Contents lists available at ScienceDirect

Journal of King Saud University – Computer and Information Sciences

journal homepage: www.sciencedirect.com

An efficient, font independent word and character segmentation algorithm for printed Arabic text

Aziz Qaroush [a,*], Bassam Jaber [a], Khader Mohammad [a], Mahdi Washaha [a], Eman Maali [a], Nibal Nayef [b]

[a] Department of Electrical and Computer Engineering, Birzeit University, Palestine
[b] L3i, University of La Rochelle, France

ARTICLE INFO

Article history:
Received 20 June 2019
Revised 27 August 2019
Accepted 27 August 2019
Available online xxxx

Keywords:
Arabic OCR
Word segmentation
Character segmentation
Cursive script
Segmentation techniques
Baseline
Projection profile

ABSTRACT

Characters segmentation is a necessity and the most critical stage in Arabic OCR system. It has attracted the interest of a wide range of researchers. However, the nature of the Arabic cursive script poses extra challenges that need further investigation. Therefore, having a reliable and efficient Arabic OCR system that is independent of font variations is highly required. In this paper, an indirect, font-in dependent word and character segmentation algorithm for printed Arabic text investigated. The proposed algorithm takes a binary line image as an input and produces a set of binary images consisting of one character or ligature as an output. The segmentation performed at two levels: a word segmentation performed in the first level, by employing a vertical projection at the input line image along with using Interquartile Range (IQR) method to differentiate between word gaps and within word gaps. A projection profile method used as a second level of segmentation along with a set of statistical and topological features, which are font-independent, to identify the correct segmentation points from all potential points. The APTI dataset used to test the proposed algorithm with a variety of font type, size, and style. The algorithm experimented on 1800 lines (approximately 24,816 words) with an average accuracy of 97.7% for words segmentation and 97.51% for characters segmentation.

1. Introduction

Although the amount of information preserved and used in digital forms has gradually increased, the number of Arabic books, in addition to the historical documents without digital redundancies are massive. Regardless, the optical scanning preserves digital copies of such a document into a digital image form, mining the content of the image to information is impossible unless a subsequent transformation process into a digital text form accomplished Alginahi (2013). Moreover, the need for reliable optical character recognition system increased due to the existence of information-based application such as information retrieval system, search engine, editing old document system, exam correction system, and security identification (i.e. license plate recognition system).

Optical Character Recognition (OCR) is the process of transferring a printed or handwritten text image captured by a camera or scanner to an editable form, that can more efficiently be searched and stored. As a result, an increase in data usage and saves individuals and businesses time and money (Lawgali, 2015). In general, the OCR system consists of six main stages (Lawgali, 2015): image acquisition, preprocessing, segmentation, feature extraction, recognition, and post-processing. The OCR systems classified based on acquisition image to offline and online. In online OCR systems, the characters and words, which taken from a pen on a tablet or a smartphone, are recognized immediately once written. On the other hand, the input of offline OCR systems is usually a stored image taken by a camera, or a scanner, or any electronic device. Offline OCR further classified into two subcategories of handwritten and typed text (Islam et al., 2017). The Pre-processing step in the OCR system is a vital stage because the next stages will use the image more efficiently. This step aims to produce a cleanup version from the original input image. The step of Pre-processing includes several methods; these methods usually independent of font variations such as contrast enhancement, noise removal, binarization, skew correction, slant correction, and morphological operations like thinning. The segmentation stage consists of analyzing and dividing the input

* Corresponding author.
E-mail addresses: aqaroush@birzeit.edu (A. Qaroush), khamadawwad@birzeit.edu (K. Mohammad), emaali@birzeit.edu (E. Maali).

Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.jksuci.2019.08.013
1319-1578/© 2019 The Authors. Production and hosting by Elsevier B.V. on behalf of King Saud University.

image into basic units, depending on the script under the study and the segmentation methodology used. The units could be regions of interest, lines, words, sub-words, ligatures, characters, and strokes. The segments resulted from the partitioning stage, such as words and characters, processed in the feature extraction step to extract a set of statistical, structural, and topological or global transformation features. This information then passed to the classifier, which represents the recognition stage to identify the segments. Finally, a series of post-processing methods applied to enhance recognition accuracy by incorporating dictionaries, language models and spell-checks method.

The most critical stage in the developing cycle of an OCR system is segmentation because the output of this stage directly fed into the recognition engine (Islam et al., 2017; Lorigo and Govindaraju, 2006). Segmentation consists of four sequencing steps: page layout analysis, line segmentation, word segmentation and character segmentation. The regions of interest are identified and categorized in page layout analysis phase for the input image, followed by labeling the text regions then fed into line segmentation phase to extract the text lines using methods like horizontal projection. Finally, extracted lines partitioned into words or sub-words for direct recognition or further segmented into characters or ligatures as the last step.

According to character segmentation method, OCR systems classified into two different approaches: segmentation-free approach (holistic approach) and segmentation-based approach (analytical approach) as shown in Fig. 1 (Alginahi, 2013; Naz et al., 2016a; Zeki et al., 2011). In segmentation free approach (Sabbour and Shafait, 2013), since Arabic characters could overlap, slant and have different styles and fonts, the recognition performed without segmenting the words into its low-level segments like ligatures, characters, strokes, and diacritics. However, it uses some features, patterns, and look-up dictionary for a certain number of words. This approach usually used when targeting to recognize particular words like numbers and the names of cities'. The obvious problem with this approach is the number of classes present in the recognition stage, which results in performance degradation as the number of vocabulary increases. In contrast, the analytical approach segments each word into low-level segments like characters (Naz et al., 2014). More processing needed in this approach, however, fewer classes used for the recognition stage, which makes this approach more general and practical than the holistic approach for real-world problems.

Segmentation-based methods divided into implicit and explicit segmentation (Rehman et al., 2009). In implicit Segmentation or recognition-based segmentation, the word image is not divided into a small unit (e.g. characters), while characters recognized during the recognition. Such methods fall into two subcategories. Windowing methods, which based on using a sliding mobile window of

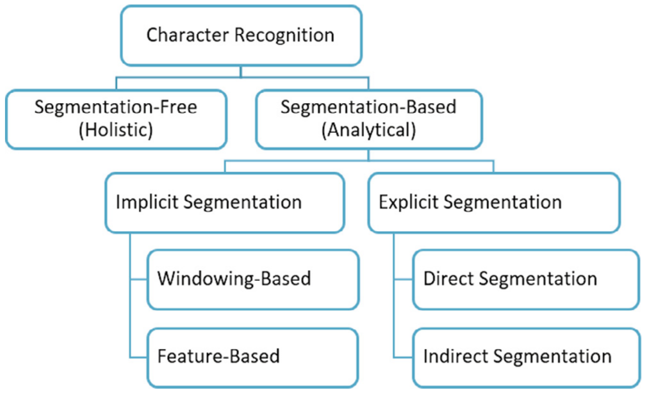

Fig. 1. Classification of character recognition methods.

variable width to provide the temporal segmentations without regarding the image features, and feature-based methods that based on detecting the physical location of image features, and then seek to segment this representation into well-classified subsets. Therefore, the former employs recognition to search for "hard" segmentation boundaries, the latter for "soft" segmentation boundaries (Casey and Lecolinet, 1996). In Contrast, in explicit segmentation, words segmented into independent units, such as ligatures, letters, or strokes. These methods classified into direct and indirect segmentation (Naz et al., 2016a; Zeki et al., 2011). In direct segmentation, a word image is divided directly into letters by employing some rules and heuristics. In contrast, indirect segmentation of a word separated into units that might be characters or part of a character called strokes (i.e dots, diacritics) (Elnagar and Harous, 2003). Then, these strokes merged by searching for certain features such as starting points, ending points, points of a sudden change in the contour, cusps, open curves, closed curves, and others. This approach has the advantage of minimizing the under-segmentation problem, but the cost of finding the optimum word from the merging of small units is expensive. Explicit-based segmentation methods were computationally complex but yielded slightly better results than less complex implicit-based segmentation methods (Rehman et al., 2009).

The character segmentation performance is highly dependent on the nature of language. In the Arabic language, the complexity of identifying and finding the correct segmentation point increases by the cursive nature of the script. Indeed, the hardest, the most crucial and the most time-consuming step of any OCR system is the character segmentation. Moreover, the Arabic language has a set of unique features that made the character segmentation task more challenging, and thus the research in the field of Arabic OCR moves slowly. The features of the Arabic script includes the following characteristics and uniqueness as depicted in Fig. 2 (Ahmed and Al-Ohali, 2000; Zeki and Zakaria, 2004; Mahmood, 2013): (•) The availability of different font types makes the shape and the contour of characters irregular and diverse (Fig. 2a); (•) The existence of diacritics called "Harakat" increases overlapping between adjacent characters (Fig. 2b); (•) Most of the characters in Arabic have four forms according to the function of their position in the world (at the beginning, middle, end, or separated) (Fig. 2c), (•) Arabic characters do not have a fixed size in terms of height and width (Fig. 2d), (•) In some font types such as Thuluth and Naskh connected characters may be combined as a new shape called Ligature, which is not similar to the result of concatenating the two basic shapes horizontally (Fig. 2e), (•) The length of connection strokes vary in different fonts (Fig. 2f). (•) Several Arabic characters can overlap with the next character in the same or adjacent word, (Fig. 2g) and (•) In some fonts or writing styles, the strokes of some characters like – SEEN character- are omitted to give them a non-standard shape (Fig. 2h).

Several methods presented in the literature to address the problem of segmenting Arabic characters, however, some of these methods are font (type, size, and style) dependent, and cannot handle different fonts, which have overlapping characters or ligatures such as "Diwani", "Andalus", and "Thuluth" which are widely used in old books and documents. Besides, some of these methods depend on predefined parameters (i.e threshold) such as the projection value to find the segmentation points (Zheng et al., 2004; Shaikh et al., 2009; Marwa Amara and Zidi, 2016; Anwar and Adiwijaya, 2015; Mousa et al., 2017). Moreover, the essential goal of these methods is to find the correct segmentation points without optimizing the shape of the extracted characters, which results in missing some important shape information especially when the characters overlapped.

