I.J. Image, Graphics and Signal Processing, **2016, 12, 8-20**
Published Online December 2016 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijigsp.2016.12.02

Graph Modeling based Segmentation of Handwritten Arabic Text into Constituent Sub-words

Hashem Ghaleb[1]
[1]Department of Studies in Computer Science, University of Mysore, Mysore, India
Email: hashemx86@gmail.com

P. Nagabhushan[1] **and Umapada Pal**[2]
[2]CVPR Unit, Indian Statistical Institute, Kolkata, India
Email: pnagabhushan@hotmail.com, umapadapal@isical.ac.in

***Abstract*—**Segmentation of Arabic text is a major challenge that shall be addressed by any recognition system. The cursive nature of Arabic writing makes it necessary to handle the segmentation issue at various levels. Arabic text line can be viewed as a sequence of words which in turn can be viewed as a sequence of sub-words. Sub-words have the frequently encountered intrinsic property of sharing the same vertical space which makes vertical projection based segmentation technique inefficient. In this paper, the task of segmenting handwritten Arabic text at sub-word level is taken up. The proposed algorithm is based on pulling away the connected components to overcome the impossibility of separating them by vertical projection based approach. Graph theoretic modeling is proposed to solve the problem of connected component extraction. In the sequel, these components are subjected to thorough analysis in order to obtain the constituent sub-words where a sub-word may consist of many components. The proposed algorithm was tested using variety of handwritten Arabic samples taken from different databases and the results obtained are encouraging.

***Index Terms*—**Arabic Handwriting Recognition; Arabic Sub-words; Sub-word Segmentation; Connected Component Extraction; Graph theoretic modeling

I. Introduction

Optical Character Recognition is the process of transforming text from the iconic form of writing to its symbolic form [1]. Numerous research results in recognizing printed and handwritten text have been reported especially for Latin and Chinese scripts. However, the state of the art for Arabic text recognition falls far behind [2,3] despite the fact that Arabic letters, along with few additional letters, are used to transcribe several languages such as Persian and Urdu. Arabic script, written from right to left, is cursive in both handwritten and printed forms. Arabic Alphabet contains 28 letters. Letters are joined to form a word (see Fig. 1(a)). However, there are 6 letters (ا, د, ذ, ر, ز, and و) which can be joined only to the letter preceding them but not to the succeeding letter. This conditional joining property leads to the emergence of PAWs (Parts of Arabic Words) [6]. A PAW, or a sub-word, is a sequence of Arabic letters that are joined together. Generally, an Arabic word consists of one or more sub-words (for example, the word in Fig. 1(a) consists of 1 sub-word whereas the word in Fig. 1(b) consists of 3 sub-words). A sub-word, in its essence, may consist of several components which occurs as a results of the intrinsic nature of Arabic letters where several letters share the same basic shape; the dot and Hamza components are used to distinguish such letters. These components are known in the literature as primary component (main body of the sub-word) and secondary components (dots and diacritics) (the word in Fig. 1(c) consists of two sub-words; the right sub-word has only one component whereas the left one consists of 4 components; one component corresponding to the main body and three to the dots). Additionally, the sub-words might horizontally overlap; i.e., share the same vertical space without being touching (see Fig. 1(d), overlapping is highlighted by encircling). This overlapping induces problems for both the word and the character segmentation [4,5].

(a) (b) (c) (d)

Fig.1. Arabic words.

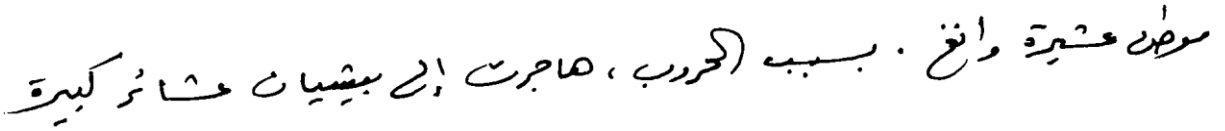


Fig.2. Arabic Text line.

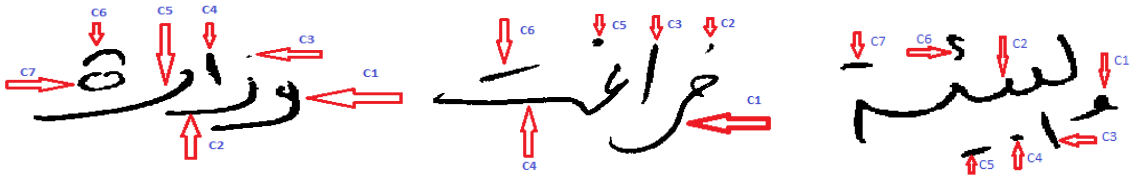
Overall, the Arabic text line can be viewed as a sequence of words which in turn are a sequence of sub-words (see Fig. 2). The sequence of sub-words usually runs from right to left with the secondary components associated with a particular main body of a sub-word are confined within the boundary of the main body. However, there might be some displacement of the secondary components in the handwriting scenario. Since segmentation of word into characters is very challenging task [18], we propose to work on considering sub-word itself as a basic unit for processing. Since a sub-word in a word in principle should happen to be the disconnected components, although segmentation by projection as said above could be difficult, we resort to graph theoretic modeling for the analysis of connected and disconnected components in coherence with which the graph theory is utilized to map complex problems into simple representations and models which allows for the definition of optimal solution techniques for such problems [16]. Thus the work proposed in this paper is devoted for segmenting text line into its sub-words utilizing the concepts of graph theory.

The rest of the paper is organized as follows- related literature is presented in section 2. The different stages of the proposed algorithm are described in section 3. The experimental results are reported in section 4. Comparative results are reported in section 5. Finally, the paper is concluded in section 6.

II. Related works

Arabic handwriting recognition systems proposed in the literature can be divided into segmentation-based (Analytical) and segmentation-free (Holistic) approaches. In segmentation-based recognition systems the word is segmented into smaller units (characters, graphemes, or primitives), and then these units are recognized. On the contrary, segmentation-free systems consider the whole image as the unit of recognition.

