned above, many researchers bypass the segmentation stage in developing a recognition system. However, this is not practical and insufficient in applications that require recognition of a large number of vocabularies where several words may have the same global shape, such as bank check processing, postal address and zip code recognition [3,4], automated handwritten document entry and understanding, mail sorting, and other business and scientific applications. In addition, segmentation has an effective role in reducing the complexity of recognition systems since the number of recognition classes will be the number of Arabic letters and not the possible combinations of them.

In this paper, we address the problem of segmenting Arabic handwritten words into characters. The proposed approach utilizes seven agents which cooperate to identify the regions where the insertion of segmentation points is illegal. The segmentation algorithm is described in **Figure 1** as a block diagram. First, the image of the

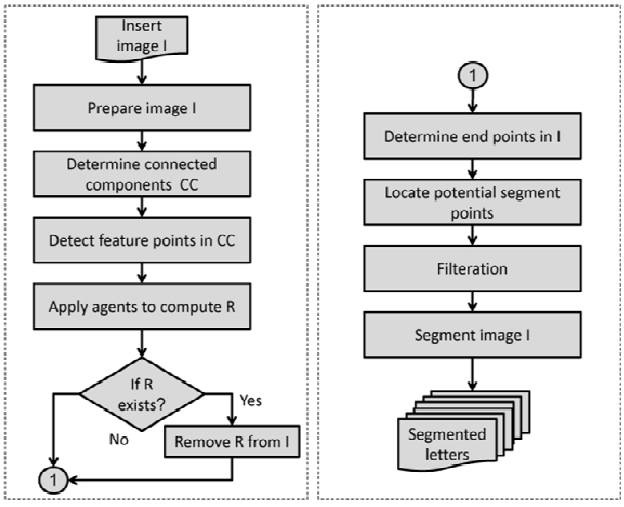

Figure 1. The basic steps in the algorithm.

*Corresponding author.

Arabic handwritten text is preprocessed and segmented into lines using horizontal projection. Next, each line is segmented into words/sub-words using vertical histogram, where each range of continuous black pixels is considered as a single word/sub-word. The resulting words are then thinned and the feature points are extracted in order to be used by agents in character segmentation. Each agent is responsible for detecting a specific region where segmentation is not permitted. These regions share general characteristics with Arabic letters such as loops and cavities, as shown in **Table 1**. After pulling out the detected regions, end-point features are extracted from the remaining regions of the word. The center point of every two successive end points is considered as a candidate cutting point. Finally, heuristic rules are applied to verify the validity of the suggested cutting points.

The rest of the paper is organized as follows. Section 2 presents related work; Pre-processing stage is described in Section 3; The seven agents are introduced in Section 4; Section 5 presents the segmentation stage. Experimental results are discussed in Section 6; and conclusions and future directions are presented in Section 7.

2. Related Work

Segmentation stage is a crucial task in any character recognition system and especially Arabic handwritings recognition. Its difficulty stems from the high variability of writing scripts and styles which could be affected by writers' health and mood. An extensive research is made and many studies appeared very early in the field of character recognition. An elaborate survey about character recognition systems is done by Amin [2]. A survey on character segmentation techniques and methods may be found in [5]. A multi-agents approach to separating handwritten connected digits is described in [6]. An overview of methods for separating handwritten characters is described in [7].

Three main approaches for segmenting a word into characters are defined in [8]: external segmentation, internal segmentation and holistic. External segmentation is the most commonly used approach to segmentation, where the possible letter boundaries are found prior to recognition.

In internal segmentation, both segmentation and recognition of letters are accomplished at the same time. In holistic based algorithms (also called no segmentation) general features of the whole word are extracted in order to recognize it and no characters are extracted.

Table 1. The main shapes that construct Arabic letters.

Loop	م ـه ـط ـظ
Cavity	ل ـن ـي ـد ـذ ـح ـخ ـج ـع ـغ ـس ـش ـر ـز
Loop and Cavity	ف ـق ـو ـص

Srihari [9] introduced one of the first external segmentation algorithms which segmented the Latin word at the minima of the lower contour of the writing. However, no performance results were reported for his algorithm.

Another segmentation method [10] is based on using mathematical morphology tools, namely singularities and regularities, where Amin, Motawa and Sabourin determined singularities by applying an opening to the Arabic handwritten word image and regularities by subtracting the singularities from the original image. These regularities are the candidates for segmentation. The algorithm achieved an accuracy of 81.88%. However, in some cases, it extracted two characters from the same segment. The same method is used for Latin handwriting [11,12].

Blumenstein and Verma applied a simple heuristic segmentation algorithm to Latin texts [13,14]. It finds segmentation points in printed and cursive handwritten words by looking for minimas or arcs between letters which can be the ideal segmentation points.