This paper presents an indirect segmentation-based algorithm for printed Arabic text. The presented algorithm takes a text-line

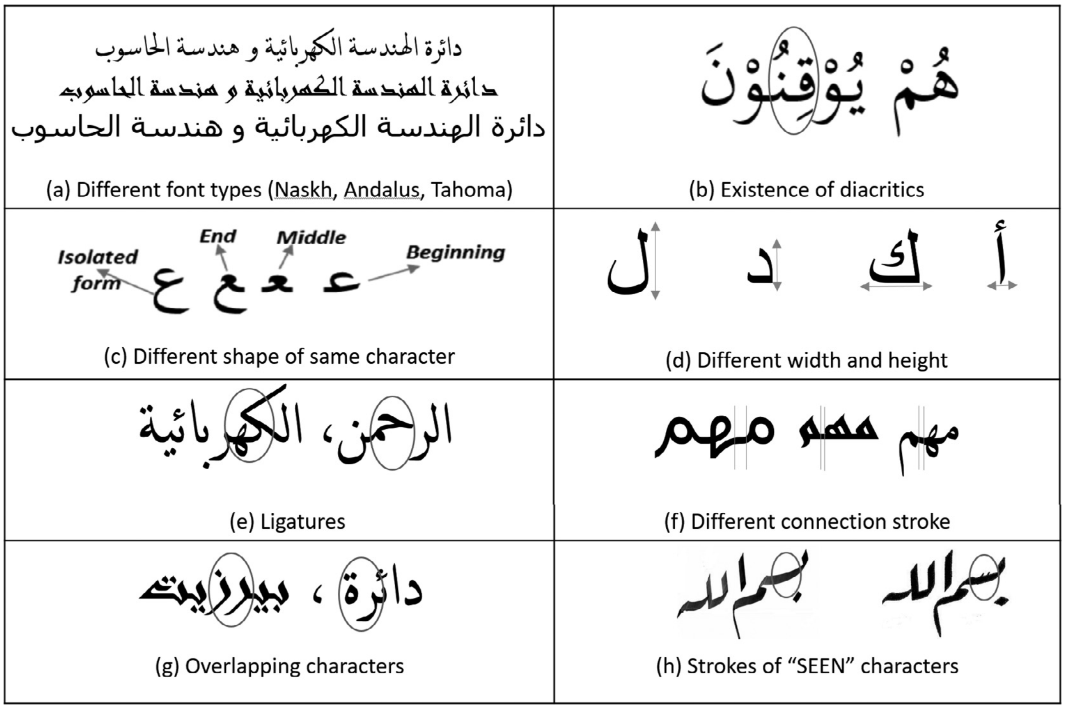
دائرة الهندسة الكهربائية و هندسة الحاسوب
(a) Different font types (Naskh, Andalus, Tahoma)
(b) Existence of diacritics
Isolated form
End
Middle
Beginning
(c) Different shape of same character
(d) Different width and height
(e) Ligatures
(f) Different connection stroke
(g) Overlapping characters
(h) Strokes of "SEEN" characters

Fig. 2. Arabic text properties.

image as an input and consists of two main stages: word and character segmentation. For word segmentation, the proposed algorithm employs a projection profile method along with using the Interquartile Range (IQR) statistical method to differentiate between word spaces and sub-word spaces (Han et al., 2012). Meanwhile, the proposed character segmentation algorithm employs a projection profile method along with using a set of statistical and topological features which are invariant to font variations, to identify the correct segmentation points from all potential segmentation points. Hence, the main contributions of this paper are as follows: First, we present a simple, efficient, and font-independent word segmentation method. Second, we present an indirect character segmentation approach that has the following characteristics: (i) the presented method is font-independent and can handle simple and complex font types, (ii) solves the problem of overlapping between adjacent characters and between sub-words, (iii) optimizes the location of the segmentation points to maintains character shape, and (iv) reduces the number of ligatures as much as possible. Third, in the evaluation stage, experimental results on the APTI dataset prove that our method achieves better performance than the state-of-the-art methods.

This paper organized as follows: Section 2 describes the state of art-related work. Section 3 introduces the proposed method. Sections 4 describes the dataset used in the experiments then introduce the results with a comprehensive comparison with other related methods. Finally, Section 5 presents our conclusion and future work.

2. Related work

2.1. Explicit segmentation

Many methods proposed for Explicit Segmentation of Arabic OCR character (Lawgali, 2015; Alginahi, 2013; Naz et al., 2016a; Zeki et al., 2011; Casey and Lecolinet, 1996). These methods are classified into: (i) projection profile; (ii) contour tracing, (iii) morphological operations, and (iv) template matching methods.

2.1.1. Projection profile methods

Projection profile methods (Zheng et al., 2004; Shaikh et al., 2009; Marwa Amara and Zidi, 2016; Anwar and Adiwijaya, 2015; Mousa et al., 2017) are commonly used method for lines, words and characters segmentation. Indeed, the horizontal projection profile used for lines segmentation, while the vertical projection profile usually used for words and characters segmentation. Projection profile methods are computationally simple and achieve good results for simple font types. However, using vertical projection alone for cursive text is directly prone to over-segmentation when the existence of characters that are composed of several parts or for under-segmentation when the overlapping between adjacent characters exists.

Zheng et al. (2004), proposed a segmentation algorithm of machine-printed Arabic character that employs a vertical projection method with some rules. These rules based on the four kinds of features, which are independent of the text size and easily computed. Shaikh et al. (2009) suggested an algorithm for Sindhi text segmentation, which is an Arabic style scripting language, using Height Profile Vector (HPV). The algorithm starts by finding the HPV of the primary stroke of a sub-word, then the HPV analyzed to determine the locations of the Possible Segmentation Points (PSPs). Marwa Amara and Zidi (2016) developed a segmentation method that is based on histogram projection along with some contextual topographies properties of Arabic writing. The potential segmentation points found uses vertical projection. These points are then filtered to find the correct segmentation points based on some structural properties of the Arabic language and their positions regarding baselines. Anwar and Adiwijaya (2015) proposed a segmentation method of Arabic character with "Harakat". Firstly, the image converted into a morphed form, then the locations where the projection profile value equal exactly two pixels are identified. Therefore, the image split in the middle of the occurrences of such locations more than three successive rows. The process repeated until the whole subword/word segmented. Mousa et al. (2017) employed a profile's amplitude filter to find the separation between two connected characters which considered as a constant amplitude in the profile. In addition to that, he used a simple edge method to determine whether it is a correct character's connection or not.

2.1.2. Contour tracing methods

In contour-tracing methods (Omidyeganeh et al., 2005; Bushofa and Spann, 1997a; Mehran et al., 2005), the pixels that form the outer shape of the word, sub-word, or character are traced and then extracted. Contour-based methods provide a clear description of the shape of the characters which can solve the under segmentation problem caused by characters overlapping (Alginahi, 2013). Besides, it reduces the errors generated when extracting baselines since there is no need to adjust the baselines many times. However, this kind of segmentation suffers from over-segmentation and also is sensitive to the existence of noise and characters brakes, which needs to do image enhancements.

Bushofa and Spann (1997a) proposed a segmentation algorithm based on the contour of the main body of the words. The algorithm starts by finding the start and the end-point of the upper contour of the word/sub-word. Then, the upper contour segmented into parts through the curvature of the same sign. To eliminate noise sensitivity, Bushofa applies a low pass filter on the contour points. Mehran et al. (2005) investigate the segmentation and recognition of Persian/Arabic scripts. They employed three basic features including vertical projection of the line image, the first derivative of the upper contour, and the distance of the tip of the pen from the baseline to identify the junction points of the upper contour of the primary stroke of sub-words called PAWs. Omidyeganeh et al. (2005) presented a new segmentation algorithm for multi-font Farsi/Arabic texts. The algorithm based on conditional labeling for up and down contours and the contour of sub-word measured by using a convolution kernel with Laplacian edge detection method. The algorithm goes through several steps including contour labeling of each sub-word, contour curvature grouping to improve the segmentation results, character segmentation, adaptive local baseline detection, and post-processing. Mazen Bahashwan and Sheikh (2017) employed a contour-based detector method to detect the corners as candidate segmentation points (e.g. branch points, ross points, and corner points). Then, they removed incorrect segmentation points by using a set of heuristic rules. Sari and Sellami (2005) developed a segmentation method to split up the isolated handwritten words into perfectly separated characters based on topological rules which constructed at the feature extraction phase.

2.1.3. Morphological and thinning methods

Morphological operations include a set of methods to extract image components, which used in the representation and description of region shape, such as boundaries, and skeletons. Most of the Arabic characters connected to baseline, therefor, Morphological segmentation allows us to break words into smaller units by applying morphological operation such as closing and opening (Alfonse et al., 2010). Besides, the skeleton of the words generated by morphological thinning method provides essential information about character shape, which simplifies the text shape and thus reducing the amount of data to handle. However, In many cases, the shape of characters after applying thinning operation differs from the original one, making the segmentation process more difficult. Also, it will not get sufficient results if it is not supported or combined with other techniques.

Timsari (1996) employed a hit-or-miss morphological method to segment words into its characters where the word described in terms of predefined patterns. The approach searches the database which is holding the description of all characters, for possible matches of characters using the hit-or-miss method. Cowell (2001) used an iterative based thinning method to generate the skeleton of isolated Arabic characters. Also, the authors discussed the problems of thinning Arabic characters for poor-quality images. Besides, he used a post-processing method to enhance the generated skeleton. Fitriyatul Qomariyah and Mahmudy (2017) proposed a segmentation method using an interesting point based on a set of rules to separate the connected Arabic character. The interest points were used as the coordinate reference to split each character.

2.1.4. Template matching methods

Template matching (Elnagar and Al-Kharousi, 1997; Margner, 1992; Saabni, 2014) is usually used to find a small part of an image that matches a predefined template image. In character segmentation, it usually applies a sliding window over the baseline and searches for a match with the characters or segments that are chosen manually and stored to use for comparison. As a result, when matching noticed, then the center pixel in the sliding window is considered as a cutting point. A major limitation of this method is that the performance decreases as the number of predefined segments increases when using more font types and styles. Besides, this approach takes more time to check all templates of the predefined segments, especially with many font styles and sizes. Moreover, the existence of noise in the input image sufficiently decreases the performance.

Margner (1992) proposed a segmentation method which searches for occurrences of an angle formed by the joining of two characters at the baseline. The method starts by finding the location of the baseline, then scanning the baseline from right to left using a $7 * 7$ window to find the candidate segmentation points. Therefore, the central pixel of the window is chosen to be a candidate point if the current window matches the pattern. Besides, the author employed some rules to avoid having wrong segmentation points such as avoiding segmenting inside a hole. Saabni (2014), proposed character segmentation approach that uses a partial segmentation method and Hausdorff distance. The algorithm takes a word/subword as input and starts calculating the size and the font type using Stroke Width Transform (SWT) method to define a set of multi-size sliding windows to search and identify characters within a given shape of a word/subword. The authors employed a novel method which uses Hausdorff distance to measure the similarity between character and sliding window image taking into account different sizes and locations of sliding windows.