The property of Arabic writing where a word is separated into many pieces (i.e., sub-words) has been utilized by the researchers for performing variety of tasks such as word recognition, word spotting, and lexicon reduction. Motivated by the Arabic letters' conditional joining property, Abdulkader introduced a two-tier approach for the recognition of handwritten Arabic text in [6] wherein an Arabic word is looked at as being composed of a sequence PAWs. PAWs can be expressed in terms of letters. The recognition problem is decomposed into two problems that are solved simultaneously. To find the best matching word for an input image, a Two-Tier Beam search is performed. In Tier one the search is constrained by a letter to PAW lexicon. In Tier two, the search is constrained by a PAW to word lexicon. In [21], characteristic loci features were used to cluster printed Farsi (Persian) sub-words based on their shapes yielding a pictorial dictionary which according to the authors can be used in a word recognition system. However, the authors did not address the segmentation of text into sub-words explicitly. Instead, a dataset of sub-words is created and later used to build the dictionary. A word spotting system for handwritten Arabic documents which adapts to the nature of Arabic writing is introduced in [22]. The system recognizes PAWs. Then, it reconstructs and spots words using language models. Number of PAWs, along with number and position of dots, were used as inputs to a lexicon pruning method in [8]. A lexicon-reduction strategy for Arabic documents based on the structure of Arabic sub-word shapes which is described by their topology is introduced in [7].There exist few attempts in the literature which aimed at segmenting the Arabic word/ text line into the composing sub-words. Vertical histogram of line image is employed to segment the printed Arabic text line into sub-words [20]. The strategy for segmenting the text line into sub-words in [14,15] is based on vertical projection of black pixels of the line onto the X-axis. Such strategy has the shortcoming of being unable to handle the overlapping nature of the sub-words. Parvez et al [3] addressed the issue of sub-word segmentation as part of recognition system which intends to recognize handwritten Arabic words. Firstly, all connected components in the line are extracted. This is followed by baseline estimation. Later on, the baseline information is used to determine the primary components of the sub-words. All other components are considered to be secondary components. After that each secondary component is assigned to a primary component using set of predefined rules. Finally, each primary component along with the associated secondary components are marked as PAW and passed to the subsequent stages of character segmentation and recognition. The technique used for sub- word segmentation in [9] proceeds as follows: word baseline is estimated using the horizontal projection histogram method. Then, the main and secondary bodies are identified; main bodies of the sub-words are extracted and the secondary bodies are assigned using predefined rules to their respective main bodies to yield the sub-word.



(a) (b) (c)

Fig.3. Arabic script based words.

Adapting the connected components analysis to segment Arabic word/text line into sub-words seems to be more appealing. However, the existing methods are essentially dependant on the strict identification of a component as either primary or secondary component which is determined by the size of the component and its distance from the baseline; the guiding criterion for such characterization is based on the relatively small size of secondary components and its far distance from the baseline. Furthermore, predefined rules are applied to associate the secondary components to their respective primary components. It could be argued that applying such rigid rules is an error-prone process especially in the handwritten text scenario. For example, the component C4 (a primary component) in Fig. 3(a) is smaller than the component C6 (a secondary one) and the component C3 (a primary component) in Fig. 3(b) has smaller size than the component C6 (a secondary component). It can be also observed that the component C6 is not so far from the baseline. The component C1 in Fig. 3(c) which constitutes a complete sub-word also has a relatively small size.

In this paper we propose an algorithm to segment a text line into its constituent sub-words based on the graph theoretic analysis of connected components. The proposed algorithm is solely inspired by the sequence of Arabic writing which allowed us to introduce a simple yet efficient technique to accomplish the segmentation. To handle the situation when the secondary components are displaced, an additional refinement stage is incorporated into the algorithm. Furthermore, the proposed algorithm is flexible in the sense it might associate a secondary component with more than one primary component ensuring that the secondary component is associated with its primary component and leaving the issue of resolving the ambiguity to the subsequent stages of the recognition system.

III. Methodology

The proposed algorithm comprises of four stages: preprocessing, extraction of connected components, segmentation of text line into sub-words, and refinement stage. The details of each stage are presented in the following sub-sections.

A. Preprocessing

The goal of the preprocessing is to prepare the image for subsequent stages. To close gaps and fill small holes, morphological closing operation is applied on the image. Furthermore, two rows containing background pixels (OFF pixels) are inserted (padded) to the image (one at the top and the other at the bottom). Similarly two columns (one to the left and one to the right) are padded to the image (The necessity of such operation will be highlighted later). The sample images, on which the experimentation is carried out, have been drawn from datasets which are already binarized and de-noised [10-13]. Hence, no further binarization or noise removal is applied.

B. Extraction of Connected Components

In a binary image it is important to identify the foreground components present in the image. The general approach for tackling this problem is to associate with each pixel of a connected component a label by which the component is identified [16]. Different techniques have been proposed for accomplishing such labeling. The output of such operation is an image of the same size as the input image. A final scan of the labeled image may be required to identify the pixels associated with each component adding an additional burden to the task of labeling which itself might require more than one pass of scanning the input image. Unlike many methods existing, the connected component extraction algorithm proposed here goes directly to the extraction of the set of foreground components in just a single pass where each foreground component is retained in terms of the pixels comprising such component. Moreover, the algorithm is compatible with the alignment of Arabic writing which further makes the subsequent analysis of connected components towards achieving the task of sub-word segmentation easier. In this stage the connected components are extracted. Each component is retained in terms of coordinates of pixels which belong to the component. Towards this, the concept of grid graph [16] is utilized to represent the binary image as a graph. Given a set of pixels F in the image and a connectivity relationship on which a digital topology is based, the grid graph G (V,E) (where V is the set of vertices and E is the set of edges) of the image is defined as follows [16]:

i. To every pixel in F there corresponds a vertex in V.
ii. An edge (u,v) exists in E whenever the pixels p and q corresponding to pixels u and v are neighbors in the digital topology.

The above mentioned concept is utilized in our work where the set F is defined to be the foreground pixels (ON pixels) in the binary image and the 8-neighborhood is used. This representation (i.e., representation of the image as a grid graph) is used to extract the connected components existing in the image. The algorithm

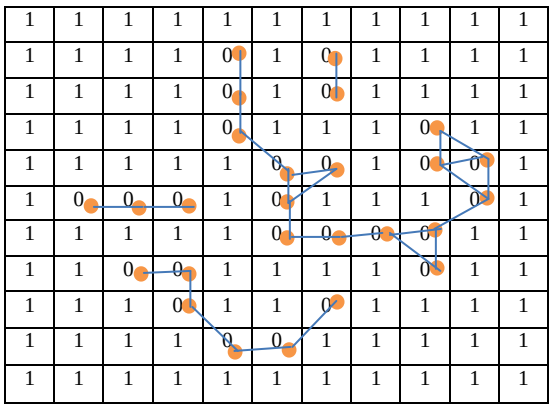
proceeds as follows: the image is scanned from top to bottom and from right to left. Upon encountering the first foreground pixel, its coordinates are stored in a list which would later contain all the pixels connected to this pixel (the coordinate system as defined in [17], in which a pixel is said to be of coordinates (x,y) where the first element of coordinate tuple refers to a row and the second element refers to a column, is used) . The pixels in 8-neighborhood of such pixel are traced; if any pixel in the neighborhood is found to be a foreground pixel, then its coordinates is also stored in the same list. The scanning continues and if a foreground pixel which does not belong to the list(s) available is encountered, a new list is created to retain the coordinates reflecting the possibility of such pixel belonging to a different component. Sometimes, two lists are merged; when a pixel which is yet to be inserted into a given list already belongs to another list which indicates that the pixels contained in the two lists belong to the same component. The significance of padding background pixel to the input image is to avoid repeated checking of a foreground pixel being on the border of the image. It also makes the tracing of 8-neighborhood pixels more consistent instead of being dependent on the position of foreground pixel. Finally, it is worth noting that the connected components are extracted in the offline mode and hence the image can be scanned from left to right as well. However, to align with direction of Arabic writing and to facilitate the subsequent stage of sub-word segmentation we opted for right to left scanning.

