Journal of Electronic Imaging

JElectronicImaging.org

Contour-based character segmentation for printed Arabic text with diacritics

Khader Mohammad
Aziz Qaroush
Muna Ayesh
Mahdi Washha
Ahmad Alsadeh
Sos Agaian

Khader Mohammad, Aziz Qaroush, Muna Ayesh, Mahdi Washha, Ahmad Alsadeh, Sos Agaian, "Contour-based character segmentation for printed Arabic text with diacritics," *J. Electron. Imaging* **28**(4), 043030 (2019), doi: 10.1117/1.JEI.28.4.043030.

Contour-based character segmentation for printed Arabic text with diacritics

Khader Mohammad,[a,*] Aziz Qaroush,[a] Muna Ayesh,[a] Mahdi Washha,[a] Ahmad Alsadeh,[a] and Sos Agaian[b]
[a]Birzeit University, Electrical and Computer Engineering Department, Birzeit, Ramallah, Palestine
[b]City University of New York, College of Staten Island and the Graduate Center, New York, United States

Abstract. Current developments in sensors open new possible uses across numerous real-life applications, including optical character recognition (OCR). An OCR system requires incorporation of text processing tools into the sensor functionality. The most critical stage in OCR systems is the segmentation stage. It refers to the challenge of subdividing a text image into characters, which can be individually processed using a classifier. The cursive nature of the Arabic script such as the existence of different shapes for each character according to its location in the word besides the existence of diacritics makes Arabic character segmentation a very challenging task. A robust offline character segmentation algorithm for printed Arabic text with diacritics is developed based on the contour extraction technique. The algorithm works through extracting the up-contour part of a word and then identifies the splitting areas of the word characters. Then a postprocessing stage is used to handle the over-segmentation problems that appear in the initial segmentation stage. The proposed scheme is benchmarked using the APTI dataset and a manually collected dataset consisting of image texts varying in font size, type, and style for more than 38,000 words. The experiments show that the proposed algorithm is able to segment Arabic words with diacritics with an average accuracy of 98.5%. [DOI: 10.1117/1.JEI.28.4.043030]

Keywords: optical character recognition; character segmentation; Arabic words with diacritics.

Paper 190219 received Mar. 7, 2019; accepted for publication Jul. 30, 2019; published online Aug. 30, 2019.

1 Introduction

The Arabic language is one of the most structured. It is the world's 5th most spoken language with a little over 350 million speakers.[1] The existence of information retrieval systems, search engines, editing old documents, and data entry applications increase the need for a reliable Arabic optical character recognition system day by day.

Optical character recognition (OCR) is the process of transferring text image into an editable form to avoid retyping.[2] Developing an OCR system passes through five classical stages, beginning with image acquisition and ending with the recognition stage and passing through preprocessing, segmentation, feature extraction, and recognition. According to image acquisition way, OCR systems are classified into online OCR systems, in which the input image is taken by pen writing on a tablet or smartphones and offline OCR systems, in which the input is usually stored as an image taken by camera, scanner, or other optical devices.[2]

Within the last couple of years, scanning and sensing systems have gained significant attention due to the need for monitoring the quality of images and products by reading labels in important areas, such as home automation, industrial automation, medical aids, mobile health-care, elderly assistance, automotive, traffic management, and many others. Additionally, smart sensors have integrated circuits that can perform one or more of the logic functions, two way communications, and make decisions. Storing information for future analysis is becoming an integral part of the intelligent and scanning system performing functions that previously could not be performed or were not economically viable.[3] Additionally, key technologies that will drive the future Internet of Things (IoT) will be related to smart sensor technologies, including wireless sensor networks, nanotechnology, and miniaturization of sensing devices.[4,5] So choosing the right sensor technology for scanning or capturing input images is essential and it affects processing and segmentation as the sensors are normally employed to gather physical or chemical information from different mediums. In this research, this is also important since the goal of the IoT is to uniquely identify, recognize, signify, and access things for day-to-day life anytime, anywhere, and allow them to be controlled as far as possible through the Internet.[4,5] So the need will be to use the right sensors technology in the different scanners such as in cell phone, the scanner itself to use up to date technology.

The segmentation stage is the most critical stage in building OCR systems. It can be viewed as the main source of errors that can appear in the recognition stage.[6] OCR systems use two different approaches for segmentation: segmentation free approach (holistic approach) and segmentation-based approach (analytical approach). In the segmentation free approach,[7] the recognition is performed without segmenting the words into its low-level segments like characters and diacritics; however, it uses some patterns and look-up dictionary for a certain number of words. This approach is usually used when targeting to recognize particular words like numbers and cities' names. On the other hand, the analytical approach segments each word into low-level segments like characters and diacritics.[8] In this approach, more processing is needed; however, more accurate results can be obtained compared with the segmentation free approach.

Performance of the character segmentation-based OCRs systems is highly dependent on the nature of language.