2.2. Implicit segmentation

In implicit segmentation, there is no need for an accurate character segmentation points. It based on searching the image for components that match classes in its alphabet. Radwan and Khalil (2016) proposed a character segmentation approach based on the multichannel neural network. The system recognizes the features of segmentation window to predict the likelihood of the current window to a segmentation area. To increase the network input context, the authors employed another two windows as an input to a multichannel neural network one as a previous window with respect to the current window and the other as a next window. In Rosenberg (2012), the authors employed a local feature extracted by SIFT algorithm for character classification. Each word scanned with increasing window sizes; thus, segmentation points set where the classifier achieves maximal confidence. Bushofa and Spann (1997b) used a combination of a heuristic algorithm and a neural network-based technique to identify incorrect segmentation points by employing a set of structural features. Naz et al. (2016b) proposed implicit segmentation of printed Urdu text-lines written in the Nasta'liq writing style using Multi-dimensional Long Short-Term Memory (MDLSTM) Recurrent Neural Networks with an output layer designed for sequence labeling for recognition.

Gouda and Rashwan (2004) proposed a segmentation method for printed Arabic character using Hidden Markov Models HMMs. In this method, after removing the secondary stroke from the word, each line scanned from right to left using a sliding window. Then, a set of features extracted from the windows then fed into the HMM models to predict the character in the current window where M different numbers of HMMs constructed for each character or ligature. Al-Muhtaseb et al. (2008) proposed a technique for automatic recognition of off-line printed Arabic text using HMMs. The authors employed a variable size of overlapping and non-overlapping hierarchical windows to extract a set of simple and effective features from each vertical sliding strip. Radwan and Khalil (2016) proposed a character segmentation approach based on the multichannel neural network. The system recognizes the features of the segmentation window to predict the likelihood of the current window to a segmentation area. To increase the network input context, the authors employed another two windows as an input to a multichannel neural network one as a previous window concerning the current window and the other as a next window. In Rosenberg (2012), the authors employed a local feature extracted by the SIFT algorithm for character classification. Each word scanned with increasing window sizes; thus, segmentation points set where the classifier achieves maximal confidence. Bushofa and Spann (1997b) used a combination of a heuristic algorithm and a neural network-based technique to identify incorrect segmentation points by employing a set of structural features. Naz et al. (2016b) proposed implicit segmentation of printed Urdu text-lines written in the Nasta'liq writing style using Multi-dimensional Long Short-Term Memory (MDLSTM) Recurrent Neural Networks with an output layer designed for sequence labeling for recognition. Gouda and Rashwan (2004) proposed a segmentation method for printed Arabic character using HMMs. In this method, after removing the second stroke from the word, each line scanned from right to left using a sliding window. Then, a set of features extracted from the windows fed into the HMM models to predict the character in the current window where M different numbers of HMMs constructed for each character or ligature. Al-Muhtaseb et al. (2008) proposed a technique for automatic recognition of off-line printed Arabic text using HMMs. The authors employed a variable size of overlapping and non-overlapping hierarchical windows to extract a set of simple and effective features from each vertical sliding strip.

3. Proposed work

In this section, an indirect, character segmentation-based approach for printed Arabic text is presented. The proposed approach takes a binary line image as an input and produces an output of a set of binary images consisting of one character or ligature of the input line image. In our algorithm, the segmentation performed at two levels: word segmentation and character segmentation. Word detection performed by employing a vertical projection of the word image along with using Interquartile Range (IQR) while Character segmentation a baseline dependent method and is performed by using a projection profile method along with a set of statistical and topological features to identify the right segmentation points from a set of all potential segmentation points.

3.1. Word segmentation

Word Segmentation is the process of converting lines of text into separated words. Each word then recognized directly as in segmentation-free approach or further segmented into characters as in segmentation-based approach. Word segmentation is usually used as a previous stage of the character segmentation and also employed as a post-processing method to enhance the recognition accuracy by incorporating language dictionaries and spell-checks method. Word segmentation in the literature review is mainly based on the analysis of the geometric relationship of adjacent characters and addresses two main aspects including (i) the way the distance of adjacent characters calculated, and (ii) classifying the previously calculated distances as either between word-gaps or within word-gaps (Louloudis et al., 2009).

Word segmentation in Arabic depends primarily on the separation distance between words and sub-words and can be used to separate lines of text into words by cutting vertically where the vertical projection equals zero. However, the resulted gaps could be either word-gaps or within word-gaps. Therefore, to classify these gaps independently of the font type, size, and style, three hypotheses regarding Arabic script must be taken into consideration: (i) the length of word-gaps is fixed for single text line and depends on the font type and on the text alignment, (ii) the length of the within word-gaps is variable for one font type and for different font types, which depend on the location of the space and the surrounding letters, and (iii) the length of word-gaps is larger than the length within word-gap. According to these hypotheses, most of the Arabic word segmentation methods are based on a predefined threshold value which depends on the font variations. Fig. 3 shows the vertical projection of Arabic text line written in three different fonts: Naskh, Traditional Arabic, and Andalus respectively, where vertical projection zero-valued indicates a potential candidate for word separation. It observed that regardless of the font type, the gap between words is enormously noticeable, fixed in the same line, and larger than spaces between subwords.

The suggested word segmentation method is font invariant and computationally simple. The method consists of three steps as shown in Algorithm 1: (i) Identifying all gaps and calculate their length, (ii) Classifying each gap as a word gap or within word gap, and (iii) Extracting word images from the line input image used by character segmentation and recognition stage. In the first step, all gap spaces inside the line are found by calculating the vertical projection of the input line image. Vertical projection can be easily computed by finding the number of pixels having binary value one for each bin in the vertical direction formulated as:

$$V[j] = \sum_{i=0}^{n-1} BinaryImage[i,j] \qquad (1)$$

Where $V(i)$ is the horizontal projection of the image for column j, and the $BinaryImage[i,j]$ is the pixel value at $[i,j]$. The zero value inside the vertical projection indicates the locations of all spaces that could be a potential word separation. The length of all spaces measured as shown in Algorithm 2, where consecutive pixels having zero projection value considered as one gap.

Algorithm 1 Word segmentation

```
1: INPUT: L as a binary line image
2: SET G as List of gaps
3: SET L as List of gaps length
4: SET Words as a list of segmented words
5: $[G, L] \leftarrow GapLocationAndTheirLength(LineImage)$
6: $[G, L] \leftarrow gapFiltration(G, L)$
7: SET $i \leftarrow 1$
8: while $i < length(G)$ do
9:     $wordStartIndex \leftarrow G[i]$
10:    $wordEndIndex \leftarrow G[i+1]$
11:    $Words.add(L.extract(wordStartIndex, wordEndIndex))$
12: end while
13: OUTPUT: Words
```

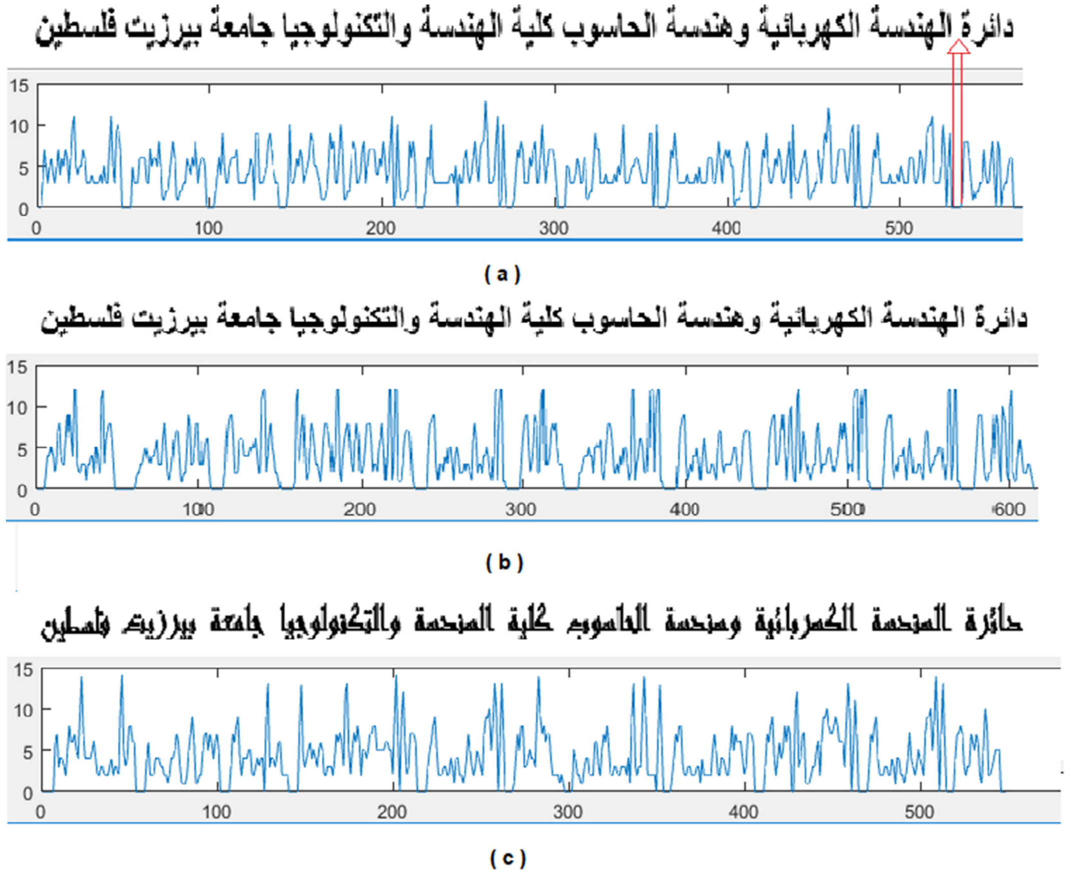

Fig. 3. Text Line written in three font types with their vertical projection, (a) Naskh, (b) Traditional Arabic, and (c) Andalus.