Algorithm: Connected Component Extraction
Input: I, binary image
Output: $S=\{C1,C2,\ldots,C_n\}$, set of connected components where $C_i=\{(x_1,y_1),(x_2,y_2),\ldots..,(x_k,y_k)\}$ is the set of coordinates of pixels comprising the component C_i.
Method
Step 1: Apply preprocessing on I
Step 2: Dummy_List←Φ
Step 3: Pool_of_Lists<-Dummy_List
Step 4: Scan the image column-wise from right to left
Step 5: if pixel is OFF goto Step 4
elsegoto Step 6
Step 6: If pixel is not appearing in any list included in the Pool_of_Lists
Create a new list
Store pixel coordinated in the list
Update Pool_of_Lists
Traverse the neighbors in the 8-neighborhood
for each ON pixel in the 8-neigborhood do
if pixel is common with another list
Merge the two lists
goto Step 4
else
Store the coordinates of the pixel in the new list
goto Step 4
end for
else
goto Step 7
Step 7: Identify the list where the pixel exists
Traverse the neighbors in the 8-neighborhood
for each ON pixel in the 8-neighborhood do
if neighbor pixel exists in the same list where the pixel under consideration exists
goto Step 4
else if neighbor pixel is common with another list
Merge the two lists
goto Step 4
else
Store the coordinates of the neighbor pixel in the same (identified) list
goto Step 4
Algorithm ends

The above algorithm is explained with a simple example given in Fig. 4. The resembling sub-graphs are shown in Fig.4(b).

1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	0	1	0	1	1	1	1
1	1	1	1	0	1	0	1	1	1	1
1	1	1	1	0	1	1	1	0	1	1
1	1	1	1	1	0	0	1	0	0	1
1	0	0	0	1	0	1	1	1	0	1
1	1	1	1	1	0	0	0	0	1	1
1	1	0	0	1	1	1	1	0	1	1
1	1	1	0	1	1	0	1	1	1	1
1	1	1	1	0	0	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1

(a)



(b)

Fig.4. Sample input image.

Without losing generality it is assumed that the given image has been already pre- processed. Firstly, a dummy list is created. Then, the input image is scanned from top to bottom and from right to left. When the first foreground pixel (located at the coordinates (5,10)) is encountered, it is checked if such pixel exists in any list available (here it is dummy list only), therefore a new list C1 is created and the coordinates (5,10) are stored in the

list C1. The 8-neighborhood pixels of the pixel under consideration are traced and the pixels located at (4,9),(5,9) and (6,10) are stored in C1. The scanning continues with the pixel located at (6,10); it is found that (6,10) is already present in C1. Therefore, the foreground pixels neighboring such pixel are also stored in C1 which reflects the fact that these pixels belong to the same component to which pixel at (6,10) belongs. This means that C1 is updated and the pixel at (7, 9) is stored in C1 (the pixels at (5,9) and (5,10) are already stored in C1. Hence, no further action is needed). When the pixel at (2,7) is scanned, it is found that does not exist in any list available till now (i.e., dummy list and C1) hence a new list C2 is created and the coordinates (2,7) are stored in C2. Again, the 8-neighborhood of pixel is traced and coordinates (3,7) are stored in the list C2. Moving on to the pixel at (5,7) it is again found that it does not exist in any list hence a new list C3 is created and the coordinates of the current pixel; that is (5,7) and of the neighboring foreground pixels (5,6) and (6,6) are stored in C3. The algorithm proceeds to the pixel at (7,7). It is found that this pixel is already stored in C1. The 8-neighborhood is scanned. However, it is established that (6,6), an 8-neighbor of the pixel at (7,7) which is stored in C1, is present at C3. Hence, C1 and C3 are merged. Overall, the algorithm produces 4 lists corresponding to the 4 different foreground components present in the image. They are:

C1={(5,10), (4,9), (5,9), (6,10), (7,9), (7,8), (8,9), (7,7), (5,7), (5,8), (6,6), (7,6), (4,5), (2,5), (3,5)} ,
C2={(2,7), (3,7)} ,
C3={(9,7), (10,6), (10,5), (9,4), (8,4), (8,3)} and
C4={(6,4), (6,3), (6,2).

Upon the completion of connected components extraction, we obtain the set of connected components ordered as per their appearance in the text image. Each component is described using its borders, i.e., (x_{min},x_{max}) and (y_{min},y_{max}). This can be straightforward computed by sorting the points of the connected component with respect to x-axis any y-axis respectively. A matrix, *component_border_matrix*, is maintained to store such information where each component is assigned an ID number according to its appearance and its borders information are retained (see Table 1) .The upcoming operations are based on this matrix.

Table 1.Comp_border_matrix

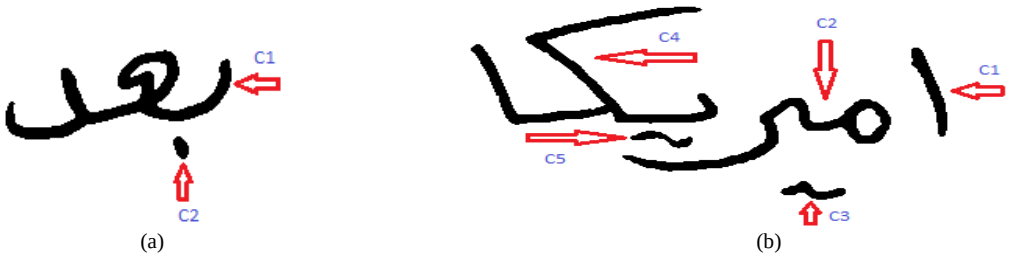
ID	x_{min}	x_{max}	y_{min}	y_{max}
1	2	8	5	10
2	2	3	7	7
3	8	10	3	7
4	6	6	2	4

C. Sub-word Segmentation

The core operation of segmenting text into sub-words is performed in this stage. It is inspired by the sequence of Arabic writing where sub-words are written -one after the other- from right to left. The sub-words segmentation task is accomplished by analyzing the components retained in the *comp_border_matrix*constructed in the previous stage. It may be recalled that an Arabic sub-word consists essentially of a main body (primary component). Sometimes, there exists diacritical marks (dots and/or Hamza) associated with the main body (they are known as secondary components). For example the first sub-word of the word given in Fig. 5(b) consists only of the main body (C1) whereas the second sub-word contains primary component (C2) and secondary component (C3). Initially, it is assumed that the position information, that is (x_{min},x_{max}) and (y_{min},y_{max}) of the primary component of the sub-word is sufficient to extract the sub-image containing such sub-word. However, as mentioned earlier, a sub-word may consist of both primary and secondary components. To handle such scenario the concept of satellite component is introduced.

Definition: Satellite Component

A component C_i is said to be a satellite component with respect to a component C_j if it is fully placed within the borders of C_j. As an example the, component C2 in Fig. 4(a) is a satellite component with respect to the component C1. The component C3 in Fig. 5(b) is a satellite component with respect to the component C2 whereas the component C5 in Fig. 5(b) is a satellite component with respect to both C4 and C2.



(a) (b)

Fig.5. Satellite Components.

The sub-word segmentation proceeds as follows: the rightmost component is considered to be the (primary component of) first sub-word in the text line. The satellite components associated with this component (if any) are identified. If there is no satellite component associated with this component, the information of the component under consideration are used to extract the sub-image contained between (xmin ,xmax) and (ymin, ymax). Otherwise, the information of satellite component (s) are used to modify (xmin ,xmax) ,if necessary, prior to the extraction of the sub-image containing the sub-word. If the sub-word is overlapping with the succeeding sub-word, a translation operation is performed to resolve the overlapping. This is repeated till all the sub-words are segmented. The detailed algorithm is listed below.