Another neuro-heuristic approach is used by Hamid and Haraty for segmenting handwritten Arabic text [15]. It is based on extracting connected block of characters and looking for topographic features to identify possible segmentation points. The distribution of these points is based on the average character width in the block. The system achieved an accuracy of 69.72%, but two major problems were encountered. The first problem is that the system failed in segmenting horizontal overlapping characters. Second, segmentation of handwritten Arabic text depends largely on contextual information, and not only on topographic features extracted from characters, which has not been dealt with in the system.

A new segmentation method is proposed by Bhowmik and Roy to segment Bangla words [16]. They traced the lower contour of each connected component in handwritten word-image anticlockwise. During this process, relevant features are extracted and their vectors are normalized. The system achieved an accuracy of 88% but it suffered from under segmentation problem. Another segmentation method was proposed by Lorgio and Govindaraju to segment Arabic handwriting [17]. The algorithm over-segments each word and then it removes extra breakpoints using knowledge of letter shapes. The accuracy of this system is 92.2% but it suffers from over segmentation. Comparing methods of internal and external segmentation approaches mentioned in the literature, experiments show that external segmentation provides greater interactivity, savings of computation, and simplifies the job of the recognizer [8]. In [18], we use a recognition-based system to segment handwritten Arabic text.

Because of the advantages mentioned above, the segmentation method in our proposed work will follow the

external segmentation approach. Moreover, our method is not only expected to resolve the shortcomings of the previous related methods but also to achieve better results by avoiding under segmentation. This depends on agents that extract the appropriate features of Arabic handwritings which improve detecting candidate segmentation points. Our proposed system is a recognition-free segmenting Arabic handwritten system. The proposed system employs multi-agents in which agents cooperate to identify potential segmentation points. In addition, heuristic rules are applied to verify the validity of the suggested cutting points. The accuracy of this recognition-free system is 86%.

3. Preprocessing Phase

The image of the Arabic handwritten text is first scanned and converted into a binary format. Then, the image is cleaned by removing any noise produced by the optical scanning device or the writing instrument. The handwritten text is then segmented into lines using horizontal projection of black pixels on the Y-axis. Each line is finally segmented into words/sub-words by projecting vertically black pixels of the line onto the X-axis. **Figures 2** and **3** show line and word segmentation, respectively. The unit used is pixels for both figures.

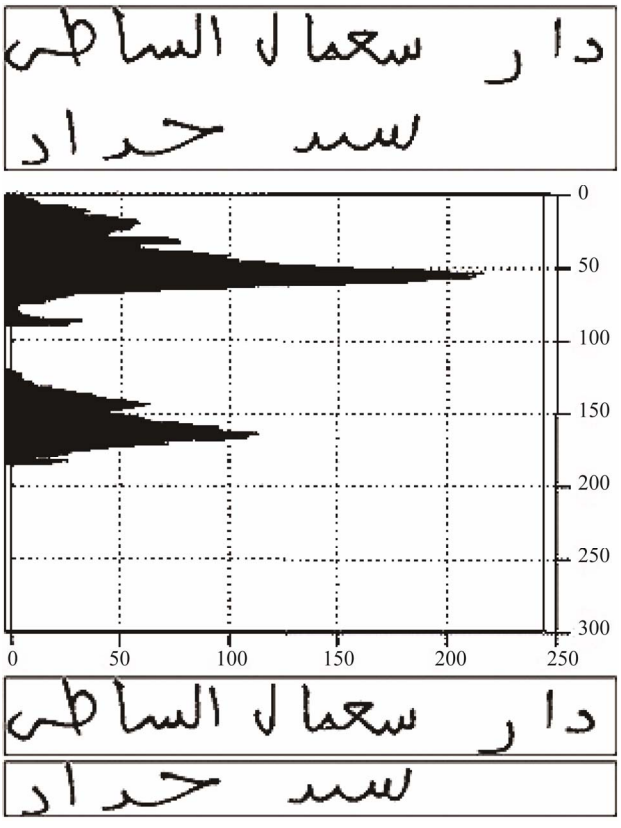

Figure 2. Two lines of text; horizontal projection of the both lines; and the resulting separate lines of text.