Algorithm 2 Finding gap space location

```
1: INPUT: Line
2: SET G as a list of gaps
3: SET L as a list of gaps length
4: VP ← verticalProjection(Line)
5: SET $flag \leftarrow 0$
6: SET $i \leftarrow 1$
7: while $i < length(VP)$ do
8:   if $(VP[i] == 0$ && $flag == 0)$ then
9:     $G.add(i)$
10:      $L.add(G[i] - G[i-1])$
11:      $flag \leftarrow 1$
12:    else if $(VP[i]! = 0$ && $flag == 1)$ then
13:      $flag \leftarrow 0$
14:    end if
15: end while
16: OUTPUT {G, L}
```

To differentiate between word and sub-word gap spaces, some further filtrations are required, taking into consideration the hypothesis mentioned earlier to be font independent. The proposed filtration method is performed at two levels as shown in Algorithm 3. In the first level, all gaps with a length less than Interquartile Range (IQR) value of the list of gap space values are obtained from the previous step considered as a sub-word space and then removed from the list. Interquartile Range (IQR) is a statistical test method, used to identify and penalize outliers. IQR is preferable in this situation because the spaces between words are identical within a margin of **1px** difference and the spaces inside the words differ depending on the surrounding letters. This step helps to remove very small gaps but does not fully differentiate between the word gap and sub-word gap. In the second level, since all small gaps were removed and the word-space is larger than sub-word space, the mean of the filtered list of gaps will be greater than sub-word gaps. Thus, all gaps with a length less than the mean of the filtered list of gaps removed. Finally, each word image is extracted from the input line image using the word gap.

Algorithm 3 Gap length filtration

```
1: INPUT: {G, L}
2: SET $iqrValue \leftarrow IQR(L)$
3: SET $i \leftarrow 1$
4: while $i < length(L)$ do
5:   if $(L[i]! = 0$&& $L[i] < iqrValue)$ then
6:     $G.remove(i)$
7:     $L.remove(i)$
8:   end if
9: end while
10: SET $meanValue \leftarrow mean(L)$;
11: SET $i \leftarrow 1$
12: while $i < length(L)$ do
13:   $L[i] < meanValue$
14:     $G.remove(i)$
15:   end if
16: end while
17: OUTPUT {G, L}
```

Table 1 shows the gap length values along with their IQR value of the image text lines shown in Fig. 3. After applying first level filtration, all gaps with a length less than IQR value (i.e. less than five in Naskh as shown in Table 1) removed. The next step to finding the mean value of the length of the remaining gaps and then, the gap length which is less than mean value (less than five in Naskh, less than five in Traditional, and less than eight in Andalus as shown in Table 1) removed. Thus, values larger than the mean considered as a gap word separation. Fig. 4 shows the output after applying the Word segmentation algorithm on the text line images of Fig. 3.

3.2. Character segmentation

The proposed algorithm provides an indirect segmentation-based method to handle most of the challenges caused by the cursive nature of Arabic script, including the existence of different shapes of the same character in addition to character overlapping problem. The algorithm which is baseline dependent method employs a projection profile method to find all potential segmentation points. Besides, it uses a set of rules to find the correct segmentation points with the optimal separation of connected adjacent characters. The algorithm generalizes to ensure working for several font variations. The proposed character segmentation algorithm consists of four sequencing stages as shown in Fig. 5. The algorithm takes an input of a binary word image along with its binary line image and produces a set of binary images of one character or ligature. The character segmentation algorithm is performed from word level and uses line-level because the algorithm calculates several measures to improve segmentation results such as baseline index, which computed more accurate from the line level.

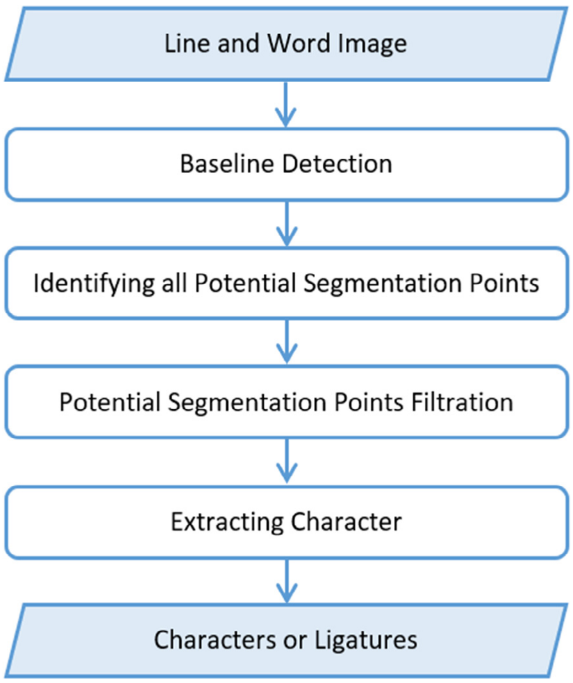
Line and Word Image
Baseline Detection
Identifying all Potential Segmentation Points
Potential Segmentation Points Filtration
Extracting Character
Characters or Ligatures

Fig. 5. Character segmentation main stages.

3.2.1. Baseline detection

The baseline considered as one of the distinctive features of Arabic script. It defined as the line with the most pixel density along with the whole line where most letters on a horizontal segment with a constant width. Usually, Arabic characters connected through baseline. Therefore, identifying the location of the baseline is important for determining potential segmentation points, skew normalization and feature extraction (AL-Shatnawi and Omar, 2008). The suggested baseline detection method begins with applying the morphological thinning operation on the input line image based on the method proposed in Deng et al. (2000) to identify the baseline more accurately, and also to create sharp peaks for

Table 1
Gaps length values for the image text lines of Fig. 3.

Font Type	Gaps Length																					IQR Value
Naskh	7	6	6	1	6	1	6	1	5	5	1	5	6	2	1	6	1	5	4	•	•	**5**
Traditional Arabic	3	6	5	5	1	5	1	1	5	5	1	6	1	5	1	5	1	6	1	1	6	**4**
Andalus	5	13	9	9	9	9	9	9	9	9	9	2	•	•	•	•	•	•	•	•	•	**0**
	Remaining Gaps After Level One Filtration using IQR																					Mean
Naskh	7	6	6		6		6		5	5		5	6			6		5	4	•	•	**5**
Traditional Arabic		6	5	5		5			5	5		6		5		5		6			6	**5**
Andalus	5	13	9	9	9	9	9	9	9	9	9	2	•	•	•	•	•	•	•	•	•	**8**
	Remaining Gaps After Level Two Filtration (Word Gap Length)																					
Naskh	7	6	6		6		6		5	5		5	6			6		5		•	•	•
Traditional Arabic		6	5	5		5			5	5		6		5		5		6			6	•
Andalus		13	9	9	9	9	9	9	9	9	9		•	•	•	•	•	•	•	•	•	•
hline																						

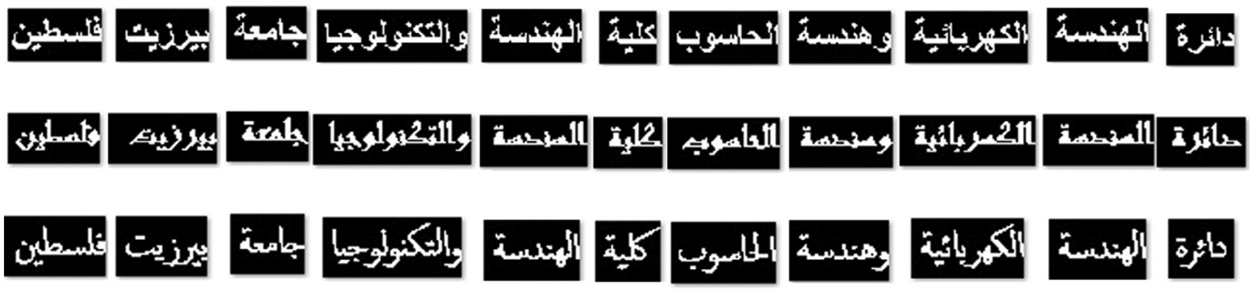

Fig. 4. Output after word segmentation of Fig. 3.

the horizontal projection inside the line itself. Then, a Horizontal Projection (HP) method applied to the thinned line image followed by a smoothing operation to reduce the number of peaks. The horizontal projection method reduces the 2D of data to 1D based on the pixels of the text image. The horizontal projection profile defined as:

$$H[i] = \sum_{i=0}^{n-1} BinaryImage[i,j] \tag{2}$$

where $H(i)$ the horizontal projection of the image for row is i, and the $BinaryImage[i,j]$ is the pixel value at $[i,j]$. Finally, the maximum peak after applying these sequential operations found which represents the index of the baseline. Algorithm 4 shows the procedure of detecting baseline for Arabic text line, where the algorithm takes a binary line image as an input and returns the index of the baseline as an output. Fig. 6 shows the location of the detected baseline (red lines) of the text lines image in Fig. 3.

Algorithm 4 Baseline detection

```
1: **Input**: Line
2: **SET** HP as an empty List
3: **SET** PV as an empty List
4: $thinedLine \leftarrow imageThinning(Line)$;
5: $HP \leftarrow horizontalProjection(thinedLine)$;
6: $HP \leftarrow smoothing(HP)$;
7: $PV \leftarrow findPeakValues(HP)$;
8: **SET** $BaseLineIndex \leftarrow 0$
9: **SET** $MAX \leftarrow 0$
10: **SET** $i \leftarrow 1$
11: **while** $i < length(PV)$ **do**
12:     **if** $PV[i] > MAX$ **then**
13:         $MAX \leftarrow PV[i]$
14:         $BaseLineIndex \leftarrow index(i)$
15:     **end if**
16: **end while**
17: **OUTPUT**: BaseLineIndex
```

3.2.2. Potential segmentation points identification

The most challenging task in the segmentation of cursive script language such as Arabic script is to find and identify the right segmentation points. These points will be used to determine the location of separation or the borderline between consecutive connected characters. The proposed approach to identify these points is an indirect-segmentation method consisting of two main steps where the task of the first step is to find all potential segmentation points, while the purpose of the second step is to reduce the over-segmentation problem.

The shape of the Arabic characters varies and depends on the location of the character and font type. However, in Arabic script, most of the Arabic characters connected through baseline, where a vertical transition above the baseline is usually an indication of a new character. This hypothesis is font independent. Therefore, we define the separation region between two consecutive connected characters as the baseline area between two consecutive vertical transitions and the segmentation point usually located around the middle of this region.