Algorithm: Sub-word Segmentation
Input: f, text line (word) image
comp_border_matrix, matrix containing components information
n, number of connected components
Output: SW1, SW2,…,SWk, Sub-words contained in the input text line
Method:
current_sub_word=1
while(current_sub-word<=n)
Obtain succeeding sub-word; i.e., successive component which are not a satellite component with respect to
current_sub_word
if succeeding sub-word is not overlapping with the current sub-word
Obtain satellite component (s) associated with the current_sub_word
if there exists no satellite component
Use the information contained in the *comp_border_matrix* to extract the sub-image containing the component
current_sub_word
else
Update (x_{min},x_{max}) using the positional information of satellite component (s)
else
Extract the sub-image (SI) surrendered by the rightmost column of the right sub-word (rc) and the leftmost
column of the left sub-word (lc)
Identify the satellite component (s) associated with the current_sub_word
$translation_amount = ymax(lc) - ymin($where:
ymax(lc): y_{max} of the left component, and
ymin(rc): y_{min} of the right component
Perform translation operation of the right component along with its satellite component (s), if any, by the
$translation\_amount$
Extract the sub-image of SI which contains the current sub-word along with satellite component (s), if any
current_sub_word=succeeding sub-word
Algorithm ends

For the sake of clarity, the proposed algorithm is explained in the light of the word shown in Fig. 6(a) which consists of 4 sub-words. The *comp_border_matrix* representing the components of the input word is shown in Table 2.

Table 2.Comp_border_matrixof the image given in Fig. 6(a)

ID	x_{min}	x_{max}	y_{min}	y_{max}
1	93	164	345	367
2	95	175	222	337
3	89	102	270	282
4	85	98	253	266
5	117	208	83	218
6	144	178	62	102
7	113	126	73	87
8	114	126	57	70

To start with, the first component (component 1 in the matrix) is considered to be (the primary component of) the first sub-word in the image. Based on the analysis of (y_{min},y_{max}) of the following component, it is deduced that there is no satellite component associated with this component; and hence the component which immediately follows (component 2) is considered to be (the primary component of) the succeeding sub- word. Based on the analysis of (y_{min},y_{max}) of the two components it is established that there is no overlapping. Since there is no satellite components associated with this component, the information (x_{min},x_{max}) and (y_{min},y_{max}) of component 1 are used to extract the sub-image; i.e., first sub-word in the image (See Fig. 6(b)). The algorithm proceeds to the (primary component of the) second sub-word. There exist two satellite components associated with component 2 and hence the component 5 is identified to be the (primary component of the) succeeding sub-word. The position information (x_{min},x_{max}) of the satellite components are used to modify (x_{min},x_{max}) of the sub-image containing the second word (Here only x_{min} is modified; see Fig. 6(c)). The component 5 is picked for processing. Here, there is no satellite components attached so the component 6 is identified to be the (primary component of the) succeeding sub- word. It is established that the (main body of the) current sub-word is overlapping with the (main body of the) successive sub-word. Therefore, the sub-image containing the current sub-word along with the succeeding sub-word is extracted (Fig. 6(d)). This is followed by applying a translation operation of the current sub-word with a suitable amount to resolve the overlapping (Fig. 6(e)). Finally, the sub-image containing the sub-word under consideration is extracted (Fig. 6(f)). Moving on to the next sub-word, it is identified that the component 6 constitutes the primary component of the sub-word. It is also found that the component 7 is a satellite component with respect to component 6, and hence the component 8 is considered to form the (primary component of the) succeeding sub-word. Based on the analysis of overlapping, process similar to the previous case is applied (Fig. 6(g)-(i)). Finally, the sub-word constituted by component 8 is extracted (Fig. 6 (j)).

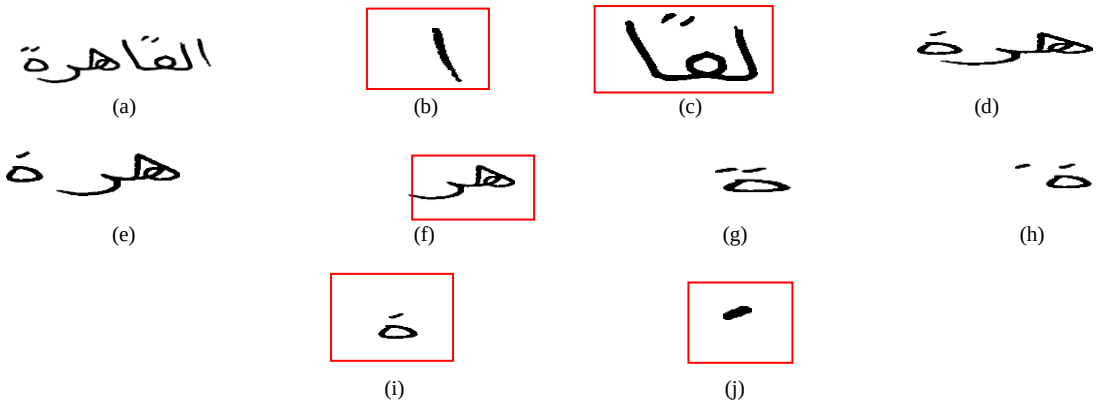


Fig.6. Stages of sub-word segmentation algorithm.

It is worth noting that the first three sub-words are segmented correctly whereas the last sub-word was over-segmented due to the displacement of the dot component which led to mis-identifying it as a separate sub- word. To address such cases, an additional stage which aims to refine the results of segmentation is developed. It is described in detail in the following sub-section.

D. Refinement of Segmentation

The results of the previous stage are subjected to refinement stage to compensate for the over-segmentation caused by the displacement of secondary components (i.e., dots and Hamza). The strategy applied for the refinement is: the height of the candidate sub-word is computed. If the height is less than a threshold and the sub-word overlaps with either the preceding or the succeeding sub-word, the sub-image containing the overlapped sub-word, along with its satellite components if any, and the sub-word under investigation is reconstructed using the information of pixels contained in the respective components. The detailed algorithm for reconstructing the sub-image containing a particular set of components in a binary image is listed below.

Algorithm: Reconstruct sub-image

Input: $C_1,C_2,\ldots,C_n$: Pixel coordinates of the components $C_1,C_2,$ and C_n respectively.

Output: SI, sub-image reconstructed.