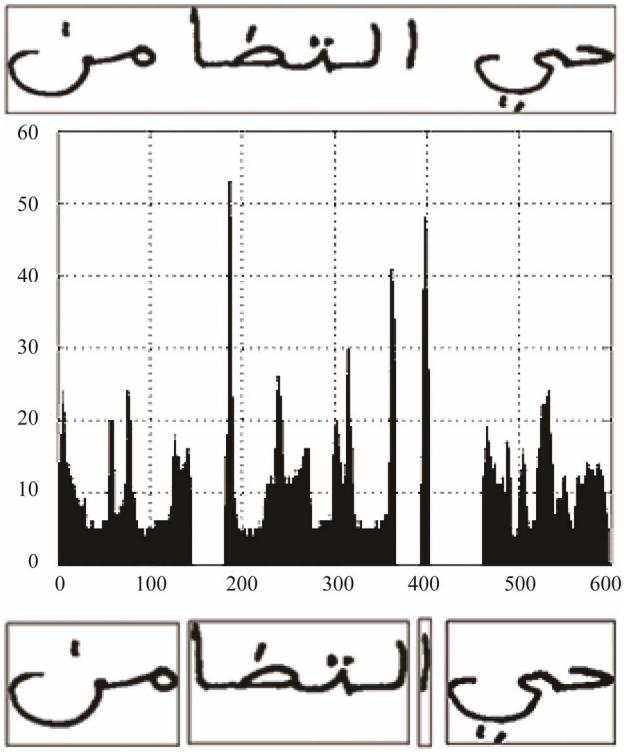

Figure 3. A line of text that consists of 4 parts; vertical projection; and the resulting separated sub-words.

Next, thinning is applied to the resulting word where the width of word's boundary is reduced to one pixel. This step is fundamental since it is needed by agents for a successful extraction of feature points.

Figure 4 shows an Arabic handwritten word before and after thinning. Finally, main connected components of the word are extracted from the whole image to isolate extra parts of the image from the handwriting.

Feature Points Extraction

Three types of feature points are extracted by agents: end point features, branch point features, and cross point features. An end point is the start or the end of a line segment. A branch point is the intersection point in letter "T"; it connects three line segments. Cross point connects four line segments, similar to the intersection point in "+" sign, rotated by different angles.

Predefined template shapes are used to extract feature points, where each pixel in the image is mapped onto its correspondent in the template looking for similarity. An

Figure 4. Original input and thinning result.

example of a word with feature points, demonstrated as bold black points, is illustrated in **Figure 5**.

4. The Employed Agents

The identification of initial cutting points strongly depends on seven agents. Six agents are major, which are:

1) Loop agent;
2) Letter Seen agent;
3) Under-baseline-cavities agent;
4) Above-baseline-right-cavity agent;
5) Above-baseline-left-cavity agent, and;
6) Above-baseline-narrow-left-cavity agent.

The other agent which is the baseline agent is a minor one since it is used by major agents to facilitate their task. The idea behind selecting these agents stems from the fact that Arabic characters consist of two main shapes: loop and cavity, as shown in **Table 1**.

However, the success of the proposed segmentation algorithm is based on the existence of feature points in the thinned handwritten word which is common in several Arabic handwriting scripts/styles. Our work is based on such families of fonts. The following sections introduce each agent in detail.

4.1. Baseline Agent

The baseline is the line where most of handwriting pixels lie. The task of the baseline agent is to detect the baseline of the handwriting before segmentation into words is performed. First, the agent counts the number of black pixels in each horizontal line in the image. Then, it finds the line with the maximum number of black pixels. Finally, it returns this line as the baseline of the word. On the other hand, the agent has a significant role in aiding the under-baseline-cavities agent to decide whether a cavity exists under the baseline of the word or not. **Algorithm 1** shows the basic steps in this algorithm.

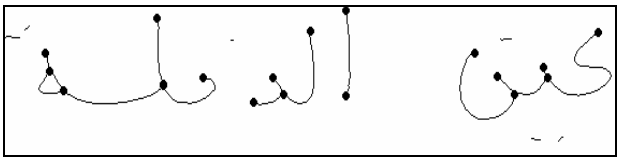

Figure 5. Feature points in the Arabic word "Ain Eddafla".

Input: image of the handwritten word
Output: the baseline of the input image
Steps:
1: for(each line ℓ in the image) **do**
count the number of black pixels
end
2: find ℓ (max(black pixels))
3: return baseline = ℓ
End algorithm

Algorithm 1. Baseline agent.

4.2. Loop Agent

Most of Arabic letters consist of loops which can be classified into: simple such as "صـ", "مـ", "ق" and complex like "ڤ" and "هـ". The loop agent is responsible for extracting loops from the handwritten word. Starting from the upper left of the image, the agent traces the word boundaries looking for continuous black pixels that enclose a loop. This process is terminated when the agent reaches the end of the word. The extracted group(s) of continuous black pixels, if any, is returned as a loop. See **Algorithm 2**.

4.3. Under-Baseline-Cavities Agent

Some Arabic words consist of cavities that lie under the baseline. The position of this type of cavities is always in the end of the main connected components of the word. **Figure 6** shows different situations of under-baseline-cavities. The baseline of each word is shown as a straight line, and the under-baseline cavities are circled. The handwritten word "صبيح", on the left of **Figure 6**, has only one main connected component which is terminated by a right cavity. In the middle, the word "مثلين" also consists of only one main connected component which is ended by an up cavity. Finally, on the right of the same Figure, the word "بوغزارة" has five main connected components. Three of them are ended by left cavities.