*Address all correspondence to Khader Mohammad, E-mail: khamadawwad@birzeit.edu

1017-9909/2019/$28.00

Font type	Font style		
	Regular	**Bold**	*Italic*
Traditional Arabic	جامعة بيرزيت	**جامعة بيرزيت**	*جامعة بيرزيت*
Andalus	جامعة بيرزيت	**جامعة بيرزيت**	*جامعة بيرزيت*

(a)

(b)

(c)

Fig. 1 Arabic text properties: (a) example of fonts varying in type and style, (b) diacritics in Arabic script, and (c) different shapes of the same character.

In the Arabic language, the cursive nature of scripts increases the complexity of the segmentation task. Additional factors make the Arabic character segmentation task more challenging, summarized in Refs. 9–11: (i) the availability of different font types and styles as in Fig. 1(a); (ii) the existence of diacritics as in Fig. 1(b); and (iii) the variation of character shape based on its location in the word (at the beginning, middle, end, or separated) as in Fig. 1(c). These properties constitute a major problem in finding the right segmentation point between two consecutive and/or connected characters. Moreover, the presence of diacritics on Arabic characters increases the probability of the under segmentation problem.

Several methods have been presented in the literature to address the Arabic character segmentation problem. However, the challenging nature of the Arabic text (e.g., cursive structure, having complex font types, overlapping between characters, and the existing diacritics) makes character segmentation a very challenging and crucial stage for segmentation-based OCR systems. In addition, previous related works have the following weaknesses: (i) limited works address the segmentation problem when the diacritics exist, (ii) most of them are designed for simple and certain font types, size, or style, and (iii) some of them are tested on small and simple unpublished datasets. In this paper, we present a character segmentation algorithm for printed Arabic text with diacritics using a contour-based tracing technique. The presented algorithm takes a text-line image as an input and consists of three main stages: line segmentation, word segmentation, and character segmentation. Contour extraction method has many advantages over other segmentation methods such as having a clear description of character shape and extracting details for small fonts.[12] Also the errors that are produced when extracting baselines are eliminated and there is no need to adjust the baselines many times. Moreover, the existence of diacritics complicates the process of other segmentation methods like morphological methods, template matching, and artificial neural networks. In addition, it will make an under segmentation problem when using projection-based methods. Hence, the main contributions of this paper are as follows: (i) the proposed method is font-independent in terms of type, size, and style; (ii) solve the problem of overlapping between characters and between subwords; (iii) efficiently handle segmentation of lines, words, and characters for Arabic text with the existence of diacritics; and (iv) finally, in the evaluation stage, experimental results on the collected and APTI dataset prove that our method achieves better performance than the state-of-the-art methods.

The reminder of this paper is organized as follows. Section 2 presents related work about the character segmentation methods designed for Arabic documents. Section 3 describes in details the proposed method. Section 4 shows the dataset used in the experiments and the results of the proposed method. Finally, the conclusion is given in Sec. 5.

2 Related Work

2.1 *Word Segmentation*

Word/subword segmentation is an important step in the segmentation phase for segmentation-based Arabic OCR (AOCR) systems since it facilitates working on the character segmentation stage. In addition, it can be employed as a postprocessing stage after character recognition to increase the recognition rate.

Cheung et al.[13] introduced a segmentation algorithm that uses a technique in which the overlapping Arabic words/subwords are horizontally separated; they also used a feedback loop between the character segmentation stage and final recognition stage. AlKhateeb et al.[14] proposed a method for baseline detection and employed it to extract the connected components of each subword. After detection of the baseline, an iterative process was used to detect the connected components based on the connected black pixels in the subword. Shaikh[15] employed a horizontal projection for line segmentation, and then each subword from the extracted line was segmented using a connected components method. The algorithm failed in the case of overlapped characters. Aljarrah et al.[16] proposed an algorithm for word/subword segmentation based on vertical projection technique where the threshold value should be determined before. The small segments/parts like dots and Hamza have to be removed, which is computationally expensive; also it does not deal with diacritics and different font sizes. In addition, Alipour[17] proposed an algorithm for word/subword segmentation based on the vertical projection profile and using a predefined constant k, which is equal to half of the line height. The problem in this method is the using of a predefined value, which will not be suitable for all different word sizes and styles.

2.2 *Character Segmentation*

OCR systems use two different approaches for segmentation: segmentation-based approach (analytical approach) and segmentation-free approach (holistic approach). In segmentation-based approach, the words are segmented into low-level segments like characters, ligature, and diacritic.[8] In this approach, more processing is needed; however, more accurate results can be obtained. On the other hand, segmentation free approach holistic approach handles the whole word as a unified unit[7] and the recognition is performed without segmenting the words into its low-level segments like characters and diacritics; however, it uses some patterns and look-up dictionary for a certain number of words. This approach is usually used when targeting to recognize particular words like numbers and cities names.