Method:

Step 1: Find the minimum x-coordinate for each component mx_C_i, i=1,2,..,n

Step 2: Find the Maximum x-coordinate for each component Mx_C_i, i=1,2,..,n

Step 3: Find the minimum x-coordinate for each component my_C_i, i=1,2,..,n

Step 4: Find the minimum x-coordinate for each component My_C_i, i=1,2,..,n

Step 5: Set mx=min(mx_C_i)
Mx=min(Mx_C_i) i=1,2,..,n
my=min(my_C_i)
My=min(My_C_i)

Step 6: Declare SI to be an image of size (Mx-mx+1) rows and (My-my+1) columns containing only background (OFF) pixels

Step 7: For each component C_i , i=1 to n do
For each pixel of the coordinate (i,j) do
Map the pixel (i,j) to the output image (SI) pixel (i-mx+1,j-my+1)
Set the output image pixel to foreground (ON) pixel

Algorithm ends

The components C3={(9,7), (10,6), (10,5), (9,4), (8,4), (8,3)} and C4=={(6,4), (6,3), (6,2) of image given in Fig. 4 is reconstructed as follows- firstly, the (xmin,xmax) and (ymin,ymax) of each component is retrieved ((xmin,xmax) and (ymin,ymax) of C3 are (8,10) and (3,7) respectively. similarly, it is (6,6) and (2,4) for C4). Therefore, mx=6, Mx=10, my=2 and My=7 are set. Then an image of (10-6+1=5) rows and (7-2+1=6) are created with all the pixels are set to be background pixels. This is followed by mapping the pixels of C3 and C4 to the output image. Pixel p with the coordinates (i,j) is mapped to the output image as follows- (xout, ,yout) = (i-mx+1,j-my+1). For example, the pixel (9,7) of C3 is mapped to (9-6+1,7-2+1)=(4,6). Similarly, (6,2) of C1 is mapped to (1,1). Each pixel of a component Ci is mapped to the corresponding output coordinate and the corresponding pixel in the output image is set to 0 (foreground).

0	0	0	1	1	1
1	1	1	1	1	1
1	0	0	1	1	1
1	1	0	1	1	0
1	1	1	0	0	1

Fig.7. Reconstruction of the components C3 and C4 of the image given in Fig. 4

Going back to the previous example where the over-segmentation results is held, we take the previous example (Fig. 6) wherein the last sub-word is over-segmented into two sub-words (the first include the main body of the sub-word along with a dot whereas the second contains the dot). When the reconstruction is applied on the components of the fourth and the fifth sub-word, the over-segmentation is handled and the output is shown in Fig. 8.

Fig.8. Output of reconstruction operation.

IV. Experimental Results

Experiments were conducted on handwritten images (word/text line) drawn from four different datasets [10-13]. The sample consists of 350 images of Arabic/Persian text. Details are listed in Table 3. Datasets are briefly described below.

Datasets

The IESK-ArDB [10] database was developed in the Institute for Electronics, Signal Processing and Communication (IESK) at Otto-von- Guericke University Magdeburg. It contains more than 4000 Arabic word images in addition to more than 6000 segmented character images. Samples were collected from 22 writers from different Arabic countries. The IFN/ENIT [13] dataset was developed by the Institute of Communications Technology (IFN) at Technical University Braunschweing in Germany and The National School of Engineers of Tunis (ENIT). It contains handwritten Arabic names of 946 towns/villages in Tunisia (A name may consist of several words) written by 411 writers. Database contains totally 26459 handwritten images. KHATT [12] dataset consists of 1000 handwritten forms written by 1000 distinct writers from different countries Overall, each writer has written 6 paragraphs: 1 fixed text paragraph (written twice), 2 randomly selected paragraphs, and 2 optional paragraphs about subject of his interest. The fourth dataset used is a Persian dataset which is part of a multi-lingual dataset introduced in [11]. The Persian part consists of 140 unconstrained handwritten pages.

Initially, the experimentation was carried out without applying the refinement operation. Overall, we obtained 77.63% successful segmentation. Some successful segmentation results are shown in Fig. 9. The detailed results of the experimentation are given in table 4. The correct segmentation results for each dataset are shown in table 5.

Table 3. Experimentation Samples Details

Tot. No. of samples	No. of word image	No. of text line images	Tot. No. of words	Tot. No. of sub-words
350	212	138	988	2325

Table 4. Experimentation Results.

Segmentation Category		Percentage
Correct Segmentation		77.63%
Over-segmentation	Dot/Hamza Displacement	16.27%
	Pen lifting	2.75%
Under-segmentation	Fully overlapped sub-words	0.58%
	Touching sub-words	2%
Incorrect association of secondary component		0.77%

Table 5. Correct Segmentation results

Dataset	No. of Samples	No. of Sub-words	No. of correctly segmented sub-word	Percent
ArDB	150	356	320	89.88%
IFN/ ENIT	150	669	549	82.06%
KHATT	25	587	432	73.59%
Persian	25	713	504	70.69%

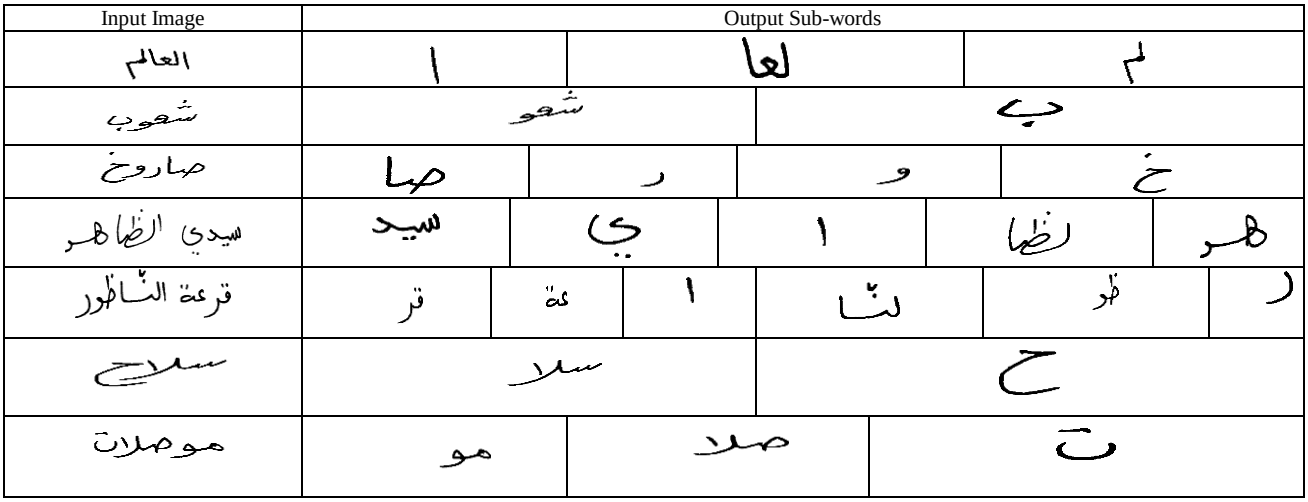


Fig.9. Some successful segmentation results.

There exist 19.02% of the sub-words which were over-segmented. The main reason for the over-segmentation is the displacement of dot/Hamza components (16.27%). Pen lifting has also caused over-segmentation (2.75%); however, in most of the cases (87%) where the pen lifting is the reason for over-segmentation, there will be no effect (consequences) on the subsequent character segmentation. Fig. 10 shows some text images wherein some sub-words were over-segmented (the over-segmented sub-words are indicated by rectangles).There exists also another segmentation error; that is the under-segmentation. The first (and inevitable) source for under-segmentation is the touching of the sub-words. Furthermore, there appear cases where a sub-word is totally overlapping with another sub-word leading it to be identified as a satellite component which causes under-segmentation. Fig. 11 shows some under-segmentation results (under-segmentation is highlighted by encircling).Finally, in few cases the algorithms was successful in segmenting the sub-word. However, it has incorrectly associated a secondary component with it. Fig. 12 shows such cases.