Hence, under-baseline-cavities can have four directions: left such as "ر" and "و", right like "ح" and "ع", straight such as "م" and up such as "س", "ن", "ص", "ل" and "ي". The detection of under-baseline-cavities needs the assistance of baseline agent. Under-baseline-cavities agent traces the boundary of the word looking for the deepest point in the boundary. Then, if the distance between this point and the baseline is larger than a predefined threshold, the agent continues tracing until it

Input: segments in the handwritten word
Output: loops that look like the Arabic letters: 'هـ،ه،و،ص،م،ق،ه' or any other letter that have a similar shape
Steps:
1: trace the boundary of the main component mC.
2: while(it is not end(mC)) **do**
search for black pixels bp around white pixels
if (bp exist) **then**
loop(++loopNo) = bp
end if
end while
3: return loop if one exists
End algorithm

Algorithm 2. Loop agent.

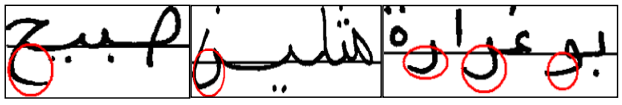

Figure 6. Under-baseline-cavities of different directions.

reaches the nearest feature point. **Algorithm 3** illustrates the main steps in the algorithm.

The existence of an under-baseline-right cavity in a word might lead to a problem called overlapping. It states that the letter with such cavity occupies and shares a space with the letter proceeds it. **Figure 7** illustrates the overlapping letters in circles. The handwritten word "صبيح", displayed in the figure on left, suffers from this problem, where the descendant part in letter "ح" shares a space with letter "ي". The same problem appears in the overlapping letters "ع" and "ي", in the word "الشرايع".

Some other characters that may lead to overlapping are Haa "ح", Khaa "خ", Jeem "ج", Ain "ع", and Ghain "غ". The agent treats this problem by eliminating the right cavity part (descender) before the segmentation process starts, as shown in **Figure 8**. After determining the cutting points, the descendant parts are retained to their locations.

Input: segments in the handwritten word
Output: loops that look like the Arabic letters: 'هـه،و،ص،م،ق،ه،' or any other letter that have a similar shape
Steps:
  Input: segments in the handwritten word
  Output: regions with descendents as the Arabic letters: "ل،ح،ي،ر،ع،س،و،ص،م،ق،ن،" or similar ones.
  Steps:
  1. extract the feature points from mC
  2. determine the baseline of the word.
  3. detect the deepest point p in the word
  4. if p lies below the baseline **then**
        starting from p, trace mC seeking a cavity cav
        if both ends of cav are end points e **then**
          if both have near equal height **then**
              the direction dir of cav is 'UP'
          else if the nearest e lies on the right of p
              the direction dir of cav='RIGHT'
          else if the nearest e lies on the left of
            dir='LEFT'
          end if
        else if one end is an end point e and the other one is a branch/cross point **then**
          if the nearest e lies on the right of p **then**
            dir='RIGHT'
          else if the nearest e lies on the left of p
            dir ='LEFT'
          else if cav is connected to a loop **then**
            dir='DOWN'
          end if
        end if
    end if
  5. return cav and dir if exist
  End algorithm

Algorithm 3. Under-baseline-cavities agent.

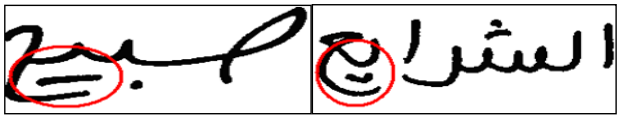

Figure 7. The overlapping problem in Arabic handwriting.

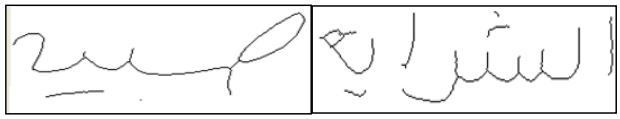

Figure 8. The right cavity extraction.

4.4. Above-Baseline-Right-Cavities Agent

This agent extracts the letters "ع", "ك" or any other letter that has a right cavity above the baseline. Feature points are also important in detecting this region, where the agent traces the boundaries of the word searching for pixels that form a right cavity shape. Each end point feature in the word is a candidate start of a right cavity. Therefore, the agent restarts tracing from each end point and keeps track of pixels which construct a right cavity. Finally, the agent seeks the help of the baseline agent in order to decide whether the extracted cavity is above the baseline or not. This agent is described in **Algorithm 4**.