To find all potential separation regions, first, we need to locate the maximum number of vertical transitions above the baseline. This done by searching for the horizontal line with the maximum number of pixel value change (transition from 1 to 0 or from 0 to 1). The search starts from the baseline index to the height of the binary line image and from right to left for each horizontal line. Thus, for each horizontal line, we count the number of transitions which defined as changing the pixel value from black to white or vise versus. Identifying the maximum number of transitions above baseline can ensure findings of all potential separation regions, in addition to minimizing the number of ligatures as much as possible. However, an over-segmentation might occur because some of the characters like "SEEN SHEEN SAD DAD" have two or three transitions above baseline. In Fig. 6 the green lines represent the index of the maximum transition line, while the red lines represent the baseline index. Algorithm 5 summarizes the procedure of detecting the horizontal line index with the maximum number of transitions. The algorithm takes a binary line image and the baseline index as an input and returns the index of the line with the maximum number of transitions.

Algorithm 5 Finding maximum transitions

```
1: **INPUT**: Line, BaselineIndex
2: **SET** $MaxTransitions \leftarrow 0$
3: **SET** $MaxTransitionsIndex \leftarrow BaselineIndex$
4: **SET** $i \leftarrow BaselineIndex$
5: **while** $(i < height(Line))$ **do**
6:     **SET** $CurrentTransitions \leftarrow 0$
7:     **SET** $FLAG \leftarrow 0$
8:     **SET** $j \leftarrow 1$
9:     **while** $j <= Width(Line)$ **do**
10:        **if** $(Line(i,j) == 1 \&\& FLAG == 0)$ **then**
11:            $CurrentTransitions++$
12:            $FLAG \leftarrow 1$
13:        **end if** $(Line(i,j)! = 1 \&\& FLAG == 1)$ **then**
14:            $FLAG \leftarrow 0$
15:        **end if**
16:    **end while**
17:    **if** $CurrentTransitions >= MaxTransitions$ **then**
18:        $MaxTransitions \leftarrow CurrentTransitions$
19:        $MaxTransitionsIndex \leftarrow i$
20:    **end if**
21: **end while**
22: **OUTPUT** MaxTransitionsIndex
```

After finding the maximum transition index, we need to identify each separation region by three indices: start index which represents the beginning of separation region, end index which represents the ending of separation region, and cut index which represents the borderline between consecutive characters such that the character segment which can be defined by the area between two consecutive cut indexes as shown in Fig. 7c. Therefore, tracing through the maximum transition horizontal line index and detect-

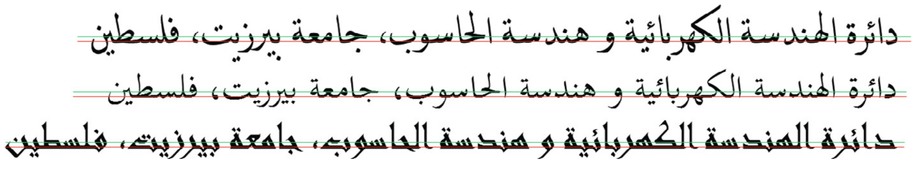
دائرة الهندسة الكهربائية و هندسة الحاسوب، جامعة بيرزيت، فلسطين
دائرة الهندسة الكهربائية و هندسة الحاسوب، جامعة بيرزيت، فلسطين
دائرة الهندسة الكهربائية و هندسة الحاسوب، جامعة بيرزيت، فلسطين

Fig. 6. Baseline index in red and maximum transition index in green for lines in Fig. 3.

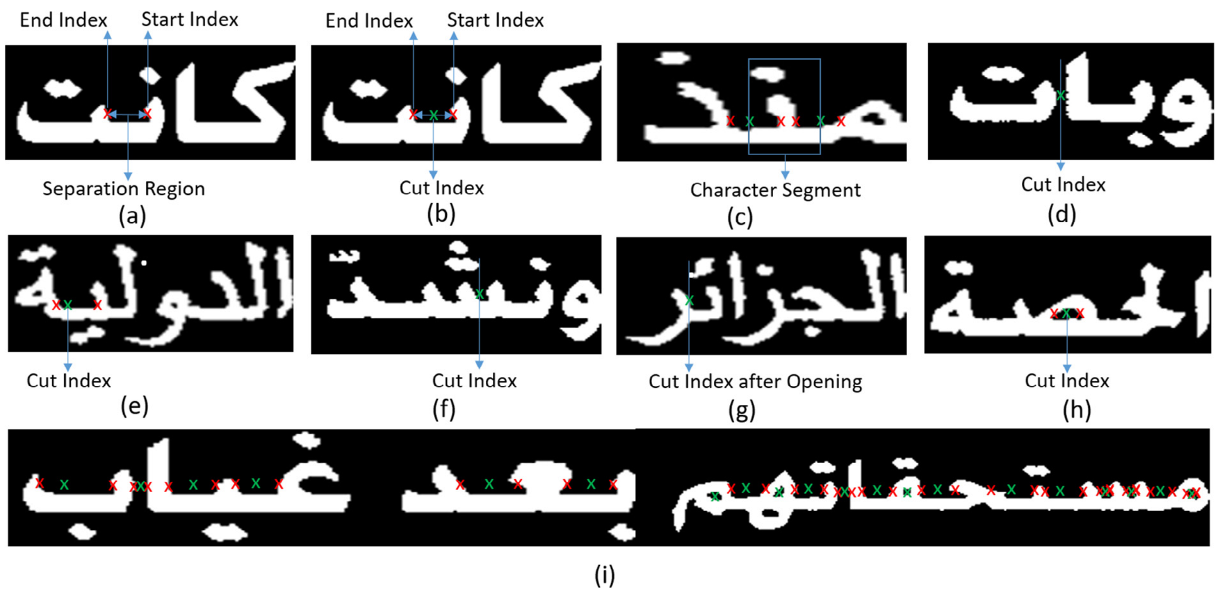

Fig. 7. Some examples on separation region indexes.

ing changes provides the starting and ending indices of each separation region. Indeed, changing the pixel value from white to black represents the start index of the separation region, while changing the pixel value from black to white represents the end of the separation region as shown in Fig. 7a. Based on the shape of the Arabic characters, the cut index is in the area between the start and end of the separation region. The cut index needs to be optimized to preserve the shape of the character, especially when the overlapping exists and also when aiming at decreasing false positive of the identified regions. Indeed, for neither connected nor overlapped characters, the best separation location should be the point where the vertical projection equals zero and is selected around the middle of the separation region as shown in Fig. 7d.

For connected characters, the best separation location should be the point where no vertical intersect is found with the structure of the character itself or the next character, but with the baseline only. Thus, the cut index is at the middle of the separation region if the vertical projection at the middle equals the baseline thickness which computed as the mode value (the Most Frequent Value MFV) of the vertical projection of the input binary line image as shown in Fig. 7b. If the vertical projection at the middle is greater than MFV value, then an overlapping is found either with the previous or next character or with the dots. In this case, we search for a point from the middle of the separation region towards the end of the separation region and the cut index will be the first point having a vertical projection equal MFV value as shown in Fig. 7e. For some cases and especially in small font size there is no point between middle and end indexes of the separation region having a vertical projection equal MFV value. To overcome this problem we apply morphological opening operation with (2*2) structure element to the line image before computing vertical projection to remove the overlapping between dots and the next character as shown in Fig. 7e. If there is no point having a vertical projection equal MFV between mid and start index, we search for a point from the middle of the separation region towards the start of the separation region and the cut index will be the first point having a vertical projection equal MFV value as shown in Fig. 7f. Finally, if there is no point having a vertical projection equal MFV value, the cut index will be at the middle of the separation region which can be treated as an invalid separation region like the point inside as shown in Fig. 7f. Algorithm 6 summarizes the procedure to identify all separation regions within the input line. The algorithm takes a binary line image, word image, and Max transition index as an input, and returns the list of all identified separation regions. Fig. 7I show the location of the indexes of the identified separation regions.

Algorithm 6 Cut point identification

```
1: INPUT: Line, Word, MTI
2: SET $i \leftarrow 1$
3: SET $FLAG \leftarrow 0$
4: $LineImage \leftarrow open(Line)$
5: $VP \leftarrow verticalProjection(Line)$
6: SET $MFV \leftarrow mode(VP)$
7: while $i <= Width(Word)$ do
8:   if $(Word(MTI, i) == 1 \&\& FLAG == 0)$ then
9:     SET $SR \leftarrow newSeparationRegions()$
10:      $SR.EndIndex \leftarrow i$
11:      $FLAG \leftarrow 1$
12:   else if $(Word(MTI, i)! = 1 \&\& FLAG == 1)$ then
13:      $SR.StartIndex \leftarrow i$
14:      $MidIndex \leftarrow (EndIndex + StartIndex)/2$
15:      if there exists $VP[k]$ == zero between start and
16:        end indexthen
17:          $SR.CutIndex \leftarrow$ the nearest $k$ to mid index
18:      end if $VP[MidIndex]$ == $MFV$ then
19:        $SR.CutIndex \leftarrow$ mid index
20:      end if there exists $VP[k] <= MFV$
21:        and end index then
22:        $SR.CutIndex \leftarrow$ the nearest $k$ to mid index
23:      end if there exists $VP[k] <= MFV$
24:        between start and mid index then
25:        $SR.CutIndex \leftarrow$ the nearest $k$ to mid index
26:      else
27:        $SR.CutIndex \leftarrow MidIndex$
28:      end if
29:      $SeparationRegions.add(SR)$
30:      $FLAG \leftarrow 0$
31:   end if
32: en while
33: OUTPUT: SeparationRegions
```

3.2.3. Separation regions filtration

Employing the maximum number of transitions above the baseline to identify borderlines between consecutive characters can ensure findings of all potential segmentation points. However, it may cause an over-segmentation in some characters because some characters have two or three transitions above baseline, which need further processing to decide whether the region is a valid separation region or not. The proposed method in handling false positive is caused by over-segmentation which relies on a set of statistical and topological features that are independent of font variations.