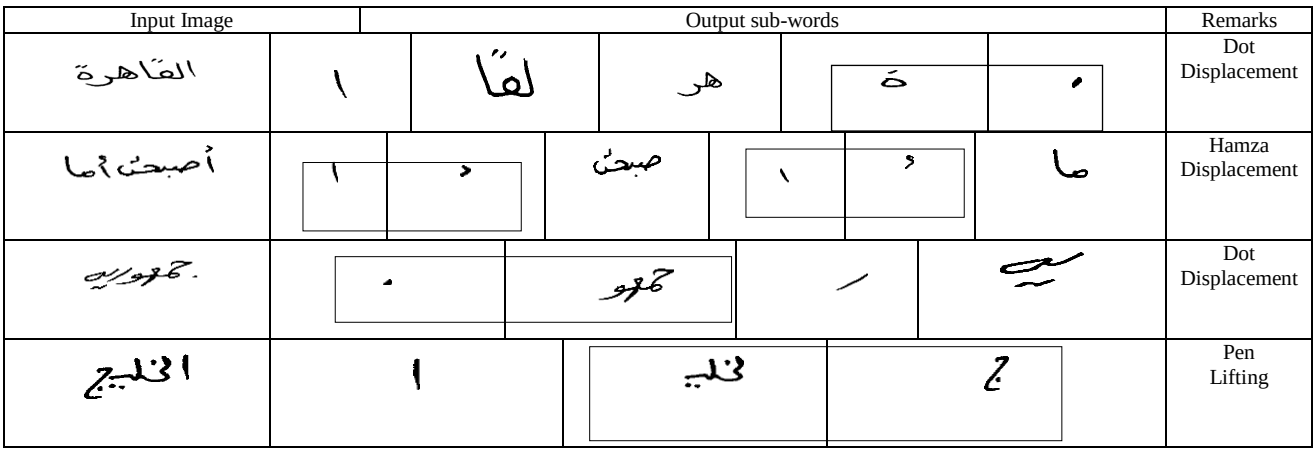


Fig.10. Some over-segmentation results.

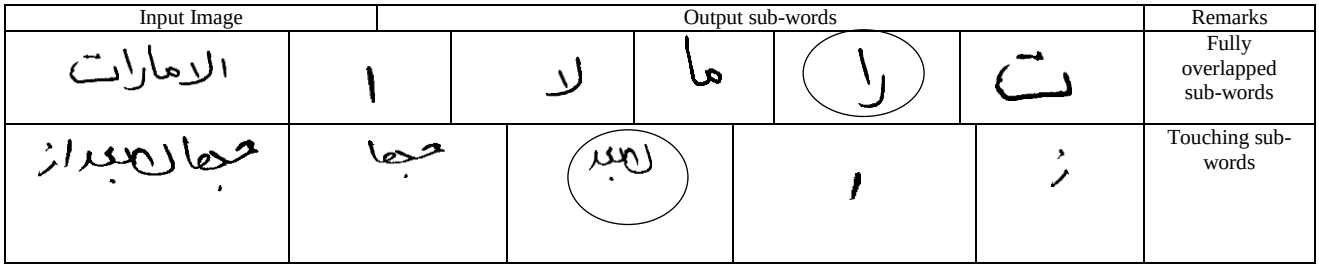


Fig.11. Some under-segmentation results.

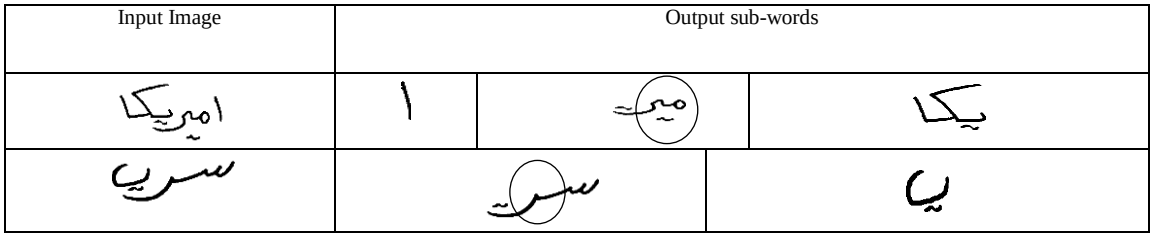


Fig.12. Incorrect Association of secondary components.

Table 6. Thresholding Experimentation

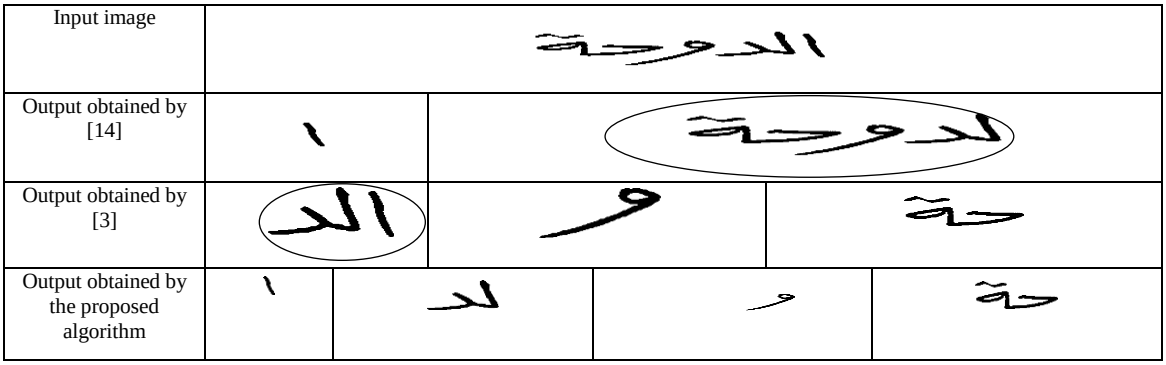
No. of secondary components	850	850	850	850	833
Threshold	50%	55%	**60%**	65%	70%
No. of secondary components detected	757	773	**811**	818	
Percentage	89%	91%	**95%**	96%	98%
No. of primary components categorized as secondary components	0	2	**4**	19	31

Table 7. Correct Segmentation results after the refinement

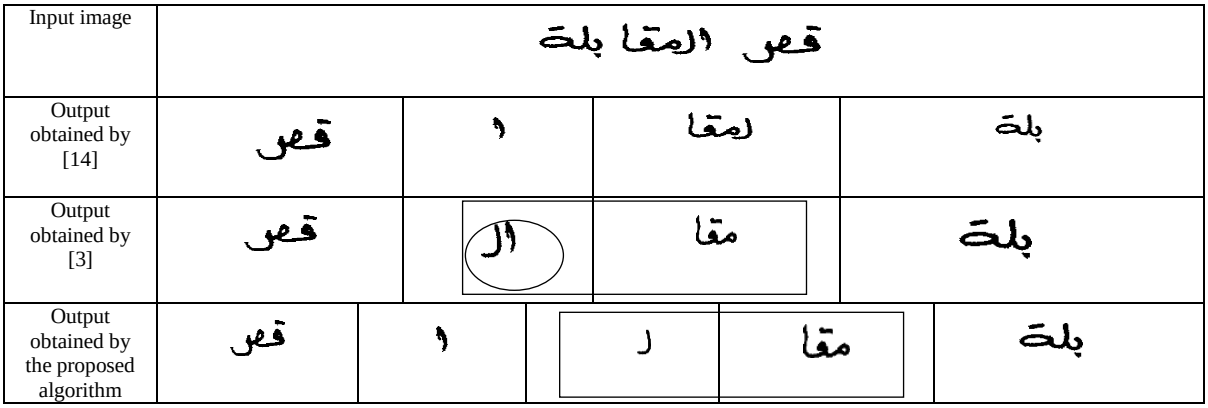
Dataset	No. of Samples	No. of Sub-words	No. of correctly segmented sub-word	Percent
ArDB	150	356	348	97.75%
IFN/ ENIT	150	669	632	94.47%
KHATT	25	587	523	89.10%
Persian	25	713	618	86.67%