4.5. Above-Baseline-Left-Cavities Agent

Letters that have above-baseline-left-cavity such as "د" are detected by this agent. Tracing the borders of the word starts from the upper left of the image following pixels of word's skeleton until a left cavity is detected then the agent stops searching. This is related to the fact that letters "د" and "ذ" always appear in the end of a main connected component. Therefore, the agent traces only that area of the word. The basic algorithmic steps of this agent are illustrated in **Algorithm 5**.

4.6. The Seen Agent

The task of this agent is similar to the previous agents. However, seen agent finds regions which look like "سد". The agent extracts the feature points from the main

Input: images of the main components
Output: regions similar to the Arabic letters: 'ء،ع،ىك' or any other letters that have similar shapes of them.
  Steps:
  1. extract end-point features e from mC, starting from the upper left corner
  2. for each end-point e **do**
      trace the boundary of mC starting from e
      search for pixels that form a cavity cav with
      dir = 'RIGHT'
      if right cav exists **then**
          find the baseline of the word using Algorithm 1
          if the right cav is above the baseline **then**
            return cav as above-baseline-cavity
          end if
      else
          no right cavity in the traced area
      end if
  end for
End algorithm

Algorithm 4. Above-baseline-right-cavities agent.

Input: images of the main components
Output: regions similar to the Arabic letters: 'ذ،د' or similar ones.
Steps:
1. extract end-point features e from mC, starting from the upper left corner
2. if number of e ≥ 2 **then**
trace mC starting from most left e
search for pixels that form a cavity cav with dir ='LEFT'
if a left cavity exists **then**
find the baseline using Algorithm 1
if the left cav is above the baseline **then**
return cav as above-baseline-cavity
end if
else
no left cavity in the traced area
end if
end if
End algorithm

Algorithm 5. Above-baseline-left-cavities agent.

component of the word and determines if the number of the extracted end points features is a multiple of three since the letter "س" has three end points.

Then, it starts from the most left end point tracing the borders of the handwriting and seeking areas similar to the Arabic letter "س". If the distance between every consecutive three end points is within a threshold, then the detected region is the letter "س". These steps are clarified in **Algorithm 6**.

4.7. Above-Baseline-Narrow-Left-Cavities Agent

The presented agent extracts regions that look like "ح". One main difference between this agent and above-baseline-left-cavities agent which extracts "د" is that this agent searches the whole word looking for regions similar to "ح" and not only the end of word's main connected components. First, the agent mines the end points features from the main component. Then, it begins tracing the boundaries of the handwriting searching for

Input: images of the main components
Output: regions similar to the Arabic letters: 'س، ش' or similar ones.
Steps:
1. extract end-point features e from mC, starting from the upper left corner
2. if number of e ≥ 3 **then**
while number of untraced e = multiple of 3 **do**
trace the boundary of the main component mC starting from the most left untraced e
search for pixels that form the shape of the Arabic letter 'س'
if 'س' shape exists **then**
return the extracted area as 'س'
end if
find the number of untraced e
end while
else
no letter 'س'in the traced area
end if
End algorithm

Algorithm 6. The "seen" ('س') agent.

"ح" region. The baseline agent is necessary to decide whether the extracted area is above the baseline or not. **Algorithm 7** shows the main steps in this algorithm. **Figure 9** displays the regions extracted by loop agent in rectangles, and the regions extracted by under-baseline-cavities agent in circles.

5. The Segmentation Phase

The goal of this stage is to find the initial candidate cutting points that separate word's characters correctly. After agents detect regions that look like some of Arabic characters, these regions are subtracted from the whole word and the remaining parts are left for further processing, as shown in **Figure 10**.

Next, end point features are extracted from the remaining parts of the handwritten word. This step is important to locate potential segmentation places; the center of every two successive end points is a candidate segmentation point, see **Figure 11**.

Input: images of the main components
Output: regions similar to Arabic letters: 'حخجچ' or any similar ones.
Steps:
1. extract end-point features e mC, starting from the upper left corner.
2. for each end-point e **do**
trace the boundary of mC starting from e
search for pixels that form a cavity cav with dir = 'LEFT'
if a left cavity cav exists **then**
if the height of cav ≤ threshold then
find the baseline using Algorithm 1
if the left cav is above the baseline **then**
return cav
end if
else
no narrow cav exists in the traced area
end if
else
no left cav in the traced area
end if
End algorithm

Algorithm 7. Above-baseline-narrow-left-cavities agent.