2.2.1 *Segmentation-based*

Segmentation-based method can be classified into: (i) projection profile; (ii) character skeleton based; (iii) contour tracing based; (iv) template matching based; (v) morphological operations based; and (vi) segmentation based on neural networks.[2,12] Projection profiles methods[18–20] are usually used for the purpose of lines, words, subwords, and characters segmentation specifically when there is a clear gap between lines, words, subwords, and characters. Indeed, horizontal projection is used for line segmentation, whereas vertical projection is usually used for word, subword, and character segmentation. Applying these types of methods directly for Arabic character segmentation results in oversegmentation problem since the segmentation region in Arabic words is thinner than round regions. In the skeleton method,[15,21] different thinning techniques are employed for this purpose. In many cases, the shape of characters after applying thinning operation differs from the original one, making the segmentation process more difficult. In contour tracing methods,[22–26] the pixels that form the outer shape of the character or word are extracted. Researchers used many ways to determine the cutting points on the contour. In general, contour-based methods avoid the problems that appear when applying thinning methods because they depend on extracting the structure of the word, which gives a clear description of it. This kind of method is sensitive for noise, which requires one to perform some enhancements as a preprocessing step. Morphological methods[27–29] employ a set of morphological operations for the purpose of segmentation. Usually, closing followed by opening operations is applied. This method is a dependent method, meaning that other techniques have to be used in addition to segmentation. Template matching methods[30,31] usually apply a sliding window over baselines. If any match is noticed then the center pixel in the sliding window is considered as a cutting point. The major limitation in this method is that if the cutting point locates under the baseline. Finally, in neural networks (NNs) segmentation, NNs are used to verify the valid segmentation points by training the NNs over manually classified valid segmentation points from the database of scanned images using features such as black pixel density and holes.

Zeng et al.[19] proposed a machine printed Arabic character segmentation algorithm that employs a vertical projection method and some rules or features including structural characteristics between background regions and character components and the characteristics of isolated Arabic characters to find real segmentation points. Cheung et al.[13] introduced a segmentation algorithm that uses a technique in which the overlapping Arabic words/subwords are horizontally separated, they also used a feedback loop between the character segmentation stage and final recognition stage. In the segmentation stage, a sequence of tentative lines has been produced in two processes, the first process uses Amin's character segmentation algorithm,[32] and the second process uses the convex dominant points detection algorithm developed by Bennamoun.[33]

Shaikh et al.[15] suggested an algorithm for Sindhi text segmentation. The height profile vector (HPV) was employed for the characters' extraction. The extra analysis was done over HPV to determine the locations of the possible segmentation points (PSPs), in some cases the algorithm failed by performing under or oversegmentation.

Omidyeganeh et al.[22] presented a segmentation algorithm based on conditional labeling for up and down contours, and the algorithm was developed for multifont Farsi/Arabic texts. The contour of the subword is measured using a convolution kernel with Lapacian edge recognition-based segmentation detection method. The algorithm goes through several steps including: contour labeling of each subword, contour curvature grouping to improve the segmentation results, character segmentation, adaptive local baseline, and postprocessing, the results showed that 97% of characters of the printed Farsi texts were segmented correctly.

Mostafa[34] developed a segmentation approach for printed Arabic text especially for "simplified Arabic" font with different sizes. The main rule used is that the most characters start with and end before a T-junction on the baseline." This rule was fine for most characters, except for some special characters like "SEEN" س, "SHEEN" ش, "SAD" ص, and "DAD" ض, which had a special treatment. The algorithm was tested and achieved a 96.5% of good segmentation accuracy.

Alipour[17] proposed an improved segmentation method for Persian script where some structural features were used to adjust the related fragments to increase the quality of segmentation. Vertical projection was used to extract the word fragments over the baseline—dots and diacritics were not considered—then the fragments were adjusted in an extra step by merging the small fragments, this step was necessary in the cases where one character is segmented into more than one part like "SEEN" س, "SHEEN" ش, "SAD" ص, and "DAD" ض.

Javed et al.[4] developed a free-segmentation approach for Urdu script. They employed different pattern matching techniques to classify each pattern. The features were extracted from the image and fed them to hidden Markov model (HMM) recognizer, which has an ability to perform recognition with great ease and efficiency.

Khaerula et al.[35] proposed a segmentation scheme of Arabic character with "harakat." First, the image was converted into a morph, then a vertical projection was applied and the locations where the projection value is exactly 2 are identified. If the occurrences such locations in more than three successive rows, the image was split in the middle. The process was repeated until the whole subword/word has been segmented.

Mahmoud et al.[36] presented a projection-based character segmentation algorithm. In order to find the PSPs, the authors employed a profiles amplitude filter to find separation between two characters, which is considered as constant amplitude in the profile, and a simple edge tool to determine whether it is a correct characters connection or not.

Radwan et al.[37] proposed a character segmentation approach based on multichannel neural network. The system recognizes the features of the segmentation window to predict the likelihood of the current window to be 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.