Table 8. Comparative segmentation results

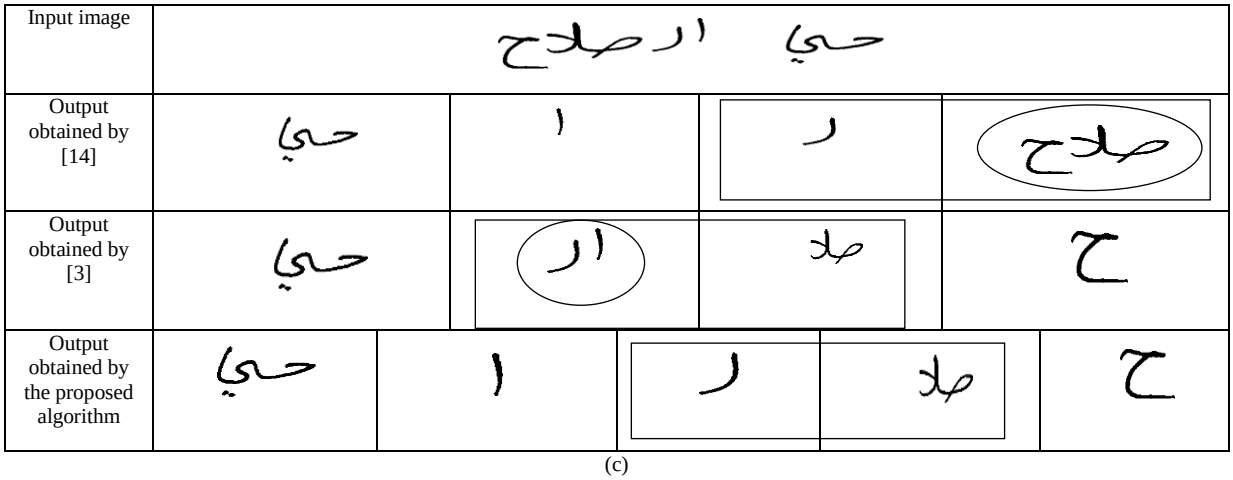
Method of [14]	Method of [3]	Proposed Method
74%	82%	91.23%



(a)



(b)



(c)

Fig.13. Comparative results; over-segmentation is highlighted by rectangle and under-segmentation by encircling.

From the above listing it is obvious that Dot/Hamza displacement is the major source of segmentation errors. This motivated us to add an additional stage which aims to handle such cases and refine the segmentation results. The detailed description of the refinement stage is stated above. It is observed that Dot/Hamza components have relatively small height. This is utilized in the stage of refinement. Towards this end, 120 handwritten images (77 word image and 43 text line image) were chosen. We have chosen the threshold as a percentage of the average height of the components in the image. The threshold of 60% was chosen and the experiments were re-conducted. The algorithm achieved 91.23% correct segmentation results after incorporating the refinement stage. The correct segmentation results for each dataset after the refinement stage is given in table 7.

An explicit experimentation with printed Arabic text

Finally, though the proposed algorithm is developed to segment handwritten Arabic text into sub-words, experimentation is carried out to test its efficiency when the input is a printed text. In this experiment 30 printed Arabic text lines is chosen from the database introduced in [19]. They are printed in 6 different styles and totally contain 1249 sub-words. The algorithm is successful in segmenting 99.44% of the sub-words where the only source of error is the incorrect association of satellite components. However, as stated earlier this could be handled in the subsequent stages of the recognition system. Barring this the algorithm is capable of segmenting printed Arabic text lines into sub-words almost perfectly.

V. Comparative Results

The method used for sub-word segmentation in [14] is based on vertical projection of the black pixels onto X-axis. It obtained 74% successful segmentation on the handwritten samples used in our experimentation (See table 3 for details). Although it can withstand the disconnectivity caused by pen lifting phenomena to some extent, it has the major limitation of being unable to segment overlapping sub-words. On the contrary, the proposed method which is based on connected component analysis efficiently handles the overlapping nature of Arabic text while facing the drawback of over-segmentation at the sub-word level. However, such over-segmentation has no consequences on the character segmentation in most of the cases as stated above. Finally, both the methods are unsuccessful in segmenting touching sub- words.

In another method proposed by Parvez et al [3], the connected components are firstly extracted. Then, each connected component is enclosed in a rectangular box and the area of such box is computed. These components which the area of their bounding box is greater than or equal to have of the average area are identified to be base components. Later on, the centroid points of such components are used to estimate the baseline. The area of the bounding box enclosing a particular component and the distance from its centroid point to the estimated baseline is used to determine if such component is a primary component or a secondary component. Finally, the secondary components are associated with their corresponding main components using the amount of overlapping; a secondary component is associated with the primary component with which it has the maximum amount of overlap. If a secondary component has the same amount of overlap with more than one component, it is associated with the component that has the smallest distance to the centroid point of the secondary component. Again, the experiment was conducted using the same handwritten samples and the algorithm successful rate of sub-word segmentation achieved is 82%. The main source of error for this algorithm is its dependence on categorizing the components into primary and secondary components; a primary component is quite frequently mis-identified as a secondary component due to the small area of the bounding box enclosing such component which leads to under-segmentation in many cases.

Sub-word segmentation is addressed as an intermediate stage of a system meant to recognize handwritten Arabic words in [9]. Firstly, the input word is segmented into

sub-words. Sub-words are segmented into graphemes which are later passed to a recognition engine that utilizes recurrent neural networks. The sub-word segmentation is accomplished as follows-the baseline is estimated using horizontal projection. Then, connected components are extracted. A component is considered to be a secondary body if it is very small compared with other components, it is relatively small and far from the baseline, or it is a vertical line with a large component below it. The components which are not secondary bodies are considered to be the main bodies of the sub-words. A set of rules are applied to assign secondary bodies to their respective main bodies. Finally, every main body is extracted with its secondary bodies as one sub-word and passed to the grapheme segmentation stage. The results of segmentation at sub-word level are not reported. Furthermore, the details of the fixing the parameters (size, distance) are not given. Hence it was not possible to subject such method for the experimental comparative analysis. However, it may be observed that the proposed method on the contrary does not require such parameters and is independent of the classification of components into primary and secondary.

VI. Conclusion

Segmenting Arabic text into sub-words is a crucial task for any recognition system. This paper is an attempt towards accomplishing such task. The proposed algorithm is based on the analysis of connected components (which are extracted using graph theory concepts). The results obtained are encouraging and it motivates us to do more research in this direction.

Acknowledgement

We would like to thank Miss. Faten Kallel Jaiem for providing the APTI printed Arabic database. The author also acknowledges University of Thamar, Yemen for financial support.