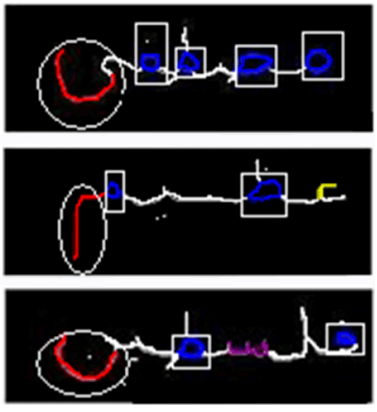

Figure 9. The resulting regions of agents cooperation.

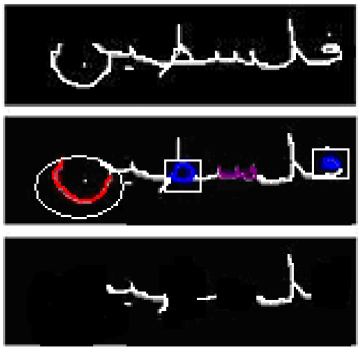

Figure 10. The remaining regions after agents cooperation.

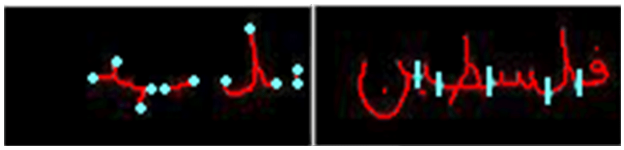

Figure 11. The insertion of segmentation points.

Finally, a verification process is applied to the set of candidate cutting points to remove the extra ones. Cutting points that segment the word more than once are eliminated from the candidate set. This case is illustrated in **Figure 12**.

In addition, if the width of a resulting segment is smaller than a threshold, the most left cutting point is excluded and the word is re-segmented based on the new filtered set of candidate cutting points. See **Figure 13**.

6. Experimental Results

Our experimental results were conducted using a sample of 550 images of Arabic handwritten words, written by different people. The sample was taken from IFN/ENIT

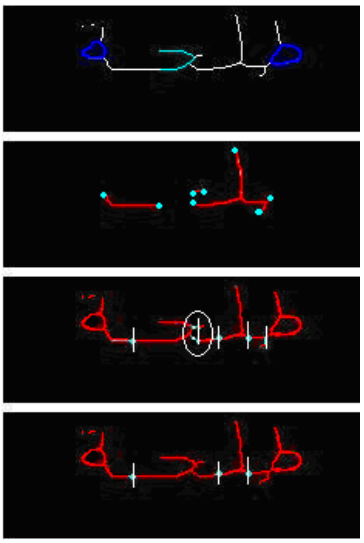

Figure 12. The final segmentation points after the filtering process was performed.

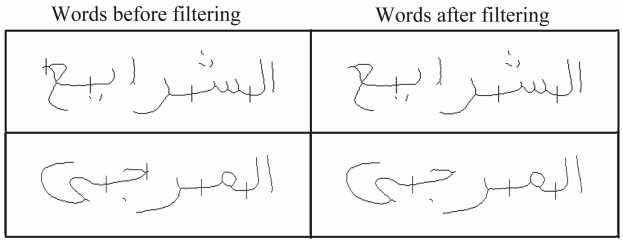

Figure 13. Segmented words before and after the second filtering process.

database and selected to represent words with different length and characters. As summarized in **Table 2**, 74% of the selected words consist of loops, 84% include under-baseline cavities with different shapes, and 54% contain above-baseline cavities with left and right directions. The segmentation results are classified into three classes: words that are correctly-segmented, words that are over-segmented, and words that are wrongly or under-segmented. **Table 3** displays the percentage of the identified segmentation points in each class, as follows.

The following figures display segmentation of words with loops (**Figure 14**), words with cavities (**Figure 15**), and words with combinations of both (**Figure 16**).

Table 2. The sample DB: % of words with.

loops	74% contain 678 loops with different shapes: complex, circular, elongated, and triangular.
under-baseline cavities	84% contain 654 under-baseline cavities with different directions: up, left, and right.
above-baseline cavities	54% contain 350 above-baseline cavities with different directions: left and right.

Table 3. The segmentation results using the selected sample.

Resulting Segmentation Classes	Percentage of Each Class
Correct Segmentation	86%
Over Segmentation	≈13.7%
Missing/Wrong Segmentation	0.3%

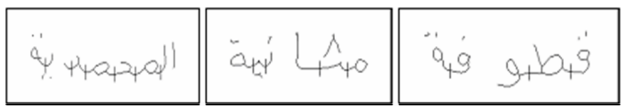

Figure 14. Segmentation of words with loops.

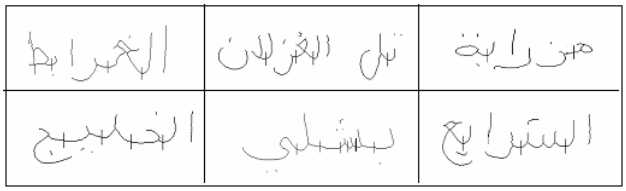

Figure 15. Segmentation of words with cavities.