Amara et al.[38] proposed a segmentation method for segmenting Arabic words with small size. They used a vertical projection to find the preliminary segmentation points. Then they employed a set of rules that depend on the contextual

topographies of Arabic writing and baselines positions along with their relation with the characters to find real segmentation points.

Qomariyah et al.[39] proposed a segmentation method using the interesting point that is 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. Their method depends on the extracting picture frame by removing all of the contour pixels in the image, except the contour appropriate with the framework using different iteration.

Firdaus et al.[40] proposed a modified connected component labeling (MCCL) method to perform Arabic character segmentation. They used MCCL along with a set of rules based on the moment location, object location, and height of the object to perform the segmentation, especially for characters that are connected with others.

Amara et al.[41] presented an enhanced segmentation method based on the vertical projection and on some rules to detect the potential segmentation points. Then they employed a binary support vector machine to decide whether to filter the extracted potential segmentation points.

Zoizou et al.[42] proposed a hybrid method for printed Arabic character segmentation based on two of the most known approaches including contour analysis and template matching. First, the text is segmented into lines and words/subwords using horizontal and vertical projections, respectively. The subword is then divided vertically into characters using template matching to segment the descender character if there is one and the contour method deals with the rest of the subword.

2.2.2 *Segmentation free*

Segmentation-free approaches employ the whole word or partial words (e.g., ligatures) as units of recognition rather than characters. Holistic techniques are known to be more robust for Urdu text because characters are overlap, slant, and have different styles and fonts, However, 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.

Naz et al.[43] proposed implicit segmentation of printed Urdu text-lines written in the Nasta'liq writing style using multidimensional long short-term memory (LSTM) recurrent neural networks (RNN) with an output layer designed for sequence labeling for recognition. Nashwan et al.[44] introduced an efficient, holistic Arabic OCR system using lexicon reduction approach based on clustering of similar shaped words. They used a discreet cosine transform-based features to compute global word level along with local block-based features to generalize new font sizes and types. Din et al.[45] presented a segmentation-free OCR for printed Urdu Nastaliq font using ligatures as units of recognition. The ligatures are extracted from text lines then separated into primary and secondary ligatures and multiple instances of the same ligature are grouped into clusters. They rely on statistical features and employ HMMs for classification. Rawls et al.[46] presented a simple, effective deep learning approach for recognizing machine print text from raw pixels. They used a fully connected neural network for high-level feature extraction over a sliding window. Then extracted features are directly fed into a stacked bidirectional LSTM. Su and Lu[47] proposed a novel word-level scene text recognition without character segmentation. The proposed method consists of three key components, including sequential feature generation, which converts a word image into sequences of column feature; multilayer RNN model with bidirectional LSTM model training classifies the two sets of sequential data; and an ensembling technique that combines outputs of multiple RNNs to produce improved word recognition accuracy. Namysl and Konya[48] proposed a segmentation-free OCR system that combines deep learning methods, synthetic training data generation, and data augmentation techniques. They used a hybrid model of convolutional neural network encoder to extract high-level features from text images and an RNN to examine the interactions between input elements.

3 Proposed Work

Our approach is a segmentation-based approach (analytical approach), which consists of three main stages as shown in Fig. 2. The proposed approach takes a binary image of multiple lines as an input and produces a set of binary images consisting of one character or ligature. In our algorithm, the segmentation is performed at three levels: line segmentation, word segmentation, and character segmentation. In this paper, we focus only on the line/word/characters segmentation stages, assuming that the input image is well preprocessed (e.g., binary images with noise eliminated and skew corrected, the text is separated from nontext). Therefore, we concentrated only to solve the complexity of Arabic segmentation (line, word, and characters) especially when the diacritics are exists.

3.1 *Line Segmentation*

For line segmentation stage, a horizontal projection method was applied to find the global maximum peak and its locations. The location of the global maximum peak performs the baseline for a single line in the input segment. Then a line was drawn at a global maximum peak to make the main body of the line as one connected component and then extracted by selecting the largest connected components after applying a connected component algorithm on the line segment. The extracted lines are without dots and diacritics, thus, its width is less than the actual width. The line width was assigned by dividing the width of the segment by the approximated count

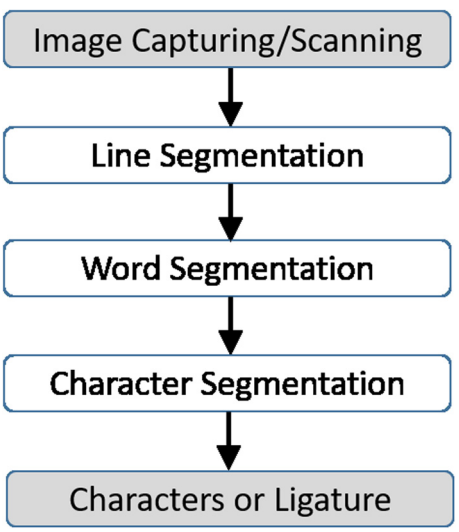

Fig. 2 Proposed work.