The proposed algorithm starts with checking the vertical projection at the cut index of the separation region. Therefore, if it equals zero, then the separation region is valid as in Fig. 8a. Besides, the separation region is valid if there is no connected path between the start and the end index of a current region as in Fig. 8b. Usually, the separation between connected characters in Arabic script occurs in the baseline in most of the cases. However, since some transitions caused by a change in the same character, the separation region is invalid in the following special cases: (i) if the segment/character which defined between the cut index of the previous region and the cut index of the next region as in Fig. 8c has a Hole like (ـصـ، ـضـ، ـطـ، ـظـ، ـو، ـفـ، ـقـ، ـة) characters. In this case, the cut index located inside the character, (ii) if there is no baseline between the start and end index and the sum of horizontal projection below baseline is greater than the sum of horizontal projection above baseline like ص ، ش ، س ، ن ، ضـق) characters as in Fig. 8d, and (iii) if it is the last region or if the vertical projection at the cut index of the next region equals zero and the height of the top-left pixel of the region is less than half the distance between baseline and the top pixel of the line like (د، ـب، ـت، ـث، ـف، ـق) as in Fig. 8g and h. For the second case, if the sum of horizontal projection above baseline is greater than the sum of horizontal projection below the baseline and the vertical projection at the cut index is equal to MFV, then the region is valid as in Fig. 8e, otherwise the region is not valid as in Fig. 8f.

After passing the above rules, if the segment/character is not a stroke, then the region is valid as in Fig. 8i. Otherwise, it could be one of the following characters (ـبـ،ـيـ، ـتـ، ـثـ، ـنـ) or part of one of the following characters (ـصـ ، ـضـ، س، ش). Indeed, the segment is a stroke if it meets the following features: (i) single connected component, (ii) the sum of horizontal projection above baseline is greater than the sum of horizontal projection below baseline, (iii) the height of the segment is less than twice the second peak value of the horizontal projection, (iv) the mode value of the horizontal projection is equal to MFV value, and (v) the segment has no Holes. Fig. 8j shows an example of a segment that meets these requirements. Therefore, if the current segment is a stroke and has a dot/s below or above baseline, then the region is valid and the segment represents one of the following characters (ـتـ، ـثـ، ـنـ ـبـ،ـيـ،) as in Fig. 8k and l. On the other hand, if the current segment, next segment, and/or after next segment is strokes without dots, then the current region, next region, and the after next region are valid which represent the (س) character as in Fig. 8m. Otherwise, if the current segment, after next segment, strokes without dots while the second segment is a stroke with dots, then the current region, next region, and the after next region are valid which represent the (ش) character as in Fig. 8n. Finally, if the next segment is not a stroke or stroke with dot/s, then the current region is invalid which represent one of the following

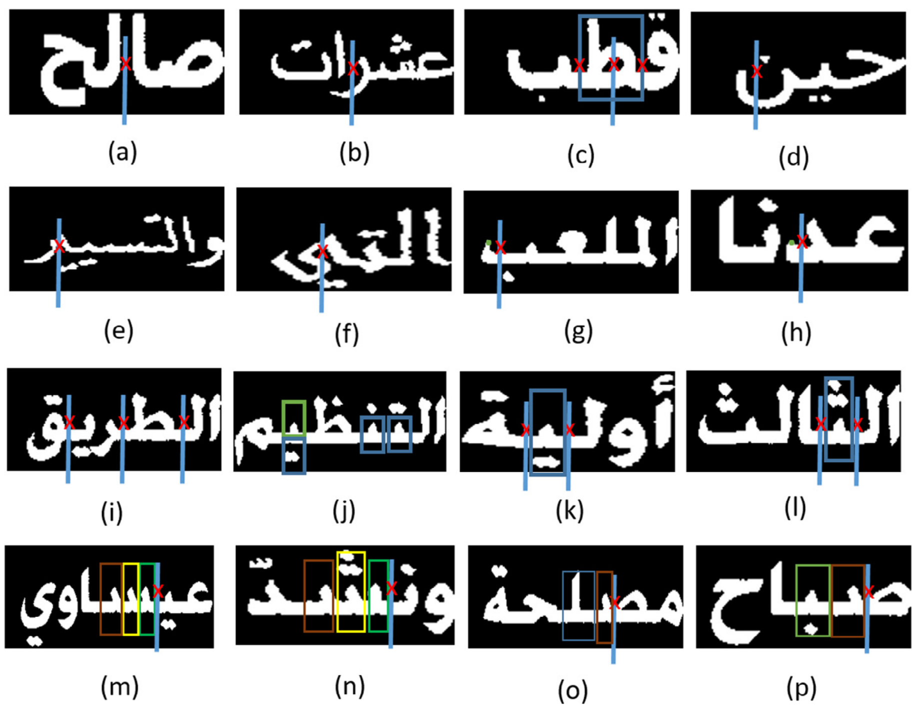

Fig. 8. Some examples on separation region filtration.

two characters (ـصـ ،ـضـ) as in Fig. 8o and p. To detect the dot/s and their location, we divide the stroke segment into two sub-segments where the first one is located above baseline index and the other below the baseline index as shown in Fig. 8j. Thus, if the two sub-segments contain more than two connected components, then the stroke has a dot/s and is located in the segments that have two or more connected component. Otherwise, the stroke didn't have a dot/s.

Algorithm 7 summarizes the proposed method of separation region filtration. It takes as an input the line image, word image, SRL, baseline index, max transitions index and MFV value. The algorithm was written to eliminate false positive as much as possible besides optimizing computational time. Therefore, the false-positive was divided into four main cases and they ordered based on (i) their frequency in Arabic script, and (ii) the discrimination power of the employed features/rules and (iii) their computation time. In the algorithm, SRL represents separation region list, SR: current separation region, HP: horizontal projection of SR, VP: vertical projection of SR, SHPA: sum of HP above baseline index, SHPB: sum of HP below baseline index, SEGP: the segment between previous cut index and next cut index, SEG: the segment between current cut index and next cut index, SEGN: the segment between next cut index and after next cut index, SEGNN: the segment between after next cut index and after next cut index, Seg Index: the index of left top pixel of SR, and Line Index: the index of left top pixel of Line Image.

3.3. Character extraction

The final stage after separation region filtration is the actual separation of the characters. Depending on the location of the optimized cut index, there are two types of separation, vertical cut method, and connected component method. Vertical cut method is simple because no further processing is required and used in the following cases: (i) when there is no overlapping and the vertical projection at cut index is equal zero as in Fig. 9a, (ii) when there is no overlapping and the vertical projection at cut index is equal MFV value as in Fig. 9b, and (iii) when there is an intra-overlapping where the vertical projection at cut index is non-zero and there is a connected path between start and end index as in Fig. 9c. This case happens in some font type and only with some cases like (ـكـ) character at the middle of the word and (ـع ،ـغ ،ـج ،ـح ،ـخ) characters at the end of the word, or in the second type of separation, the connected components of the segment after previous cut index is taken to ensure the extraction of the full shape of the character which improves the performance of the recognition phase. It used when an inter-overlapping where the vertical projection at cut index is non-zero and there is no connected path between start and end index as in Fig. 9d.

Algorithm 7 Separation region filtration

```
1: INPUT: Line, Word, SRL, BaselineIndex,
   MaxTransitionsIndex, MFV
2: SET $i \leftarrow 0$
3: while $i$ is less than length of SRL do
4:   $SR \leftarrow SRL(i)$
5:   if $VP[SR.CutIndex]$ equal zero then
6:     $ValidSeparationRegions.add(SR), i \leftarrow i+1$
7:   else if no path between start and end index then
8:     $ValidSeparationRegions.add(SR), i \leftarrow i+1$
9:   else if SEGP has a Hole then
10:      $i \leftarrow i+1$
11:    else if no baseline between start and end index then
12:      if $SHPB > SHPA$ then
13:        $i \leftarrow i+1$
14:      else if $VP[SR.CutIndex]$ less than MFV then
15:        $ValidSeparationRegions.add(SR), i \leftarrow i+1$
16:      else
17:        $i \leftarrow i+1$
18:      end if
19:    else if SR is the last region or cut
   index of next regionis equal zero and
20:      height of the segment is less than half line heightthen
21:      $i \leftarrow i+1$
22:    else if SEG is not a Stroke then
23:      if
   no baseline between start and end index of next region and
24:      cut index of next region is less than or equal MFV
   then
25:        $i \leftarrow i+1$
26:      else
27:        $ValidSeparationRegions.add(SR), i \leftarrow i+1$
28:      end if
29:    else if SEG is stroke with dots below or above then
30:      $ValidSeparationRegions.add(SR), i \leftarrow i+1$
31:    else if SEG is stroke without Dots then
32:      if SEGN stroke without Dots then
33:        $ValidSeparationRegions.add(SR), i \leftarrow i+3$
34:      end if
35:      if
   SEGN stroke with Dots and SEGNN stroke without Dots then
36:        $ValidSeparationRegions.add(SR), i \leftarrow i+3$
37:      end if
38:      if SEGN is Not Strok or Strok with Dots then
39:        $i \leftarrow i+1$
40:      end if
41:    end if
42: end while
43: OUTPUT: ValidSeparationRegions
```

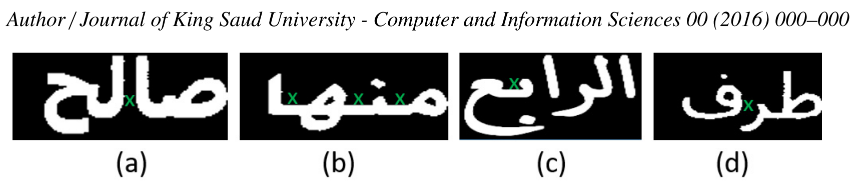
Author / Journal of King Saud University - Computer and Information Sciences 00 (2016) 000–000
صالح
منها
الرابع
طرف
(a)
(b)
(c)
(d)

Fig. 9. Some examples on character extraction cases.

4. Experiments and results

In this section, the effectiveness of our proposed method evaluated by using a well-known published dataset and performance measurement. In addition, the obtained results are discussed, besides comparing them with other related methods.

4.1. Dataset description

Several datasets are available for Arabic character recognition (Lawgali, 2015). The proposed algorithm was tested with Arabic Printed Text Image Database (APTI) (Slimane et al., 2009). APTI dataset is a standard large-scale benchmark used in Arabic Recognition Competition: Multi-font Multi-size Digitally Represented Text held in the context of the 11th International Conference on Document Analysis and Recognition (ICDAR2011). It has a variability in the generation procedure of text images including different font types (Andalus, Arabic Transparent, Advertising Bold, Diwani Letter, DecoType Thuluth, Simplified Arabic, Tahoma, Traditional Arabic, DecoType Naskh, and M Unicode Sara) as shown in Fig. 10 (Slimane et al., 2009), sizes (6, 7, 8, 9, 10, 12, 14, 16, 18, and 24 points), and styles (plain, italic, bold and combination of italic and bold). The images in the dataset are 72 dpi resolution images. Besides, it includes very large open-vocabulary, various forms of ligatures, overlapping characters, low-resolution images, and variability of the height of each word image. Moreover, APTI covers all styles that are widely used in computer typing, newspapers, magazines, and books.