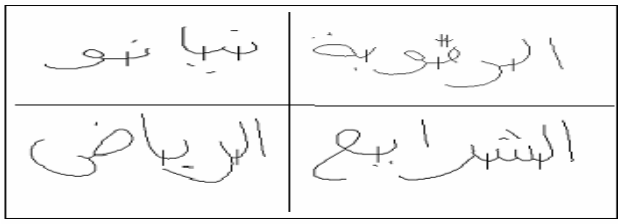

Figure 16. Segmentation of words: Of loops and cavities.

Words in **Figure 14** are successfully segmented, except the left word "المحمدية" where the letter "د" was over-segmented because of its distorted shape.

Another example is demonstrated in **Figure 15**. The agents work well in detecting their related regions and hence the segmentation was done successfully. The same results were achieved in **Figure 16**.

Other segmentation results are displayed for words with under-baseline cavities (**Figure 17**) and above-baseline cavities (**Figure 18**). As obtained in the experimental results, the agents correctly detected the cavities with their directions and segmentation locations were precisely determined.

In addition, the segmentation results of one word written by different writers are shown in **Figures 19-21**.

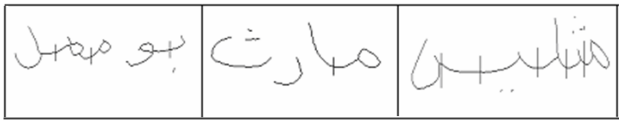

Figure 17. Segmentation with under-baseline cavities.

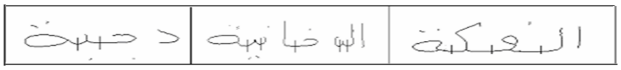

Figure 18. Segmentation with above-baseline cavities.

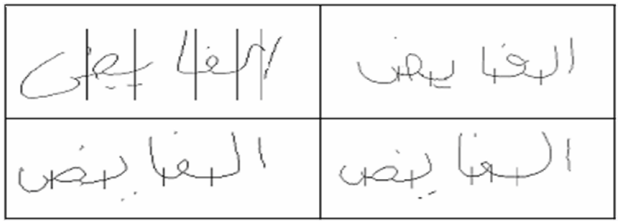

Figure 19. The segmentation of Arabic word "Elfayedh", written by four writers.

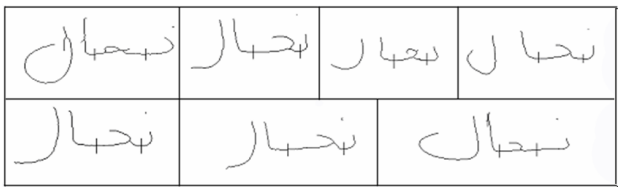

Figure 20. The segmentation of Arabic word "Nahhaal", written by seven writers.

Figure 21. The segmentation of Arabic word "Elmanzah", written by nine writers.

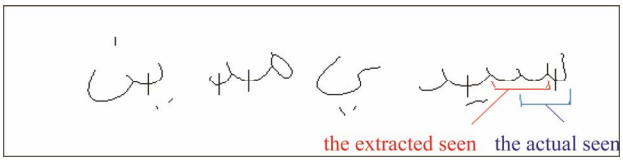

Figure 22. The wrong detection of letter seen location.

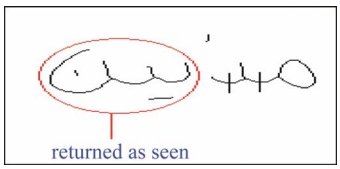

Figure 23. The misdetection of letter seen.

The above three figures illustrate the role of neat handwriting in obtaining better segmentation results. **Figure 19** suffered from over-segmentation. The letter "ض" was considered as two letters, one consists of a loop and the other has a cavity. Therefore, a segmentation point was inserted in between. More examples also can be seen in **Figure 21**, in the word "المنزه". The upper right example was over-segmented whereas the rest were correctly separated into letters.

The wrong/missing segmentation cases, shown in **Table 3**, were detected in scenarios, where the letter seen "س" exists or some other letters that look like it and wrongly detected as letter seen. **Figure 22** clarifies the wrong detection of letter seen location in the Arabic word "seedi madyan".

The word shown in **Figure 23** depicts the case of misdetection of letter seen. This problem appears when a number of adjacent letters are extracted by the Seen agent and returned as letter seen because they have the same features as it.

7. Conclusions

We introduced a multi-agents approach to segment Arabic handwritten text. Our proposed methodology employs seven agents that are applied on the thinned image of the handwritten text; the agents cooperate to identify

illegal segmentation points. Specific set of topology-based rules are used to the remaining parts of the thinned image to determine the set of potential cut-points that will lead to a successful segregation of individual characters. Experimental results are not only encouraging but also very promising.