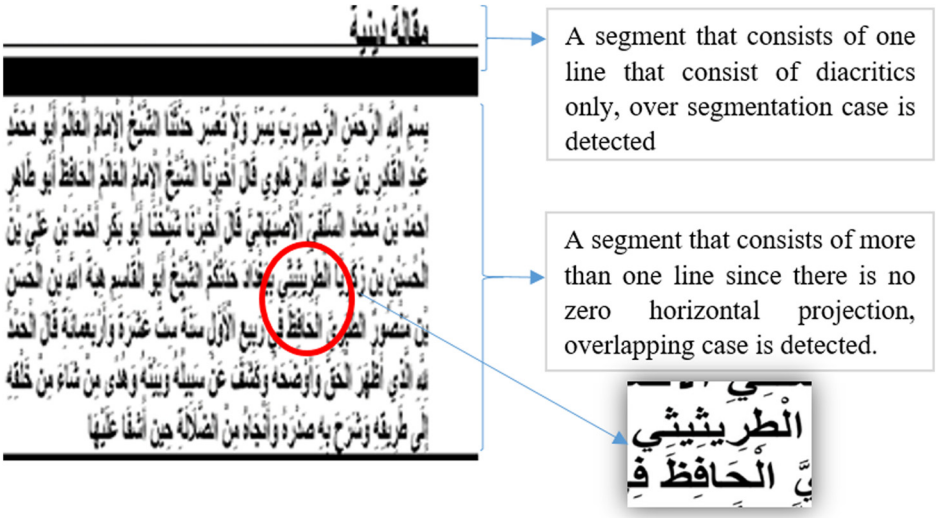

Fig. 3 Over- and under-segmentation example.

of lines. The estimated number of lines in the segment is then determined by dividing the segment width by the line width. If the resulted count of lines inside the segment is greater than one, then the segment is passed for extra processing to split the overlapped lines, but if the segment consists of one line, then the line is passed for over-segmentation checking, and if the over-segmentation condition is met, then the line is linked to the previous or next line. Figure 3 shows an example of the over- and under-segmentation problems; also Algorithm 1 shows the algorithm that summarizes the steps of our line segmentation method.

Algorithm 1 Line segmentation.

1: **INPUT**: **TR** Text Region
2: *Horizontal Projection HP* ← [];
3: *Maximum Peaks MP* ← [];
4: *MP Locations MPL* ← [];
5: *Lines* ← [];
6: *HP*←*HorizontalProjection*(*TR*);
7: *MP*←*globalPeaks*(*HP*);
8: *MPL*←*getLocations*(*MP*);
9: *TR*←*drawLines*(*TR,MPL*);
10: *Lines*←*extractLines*(*TR*);
11: *EstimatedLineCount*←*count*(*Lines*);
12: **if** *EstimatedLineCount* == 1 **then**
13: *checkOverSegmentation* (*Lines*);
14: **else**
15: *handleOverlapping* (*TR*);
16: **end if**
17: **OUTPUT**: Separated Lines

3.2 *Word/Subword Segmentation*

For segmentation-based OCR system, the next stage after the line segmentation is mainly words segmentation. The proposed methodology for words segmentation is mainly based on the vertical projection profile method, which consists of three sequential steps as shown in Fig. 4. The algorithm takes a binary text-line image of printed Arabic text with/without diacritics as an input, and return segmented words/subwords images as an output. Below is the detailed description of each step.

3.2.1 *Text-line image processing*

In this step, to facilitate working on the word and character segmentation stages, two additional versions of the input text-line image are generated. The first version contains just the main body of the text without diacritics and dots, whereas the second one contains only the diacritics and dots. To generate these versions, a horizontal projection method is applied to the original text-line input image [Fig. 5(a)] to find the global maximum peak and its location, this represents the baseline of the input text-line image. Then a horizontal line is drawn at the location of the baseline as shown in Fig. 5(b).

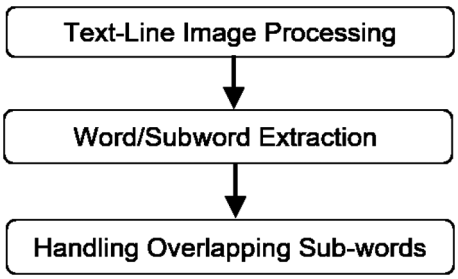

Fig. 4 Word/subword segmentation algorithm.