References

[1] R.M. Bozinovic, S.N. Srihari, “Off-line cursive script word recognition”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 11, No. 1, pp. 68-83,1989.

[2] S.N. Srihari, G. Ball, An assessment of Arabic handwriting recognition technology, in: V. Margner, H. El Abed (Eds.), Guide to OCR for Arabic Script, Springer-Verlag, London, pp. 3-34, 2012.

[3] M.T. Parvez, S.A. Mahmoud, “Arabic handwriting recognition using structural and syntactic pattern attributes”, Pattern Recognition, Vol. 46, No. 1, pp. 141-154, 2013.

[4] A.Cheung, M. Bennamoun, N.W. Bergmann, “An Arabic optical character recognition system using recognition-based segmentation”, Pattern Recognition, Vol. 34, No. 2, pp. 215-233, 2001.

[5] M. Zand, A.N Nilchiand, S.A. Monadjemi, “Recognition-based Segmentation in Persian Character Recognition”, World Academy of Science, Engineering and Technology, Vol. 38, pp. 183-187, 2008.

[6] A. AbdulKader, Two-Tier Approach for Arabic Offline Handwriting Recognition based on conditional joining rules, in: Proceedings of the 2006 Summit on Arabic and Chinese Handwritten Recognition, pp. 121-127, 2006.

[7] Y. Chherawala, M. Cheriet, “W-TSV: Weighted topological signature vector for lexicon reduction in handwritten Arabic documents”, Pattern Recognition, Vol. 45, No. 9, pp. 3277-3287, 2012.

[8] S. Wshah, V. Govindaraju, Y. Cheng, H. Li, “A Novel Lexicon Reduction Method for Arabic Handwriting Recognition”, in: Proceedings of the Twentieth International Conference on Pattern Recognition, pp. 2865-2868, 2010.

[9] G.A. Abandah, F. Jamour, E. Qaralleh, “ Recognizing handwritten Arabic words using grapheme segmentation and recurrent neural networks”, International Journal of Document Analysis and Recognition, Vol. 17, No. 3, pp. 275-291, 2014.

[10] M. Elzobi, A. Al-Hamadi , Z. Al Aghbari, L. Dings, “IESK-ArDB: a database for handwritten Arabic and an optimized topological segmentation approach, International Journal of Document Analysis and Recognition”, Vol. 16 No. 3, pp. 295-308, 2013.

[11] A. Alaei, P. Nagabhushan, U. Pal, “Dataset and Ground Truth for Handwritten Text in Four Different Scripts”, International Journal of Pattern Recognition and Artificial I, Vol. 26, No. 4, 2012.

[12] S.A. Mahmoud, I. Ahmad, W.G, “Al-Khatib, M. Alshayeb, KHATT:An open Arabic offline handwritten text database”, Pattern Recognition, Vol. 47, No. 3, pp. 1096-1112, 2014.

[13] M. Pechwitz, S. S. Maddouri, V. Märgner, N. Ellouze, H. Amiri, “IFN/ENIT- Database of Handwritten Arabic Words”, in: Proceedings of CIFED : colloque international francophone surl'écrit et le document, pp.129-136, 2002.

[14] A. Elnagar, R. Bentrcia, “A Multi-Agent Approach to Arabic Handwritten Text Segmentation”, Journal of Intelligent Learning Systems and Applications, Vol. 4, No. 3, pp. 207-215, 2012.

[15] A. Elnagar, R. Bentrcia, “A Recognition-Based Segmentation Approach to Segmenting Arabic Handwritten Text”, Journal of Intelligent Learning Systems and Applications, Vol. 7, No. 4, pp. 93-103, 2015.

[16] S. Marchand-Maillet, Y. M. Sharaih, Binary Digital Image Processing A Discrete Approach, first ed. Academic Press, London, 2000.

[17] R. C. Gonzalez, R. E. Woods, Digital Image Processing, third ed., Dorling Kindersley (India) Pvt. Ltd., India, 2009.

[18] S.N. Srihari, G. R. Ball, H. Srinivasan, “Versatile Search of Scanned Arabic Handwriting”, in: Proceedings of the 2006 Summit on Arabic and Chinese Handwritten Recognition, pp. 57-69, 2006.

[19] F. K. Jaiem, S. Kanoun, M. Khemakhem, H. El Abed, J. Kardoun, “Database for Arabic Printed Text Recognition Research”, in: Proceedings of the Seventeenth International Conference on Image Analysis and Processing, pp. 251-259, 2013.

[20] L. Zheng, A.H. Hassin, X. Tang, “A new algorithm for machine printed Arabic character segmentation”, Pattern Recognition Letters, Vol. 25, No. 15, pp. 1723-1729, 2004.

[21] A. Ebrahimi, E. Kabir, “A pictorial dictionary for printed Farsi subwords”, Pattern Recognition Letters, Vol. 29, No. 5, pp. 656-663, 2008.

[22] M. Khayyat, L. Lam, C.Y. Suen, “Learning-based word spotting system for Arabic handwritten documents”, Pattern Recognition, Vol. 47, No. 3, pp. 1021-1030, 2014.

Authors' Profiles

Hashem Ghaleb received his B.E. in Computer Engineering from King Saud University, Riyad, Saudi Arabia and M.Tech from University of Mysore, Mysore, India. Currently he is a PhD studentat the Department of Studies in Computer Science, University of Mysore, India. His research interest includes Image Processing, Document Image Processing, and Pattern Recognition

P. Nagabhushan (B.E.—1980, M.Tech.—1983, Ph.D.—1989) is a professor at the Department of Studies in Computer Science and was Director—Planning Monitoring and Evaluation Board at the University of Mysore, India. He is an active researcher in the areas pertaining to Pattern Recognition, Document Image Processing, Symbolic Data Analysis and Data Mining. He has over 400 publications in journals and conferences of International repute. He has chaired several international conferences. He is a visiting professor to USA, Japan and France. He is a fellow of Institution of Engineers and Institution of Telecommunication and Electronics Engineers, India.

Umapada Pal received his Ph.D. from Indian Statistical Institute (ISI). He did his Post-Doctoral research at INRIA, France. During July1997–January1998 he visited GSF- For schungszentrum fur Umweltund Gesundheit GmbH, Germany as a guest scientist. From January1997, he is a faculty member of the Computer Vision and Pattern Recognition Unit, ISI, Kolkata. He has published numerous research papers in various international journals, conference proceedings, and edited volumes. He received student best paper award from Chennai Chapter of Computer Society of India and a merit certificate from Indian Science Congress Association in 1995 and 1996, respectively. Dr. Pal achieved 'ICDAR Outstanding Young Researcher Award'fromTC-10 and TC-11 committees of IAPR in 2003. He has been serving as a guest editor, co-editor, program chair, and program committee member of many international journals and conferences. He is a life member of Indian unit of IAPR and a senior life member of Computer Society of India.

How to cite this paper: Hashem Ghaleb, P. Nagabhushan, Umapada Pal,"Graph Modeling based Segmentation of Handwritten Arabic Text into Constituent Sub-words", International Journal of Image, Graphics and Signal Processing(IJIGSP), Vol.8, No.12, pp.8-20, 2016.DOI: 10.5815/ijigsp.2016.12.02