Moreover, we handled the cases of text (each word) over-segmentation by combining the resulting segments to form complete characters. Our future work is to include more styles of Arabic handwritings and more agents to improve segmentation results.

REFERENCES

[1] B. Verma, M. Blumenstein and S. Kukarni, "Recent Achievements in Off-Line Handwriting Recognition Systems," *International Conference on Computational Intelligence and Multimedia Applications* (*ICCIMA*'98), Melbourne, 1998, pp. 27-33.

[2] A. Amin, "Off-Line Arabic Character Recognition: The State of the Art," *Pattern Recognition*, Vol. 31, No. 5, 1998, pp. 517-530. doi:10.1016/S0031-3203(97)00084-8

[3] L. Lam, *et al.*, "Automatic Processing of Information on Cheques," *International Conference on Systems*, *Man & Cybernetics*, 1995, pp. 2353-2358.

[4] A. Brakensiek, J. Rottland and G. Rigoll, "Confidence Measures for an Address Reading System," *Seventh International Conference on Document Analysis and Recognition* (*ICDAR*2003), 2003, pp. 294-298.

[5] R. G. Casey and E. Lecolinet, "A Survey of Methods and Strategies in Character Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 18, No. 7, 1996, pp. 690-706. doi:10.1109/34.506792

[6] R. Alhajj and A. Elnagar, "Multiagents to Separating Handwritten Connected Digits," *IEEE Transactions on Systems*, *Man and Cybernetics*, *Part A*: *Systems and Humans*, Vol. 35, No. 5, 2005, pp. 593-602. doi:10.1109/TSMCA.2005.843389

[7] Y. Lu and M. Shridar, "Character Segmentation in Handwritten Words—An Overview," *Pattern Recognition*, Vol. 29, No. 1, 1996, pp. 77-96. doi:10.1016/0031-3203(95)00072-0

[8] C. C. Tappert, C. Y. Suen and T. Wakahara, "The State of The Art in On-line Handwriting Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 12, No. 8, 1990, pp. 787-808. doi:10.1109/34.57669

[9] S. Srihari and R. Bozinovic, "A Multi-Level Perception Approach to Reading Cursive Script," *Artificial Intelligence*, Vol. 33, No. 2, 1987, pp. 217-255. doi:10.1016/0004-3702(87)90035-X

[10] D. Motawa, A. Amin and R. Sabourin, "Segmentation of Arabic Cursive Script," *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, Ulm, 18-20 August 1997, Vol. 2, pp. 625-628.

[11] M.-Y. Chen, A. Kundu and J. Zhou, "Off-Line Hand-Written Word Recognition Using a Hidden Markov Model Type Stochastic Network," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 16, No. 5, 1994, pp. 481-496. doi:10.1109/34.291449

[12] M.-Y. Chen, A. Kundu and S. N. Srihari, "Variable Duration Hidden Markov Model and Morphological Segmentation for Handwritten Word Recognition," *IEEE Transactions on Image Processing*, Vol. 4, No. 12, 1995, pp. 1675-1688. doi:10.1109/83.477074

[13] M. Blumenstein and B. Verma, "A Neural Based Segmentation and Recognition Technique for Handwritten Words," *Proceedings of IEEE World Congress on Computational Intelligence*, *WCCI*'98, Anchorage, 4-9 May 1998, pp. 1738-1742.

[14] M. Blumenstein and B. Verma, "An Artificial Neural Network Based Segmentation Algorithm for Off-Line Handwriting Recognition," *International Conference on Computational Intelligence and Multimedia Applications* (*ICCIMA*'98), Melbourne, 1997.

[15] A. Hamid and R. Haraty, "A Neuro-Heuristic Approach for Segmenting Handwritten Arabic Text," *International Conference on Computer Systems and Applications*, *ACS*/*IEEE*, Beirut, 25-29 June 2001, pp. 110-113.

[16] T. K. Bhowmik, A. Roy and U. Roy, "Character Segmentation for Handwritten Bangla Words Using Artificial Neural Network," *Proceeding of the First IAPR TC*3 *NNLDAR Workshop*, Seoul, 2005.

[17] L. Lorgio and V. Govindaraju, "Segmentation Pre-Recognition of Arabic Handwriting," *Proceedings of the Eighth International Conference on Document Analysis and Recognition*, 29 August-1 September 2005, Vol. 2, 2005, pp. 605-609. doi:10.1109/ICDAR.2005.207

[18] A. Elnagar and R. Bentrecia, "Recognition-Based Segmentation of Arabic Handwriting," *the* 9*th International Workshop on Pattern Recognition in Information Systems*, Milan, 6-7 May 2009.