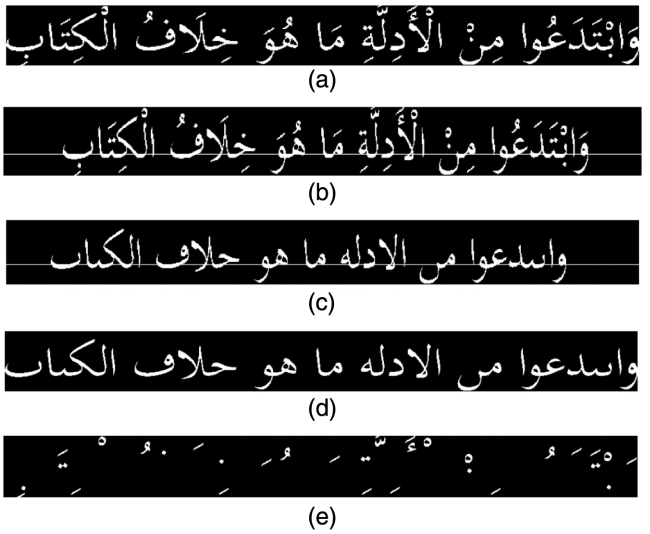

Fig. 5 An example of line processing step: (a) original line image, (b) the drawn line at the location of baseline, (c) main body line version with horizontal line, (d) main body line version image, and (e) diacritics and dotting line image.

To generate the first version of the input text-line image, we apply a connected component algorithm on the image obtained from the previous stage and select the largest component (all text that touches the baseline), which represents the body of the text without diacritics and dots as shown in Fig. 5(c). To remove the drawn horizontal line, we made a logical AND operation between the resulted image and the original text-line image. The second version of the line image is obtained by subtracting the original input text line image [Fig. 5(a)] from the generated first version [Fig. 5(d)] as shown in [Fig. 5(e)].

3.2.2 *Words/subwords extraction*

For the Arabic script, each word consists of one or more subwords. To extract words/subwords, a vertical projection is applied as an initial step to find the gap between them (where the projection equals zero). The challenge after this step is to decide if the gap is located between two consecutive words or between two subwords. In other words, what is the suitable threshold to determine the separation space between the subwords as an intraspace in the same word or a separation space between two distinct words? However, the gap between two consecutive words or subwords is not fixed and depends on the font type, size, and style. To handle this issue, first we compute the pen size, which is the pen thickness used for writing[9–11] of the current two consecutive words/subwords and compare it with the length of the separation space between the current two consecutive words/subwords. Calculating the pen size can handle by taking the most frequent value in the vertical projection applied for each subword. But taking the most frequent value from the vertical projection of some individual characters like "Aleph" "ا" gives a wrong estimation of the pen size. For this reason, the pen size is calculated by taking into account the most frequent value calculated from the horizontal projection. Thus if the most frequent value calculated from horizontal projection is greater than the most frequent value calculated from the vertical projection, then the pen size is assigned the most frequent value calculated from the vertical projection.

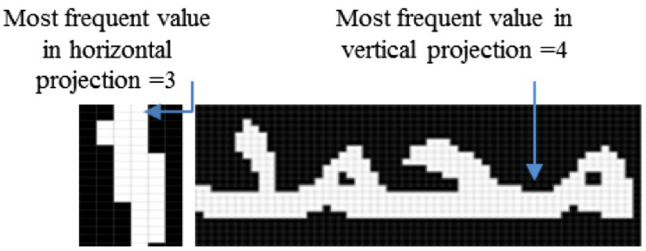

Fig. 6 Example of pen size calculation.

This means, if the subword consists of more than one character, the pen size is the thickness of the baseline and vice versa. Pen size calculation is formally defined as

$$\mathrm{PS} = \begin{cases} \mathrm{MFV(VP)}, & \max[\mathrm{HP(SW)}] > \max[\mathrm{VP(SW)}] \\ \mathrm{MFV(HP)}, & \text{otherwise} \end{cases}, \tag{1}$$

where PS represents pin size, SW represents subword, HP represents horizontal projection, VP represents vertical, and MFV represents the most frequent value. Figure 6 shows an example of pen size calculations for the two cases. In the example, there are two subwords, for the first one (left), the pen size is chosen as the most frequent value from the horizontal projection, whereas in the second subword, the pen size is chosen as the most frequent value from the vertical projection.

After calculating the pen size, the pen size is compared with the separation space. Therefore, if the separation space between two consecutive words/subwords is larger than the mean of the pen size of these two consecutive words/subwords, then the separation region performs a separation between two different words, else the separation region is between two subwords in the same word defined formally as

$$\mathrm{SR} = \begin{cases} \mathrm{WS}, & \mathrm{SS}[\mathrm{SW}(i), \mathrm{SW}(i+1)] > \frac{\mathrm{PS}(i)+\mathrm{PS}(i+1)}{2} \\ \mathrm{SWS}, & \text{otherwise} \end{cases}, \tag{2}$$

where SR represents separation region, WS represents word separation, SWS represents the subword separation, SS represents separation space, WS represents subword, and PS represents pin size. Figure 7 shows how to determine if two separated parts are related to the same word or different words. This figure shows that there are three separated parts ("بن", "حمد", and "ا") and the pen size of these parts are 3, 4, and 3, respectively. The separation space between the first

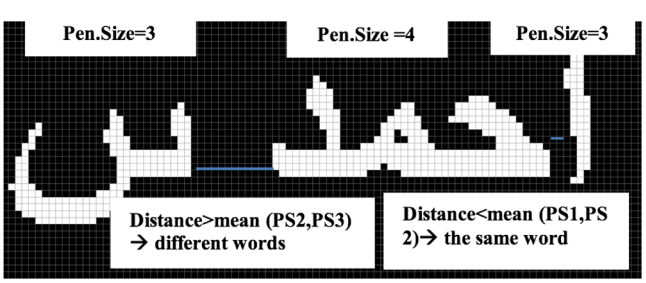

Fig. 7 Distance between two subwords in the same word and different words.