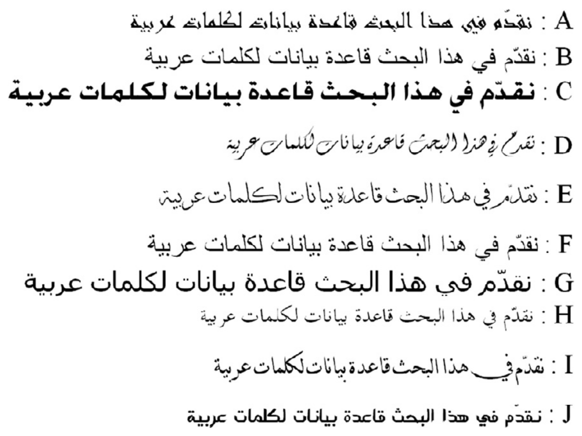
A : نقدّم في هذا البحث قاعدة بيانات لكلمات عربية
B : نقدّم في هذا البحث قاعدة بيانات لكلمات عربية
C : نقدّم في هذا البحث قاعدة بيانات لكلمات عربية
D : نقدّم في هذا البحث قاعدة بيانات لكلمات عربية
E : نقدّم في هذا البحث قاعدة بيانات لكلمات عربية
F : نقدّم في هذا البحث قاعدة بيانات لكلمات عربية
G : نقدّم في هذا البحث قاعدة بيانات لكلمات عربية
H : نقدّم في هذا البحث قاعدة بيانات لكلمات عربية
I : نقدّم في هذا البحث قاعدة بيانات لكلمات عربية
J : نقدّم في هذا البحث قاعدة بيانات لكلمات عربية

Fig. 10. Fonts used in APTI Database: (A) Andalus, (B) Arabic Transparent, (C) Advertising Bold, (D) Diwani Letter, (E) DecoType Thuluth, (F) Simplified Arabic, (G) Tahoma, (H) Traditional Aatbic, (I) DecoType Naskh, (J) M Unicode Sara.

4.2. Experiment setup and results

This section presents the results of conducted experiments using API dataset. We used MATLAB to implement and then experiment our proposed method because MATLAB platform provides well-implemented toolbox in image processing. The performance measured in terms of word/character segmentation accuracy which is computed by the ratio of the number of word/character that correctly segmented to the total number of input word/character.

4.2.1. Word segmentation results

The results of the words segmentation stage in terms of word segmentation accuracy reported in Table 2. The proposed word segmentation method experimented with 1800 lines (around 24,816 words) with ten font types, three styles, and 10 font sizes with an average accuracy of 97.7%. In general, the results show that the algorithm has almost the same performance when varying font type, style, and size except for Thulth, Diwani, and Naskh fonts. Indeed, these fonts have lower accuracy than other fonts due to the overlapping between consequent words which leads to under segmentation in most cases as shown in Fig. 11a. For other font types, under segmentation usually caused by the existence of noise in the input images as in Fig. 11c and form the absence of space after punctuation marks between words, which results in miss classifying the space or reduce the value of spaces between words as shown in Fig. 11b. On the other side, the over-segmentation in all fonts as reported in Table 2 are negligible and mostly caused by the existence of noise in the input images.

4.2.2. Character segmentation results

The results of the character segmentation stage in terms of character segmentation accuracy are reported in Table 3. The proposed method experimented with 24,00 words (around 100,000 characters) with ten font types, three styles, and ten font sizes. The accuracy of character segmentation reported in two ways: first, the accuracy defined as the ratio of the total number of correctly segmented characters (without including ligatures) to the total number of input characters. In general, the results show that the algorithm achieves similar accuracy with an average of 91.77% when varying font type, style, and size except for Thulth, Diwani, and Naskh especially for the under segmentation problem. Because the structure of these fonts produces lots of ligatures that consist of two or three characters due to the intra-overlapping problem as shown in Fig. 12a (Naz et al., 2013). The second way, accuracy defined as the ratio of the total number of correctly segmented characters and well-known ligatures to the total number of input characters. Indeed, most Arabic OCR systems treated ligatures as new classes since segmenting them are very hard because the

Table 2
Word segmentation results.

Font Type	#Input Words	#Words Correctly Segmented	#Over Segmentation	#Under Segmentation	Accuracy
Tahoma	2319	2305	1	13	99.4%
Advertising Bold	1773	1750	2	21	98.7%
Andalus	2833	2811	4	18	99.2%
Thulth	2648	2495	6	147	94.2%
Diwani	3548	3299	3	246	92.9%
M Unicode Sara	3090	3074	3	13	99.5%
Naskh	2921	2808	5	108	96.1%
Simplified Arabic	2884	2850	5	29	98.8%
Traditional Arabic	2800	2764	4	32	98.7%
Arabic Transparent	2860	2837	3	20	99.1%
Average Accuracy					**97.7%**

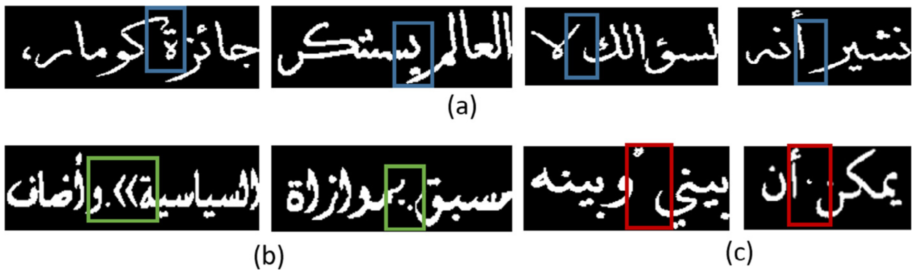

Fig. 11. Some examples on word under segmentation.

Table 3
Character segmentation results.

Font Type	#Input Characters	#Correctly Segmented Characters	#Over Segmentation	#Under Segmentation	Accuracy Exculding Ligatures	Accuracy Including Ligatures
Tahoma	12,262	11,892	278	92	96.98%	97.00%
Advertising Bold	7,896	7,307	176	413	92.54%	94.70%
Andalus	14,032	13,193	406	433	94.02%	96.31%
Thulth	11,582	9682	311	1,589	83.59%	92.40%
Diwani	18,329	16,951	518	860	92.48%	95.13%
M Unicode Sara	14,589	13,500	306	783	92.54%	95.07%
Naskh	12,585	11,345	342	898	90.02%	94.52%
Simplified Arabic	13,572	12,383	379	810	91.24%	96.10%
Traditional Arabic	12,751	11,860	320	571	93.00%	95.50%
Arabic Transparent	13,120	12,630	215	275	91.24%	96.26%
Average Accuracy					**91.77%**	**95.30%**

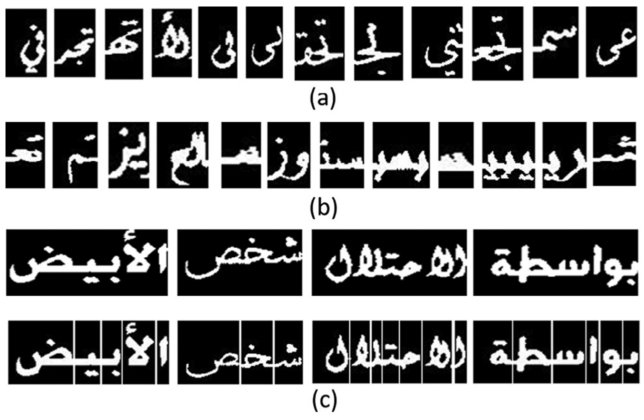

Fig. 12. Some examples on character segmentation results, (a) some examples on ligatures, (b) some examples on segmentation errors, (c) some examples on character segmentation stage.

characters overlapped vertically and do not touch each other which lead to un-clear segmentation point. In this case, the proposed character segmentation method achieves an average accuracy of 95.3%. Besides, the proposed method reduces the total number of ligatures compared to the list of ligatures defined in w3c (2001).

In contrast, most of the segmentation errors cases which are common with all fonts happened in (س، ش) characters especially when small font size like 6, 7, 8, and 9 which rarely used in Arabic typing. In small font size, segmentation points may not be visible and thus cannot be detected because the transition above baseline is very small. Besides, things get worse in terms of under and over-segmentation with the existence of lower resolution images, noise and binarization problems as shown in Fig. 12b. This makes some isolated characters, which can be detected in a larger size, may not be detected in a smaller size. Therefore, excluding these very small sizes, the algorithm achieves an average accuracy of 97.51%. Fig. 12c shows some output of the character segmentation stage.

Table 4
Comparing with other related works.

Reference	Segmentation Method	Data-set	Font Types	Font Sizes	Font Styles	Accuracy
Zheng et al. (2004)	Vertical histogram and some structural characteristics rules	500 samples of Arabic text	Simplified Arabic and Arabic Transparent	12, 14, 16, 18, 20 and 22	Plain	94.8
Mousa et al. (2017)	Projection-based using profile's amplitude filter	50,931 words, 224,781 characters	Not reported	Not reported	Not reprted	98%
Anwar and Adiwijaya (2015)	Vertical Projection	127 sentences composed of 1061 letters	Traditional Arabic	70pt	Not reported	97.55%
Marwa Amara and Zidi (2016)	Histogram projection and contextual topographies	APTI: 500 samples of Arabic words	Advertising Bold	10pt	Plain, Bold, Italic	85.6%
Radwan and Khalil (2016)	Multichannel Neural Networks	APTI	Arial, Tahoma, Thuluth, and Damas	18	Plain	95.5%
Proposed Method	Projection profile method with statistical and topological features	APTI: around 100,000 characters	Andalus, Transparent, Advertising Bold, Diwani, Thuluth, Simplified, Tahoma, Traditional, Naskh, and M Unicode Sara	10, 12, 14, 16, 18, and 24	Pain, Italic, Bold	97.51%

4.2.3. Comparison with related work

Table 4 shows our method compared with previous related works in terms of the segmentation method, used Dataset, font types, styles, sizes, and accuracy. Indeed, it is fairer to compare with other related works using the same dataset (same font types, sizes and styles). However, most of the authors tested their work on their own collected data and didn't make it public. In addition, the implementation code of the proposed methods in the related work is not available and sometimes it is difficult to write the code since it may depend on parameters, hypothesis, tools and APIs that are not mentioned clearly in published paper. Therefore, we tested our method using APTI benchmark dataset in order to make some fair comparisons with other related works. As shown from the Table 4 and other methods summarized in section two this method has one or more of the following weakness: (i) some of these methods are font type, style, and size-dependent, (ii) parameterized method, (iii) handle simple font types and styles, and (iv) tested on a small and unpublished standard dataset. On the other side, the proposed method has the following effectiveness: (i) Font type, style, and size-independent, (ii) Non-parametric method, (iii) handles simple and complex font types and styles, (iv) solves the problem of inter overlapping between sub-words, (v) optimizes the location of the cut point to save character shape, (vi) reduces the number of ligatures, and (vii) tested on large and publish dataset. Besides, we proposed a simple, efficient, font invariant and parameter-free word segmentation approach that can employed at the recognition stage as a post-processing method to enhance the recognition performance.