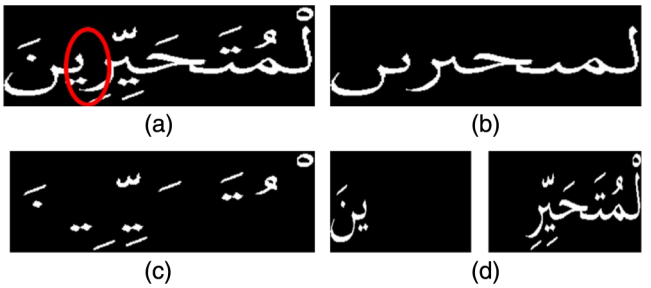

Fig. 8 Subwords overlapping example: (a) overlapped subwords, (b) two overlapped main bodies of subwords, (c) diacritics image of overlapped subwords, and (d) separated overlapped subwords with its diacritics.

part and the second part is less than the mean pen size of these parts, thus these two parts related to the same word. On the other hand, the separation space between the second part and the third part is larger than the mean pen size of these parts, thus these two parts related to different words.

3.2.3 *Handling overlapping subwords*

Arabic words might be composed of two or more not connected components. Identifying the subwords is important to treat some complex special characters like "س, SEEN." Subwords might be overlapped among each other. So using a vertical projection technique directly fails to extract them as well as it fails to attach dots and diacritics to its related subword. Since each subword is represented as a group of connected pixels, we can apply the connected components method on the abstracted version of the input word image (word without dots and diacritics) to extract the subwords images, where the number of the extracted connected components is equal to the number of subwords that the input word contains. Then the extracted connected components are ordered from right to left using the maximum column index (depending on Arabic script characteristic) of each connected component. To assign each diacritic to its related subword, all diacritics that are associated to the abstracted input word image are extracted by subtracting the subword image of the original binary input image [Fig. 8(a)] from the generated abstracted input word image [Fig. 8(c)]. Then we calculate the overlapping percentage for each diacritic regarding each overlapped subwords extracted from the previous step. The diacritics, Fig. 8(c), are assigned to the subword by calculating the overlapping between the considered diacritic and the considered subword. The diacritic is assigned for the subword that has a high overlapping ratio as shown in Fig. 8(d). If the subword consists of one connected component then no overlapping is detected, so all diacritics in the diacritics image version are related to it. We compute the overlapping ratio using the following equation:

$$\text{overlapping\_ratio} = \frac{\text{overlapping\_area\_size}}{\text{diacritic\_size}}, \tag{3}$$

where overlapping area size is equal to the intersection area between the diacritic and the subword at the column level. It is worth mentioning that while assigning the diacritics to the related subword, the generated images' sizes should be equal to the original image size in case of overlapped subwords. So the indices that have been obtained from the extraction of the connected components can be used again so the indices that have been obtained from the extraction of the connected component algorithm can be applied a gain to the new image because all image operation are based on the reference image. Algorithm 2 shows the algorithm for word/subwords segmentation method.

Algorithm 2 Words/subwords segmentation.

1. **INPUT**: **DBLI** Diacritics Binary Line Image, **MBLI** Main Binary Line Image
2. *Vertical Projection VP* ← [];
3. *Separation Indices SI* ← [];
4. *Separation Regions SR* ← [];
5. *Words/subWord List WSL* ← [];
6. *VP*←*verticalProjection*(*MBLI*);
7. *SI*←*returenIndiciesOfZeroValue*(*VP*);
8. *SR*←*mergeContinuousIndices*(*SI*);
9. **for** *i*←2 **to** *length*(*SR*); **do**
10. *SR*(*i*).*minColumnIndex*←*max*[*SR*(*i*)];
11. *SR*(*i*).*maxColumnIndex* ← *min*[*SR*(*i*+1)];
12. *SR*(*i*).*length*←*max*(*SR*(*i*)−*min*[*SR*(*i*+1)];
13. **if** {*SR*(*i*).*length*>*mean*[*SR*(*i*).*penSize*(),*SR*(*i*+1).*penSize*()]} **then**
14. *SR*(*i*).*segmentType*←*wordSegment*;
15. **else**
16. *SR*(*i*).*segmentType*←*subWordSegment*
17. **end if**
18. *diacriticsSegment*←*extractSegment*{*DBLI*,*min*[*SR*(*i*)], *max*[*SR*(*i*)]};
19. *bodySegment*←*extractSegment*{*MBLI*,*min*[*SR*(*i*)], *max*[*SR*(*i*)]};
20. *WSL*←*overlapping*(*diacriticsSegment*,*bodySegment*);
21. **end for**
22. **OUTPUT**: WSL

3.3 *Character Segmentation*

The proposed algorithm for character segmentation is based on contour extraction technique. The algorithm consists of four main stages shown in Fig. 9. The algorithm takes a binary word/subword image of printed Arabic text with/without diacritics as an input and return segmented words/subwords images as an output. Below is the detailed description of each step.