5. Conclusion

Character segmentation of cursive text such as Arabic text is error-prone and treats as the most critical stage for any OCR System. It is the stage where most of the errors occur that affect directly the result of feature extraction and recognition stages. In this paper, an efficient word and character segmentation algorithm for printed Arabic text is proposed. The algorithm is based on using vertical projection along with Interquartile Range (IQR) method for word segmentation and based on using vertical projection method along with a set of statistical and topological shape features for character segmentation. The proposed algorithm developed such that the segmentation is done in such a way to minimize under and over-segmentation problems and to maximize the recognition rate by saving the character's shape. The algorithm was able to segment characters of complex Arabic fonts where the overlapping between characters exists. The algorithm experimented on the APTI dataset which has variability in the generation procedure of text images including different font types, font size, and styles. The experimental results showed the reliability of our algorithm in segmenting words and characters correctly with an average accuracy of 97.7% and 97.51% respectively. Compared with the previously-proposed approach, our algorithm has the following points of strength: (i) promising results with variability of font types, sizes, and styles, (ii) non-parametric method, (iii) facilitating the recognition stage by saving the shape of the extracted characters as much as possible, (iv) reducing number of ligatures, and (v) proposing an efficient word segmentation which can be used to enhance the accuracy of the recognition stage through incorporating language dictionaries and spell-checks method.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Ahmed, P., Al-Ohali, Y., 2000. Arabic character recognition: progress and challenges. J. King Saud Univ.-Comput. Inf. Sci. 12, 85–116.

Alfonse, M., Almorsy, M., Samir, Barakat M., 2010. Eastern arabic handwritten numerals recognition. Int. J. Comput. Electr. Eng., 277–282.

Alginahi, Y.M., 2013. A survey on arabic character segmentation. Int. J. Document Anal. Recogn. 16 (2), 105–126.

Al-Muhtaseb, H., Mahmoud, S., Qahwaji, R., 2008. Recognition of off-line printed arabic text using hidden markov models. Signal Process. 88, 2902–2912.

AL-Shatnawi, A., Omar, K., 2008. Methods of arabic language baseline detection – the state of art 8. .

Anwar, K., Adiwijaya, Nugroho H., 2015. A segmentation scheme of arabic words with harakat. In: 2015 IEEE International Conference on Communication, Networks and Satellite (COMNESTAT), pp. 111–114.

Bushofa, B.M.F., Spann, M., 1997a. Segmentation of arabic characters using their contour information. Proceedings of 13th International Conference on Digital Signal Processing, vol. 2, pp. 683–686.

Bushofa, B., Spann, M., 1997b. Segmentation and recognition of arabic characters by structural classification. Image Vis. Comput. 15 (3), 167–179.

Casey, R.G., Lecolinet, E., 1996. A survey of methods and strategies in character segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 18 (7), 690–706. https://doi.org/10.1109/34.506792.

Cowell, J.H.F., 2001. Thinning arabic characters for feature extraction. In: Proceeding of SPIE. Document Recognition III, pp. 181–185.

Deng, W., Iyengar, S.S., Brener, N.E., 2000. A fast parallel thinning algorithm for the binary image skeletonization. Int. J. High Perform. Comput. Appl. 14 (1), 65–81.

Elnagar, A.F., Al-Kharousi, S.H., 1997. Handwritten arabic and hindi numerals recognition based on a decision tree classifier. In: IEEE International Conference on Systems, Man and Cybernetics, pp. 983–988.

Elnagar, A., Harous, S., 2003. Recognition of handwritten hindu numerals using structural descriptors. J. Exp. Theor. Artif. Intell. 15 (3), 299–314.

Fitriyatul Qomariyah, F.U., Mahmudy, W.F., 2017. The segmentation of printed arabic characters based on interest point. J. Telecommun. Electron. Comput. Eng. 9, 19–24.

Gouda, A.M., Rashwan, M.A., 2004. Segmentation of connected arabic characters using hidden markov models. In: 2004 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications, 2004 CIMSA, pp. 115–119.

Han, Jiawei, Kamber, Micheline, Pei, J., 2012. Data Mining Concepts and Techniques .

Islam, N., Islam, Z., Noor, N., 2007. A survey on optical character recognition system. CoRR. arXiv:1710.05703.

Lawgali, A., 2015. A survey on arabic character recognition. Int. J. Signal Process. Image Process. Pattern Recogn. 8 (2), 401–426.

Lorigo, L.M., Govindaraju, V., 2006. Offline arabic handwriting recognition: a survey. IEEE Trans. Pattern Anal. Mach. Intell. 28 (5), 712–724.

Louloudis, G., Gatos, B., Pratikakis, I., Halatsis, C., 2009. Text line and word segmentation of handwritten documents. Pattern Recogn. 42 (12), 3169–3183.

Mahmood, A., 2013. Arabic & urdu text segmentation challenges & techniques. Int. J. Comput. Sci. Technol. 4, 32–34.

Margner, V., 1992. Sarat-a system for the recognition of arabic printed text. In: 11th International Conference on Pattern Recognition Methodology and Systems, pp. 561–564.

Marwa Amara, K., Zidi, K.G.S.Z., 2016. New rules to enhance the performances of histogram projection for segmenting small-sized arabic words. In: International Conference on Hybrid Intelligent Systems.

Mazen Bahashwan, S.A.B., Sheikh, U., 2017. Efficient segmentation of arabic handwritten characters using structural features. Int. Arab J. Inf. Technol. 14.

Mehran, R., Pirsiavash, H., Razzazi, F., 2005. A front-end ocr for omni-font persian/arabic cursive printed documents. In: Digital Image Computing: Techniques and Applications (DICTA'05), pp. 56–56.

Mousa, M.A.A., Sayed, M.S., Abdalla, M.I., 2017. Arabic character segmentation using projection based approach with profile's amplitude filter. ArXiv:abs/1707.00800.

Naz, S., Hayat, K., Imran Razzak, M., Waqas Anwar, M., Akbar, H., 2013. Arabic script based language character recognition: Nasta'liq vs naskh analysis. In: 2013 World Congress on Computer and Information Technology, WCCIT 2013, pp. 1–7. ISBN 978-1-4799-0460-0.

Naz, S., Umar, A.I., Ahmed, S.B., Shirazi, S.H., Razzak, M.I., Siddiqi, I., 2014. An ocr system for printed nasta'liq script: a segmentation based approach. In: Multi-Topic Conference (INMIC), 2014 IEEE 17th International. IEEE, pp. 255–259.

Naz, S., Umar, A.I., Shirazi, S.H., Ahmed, S.B., Razzak, M.I., Siddiqi, I., 2016a. Segmentation techniques for recognition of arabic-like scripts: a comprehensive survey. Educ. Inf. Technol. 21 (5), 1225–1241.

Naz, S., Umar, A., Ahmad, R., Razzak, M., Rashid, S.F., Shafait, F., 2016. Urdu nasta'liq text recognition using implicit segmentation based on multi-dimensional long short term memory neural networks 5.

Omidyeganeh, M., Nayebi, K., Azmi, R., Javadtalab, A., 2005. A new segmentation technique for multi font farsi/arabic texts. In: ICASSP, pp. 757–760.

Radwan, M.A., Khalil, M.I.A.H., 2016. Predictive segmentation using multichannel neural networks in arabic ocr system. In: Artificial Neural Networks in Pattern Recognition. Springer International Publishing, Cham, pp. 233–245.

Rehman, A., Mohamad, D., Sulong, G., 2009. Implicit vs explicit based script segmentation and recognition: a performance comparison on benchmark database 2.

Rosenberg, A., 2012. Using sift descriptors for ocr of printed arabic. .

Saabni, R., 2014. Efficient recognition of machine printed arabic text using partial segmentation and hausdorff distance. In: 2014 6th International Conference of Soft Computing and Pattern Recognition (SoCPaR), pp. 284–289.

Sabbour, N., Shafait, F., 2013. A segmentation free approach to arabic and urdu ocr. Proc. SPIE Int. Soc. Opt. Eng. 8658. https://doi.org/10.1117/12.2003731. 86580N–86580N.

Sari, T., Sellami, M., 2005. Cursive arabic script segmentation and recognition system. Int. J. Comput. Appl. 27, 161–168.

Shaikh, N.A., Mallah, G.A., Shaikh, Z.A., 2009. Character segmentation of sindhi, an arabic style scripting language, using height profile vector. Aust. J. Basic Appl. Sci. 3 (4), 4160–4169.

Slimane, F., Ingold, R., Kanoun, S., Alimi, A.M., Hennebert, J., 2009. A new arabic printed text image database and evaluation protocols. In: 2009 10th International Conference on Document Analysis and Recognition, pp. 946–950.

Timsari, B.F.H., 1996. Morphological approach to character recognition in machine-printed persian words. In: Proceeding of SPIE. Document Recognition III.

w3c, 2001. The unicode standard 3.1. In: w3c.

Zeki, A.M., Zakaria, M.S., 2004. Challenges in recognizing arabic characters. International Islamic University Malaysia (IIUM), Kuala Lumpur, Malaysia, National University of Malaysia (UKM), Bangi, Selangor, Malaysia.

Zeki, A.M., Zakaria, M.S., Liong, C.Y., 2011. Segmentation of arabic characters: a comprehensive survey. Int. J. Technol. Diffus. 2 (4), 48–82.

Zheng, L., Hassin, A.H., Tang, X., 2004. A new algorithm for machine printed arabic character segmentation. Pattern Recogn. Lett. 25 (15), 1723–1729.