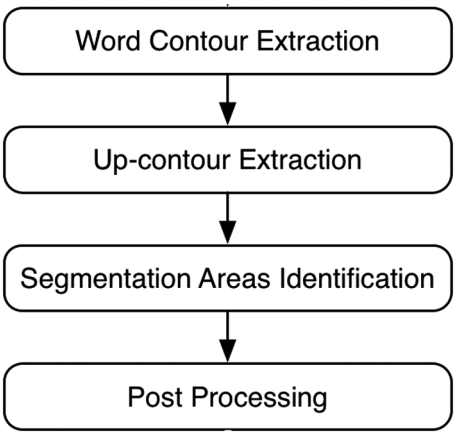

Fig. 9 Character segmentation algorithm.

3.3.1 *Word/subword contour extraction*

Contour-based segmentation technique gives a clear description of the word characters shape. This method facilitates determining the right segmentation points. Many methods have been tested to extract the contour of the abstracted word/subword image. The best results can be obtained when using the contour extraction method implemented in the MATLAB environment[49] named as `imcontour()` function. `imcontour()` is a MATLAB API from the MATLAB image processing toolbox used to draw a contour plot of the grayscale image. We exploit this function with some enhancements added to it by filling the holes in the main body of the abstracted word/subword image before applying the `imcontour` function. This MATLAB function gives better results especially for small font sizes. However, extracting the contour by a sequence of morphological operations raises some problems like the connection between contour parts. In addition, part of the word/subword sometimes is removed, which causes major problems for the next stages. Thus a morphological closing operation is applied to fill these gaps without affecting the details of small font sizes and thus the structure of some characters like "س, SEEN" is not removed. Figure 10 shows an example of the contour extraction stage.

3.3.2 *Upcontour extraction*

In this step, two points from the resulting contour are elected as start and end points. These points are used to search and retrieve the path that forms the up contour.

To determine the first and the last parts of the word/subword's main body, first, a threshold value is determined according to the average length of the character for the currently used pen size. Therefore, the first part will be located between the maximum column index of the subword contour and second column, which is determined by subtracting the previous threshold value from the max column index. The last part is determined in the same manner, but this time

Fig. 10 Contour extraction example.

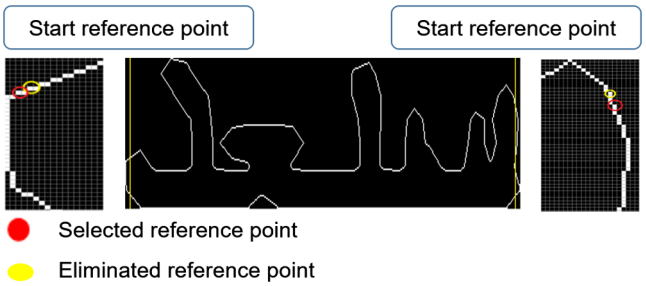

Fig. 11 Elected start and end points of the word contour.

the region will be located between the first column index in the contour and the second column, which is assigned by adding the previous threshold to the first column index value. To elect the start point, the two pixels that have the minimum row index values are located, and then the pixel with maximum column index is elected. The same for the end point election, the two pixels that have the minimum row index values are located, then the pixel with minimum column index is elected.[50] Figure 11 shows the first and last parts in the word contour along with the selected start and end points.

To extract the up-contour image, a tracing operation is started from the elected start point pixel until reaching the elected end point pixel. The resulted path is determined by moving from the start point pixel in a counterclockwise direction using the eight-neighboring connectivity, which is used to check different possibilities for the next pixel location. Each time the movement action is taken a place, the previous point is tracked to overcome forming a loop. The tracing operation ends when the selected end point in the contour is reached, resulting in a new image having a size similar to word/subword contour.

3.3.3 *Segmentation areas identification*

In this step, the up-contour as shown in Fig. 12 is examined to identify the segmentation (splitting) areas. The process starts by scanning the up-contour row by row where the continuous regions that have the same rows index and consequent columns index are extracted. In addition, these regions have a local minima region to consider as a cutting region, and this condition satisfied when the pixels above the starting and the ending pixels of the continuous regions are equal to 1 (pixels that have data in binary image). After that, the first and last points in the region are considered as a splitting area reference points. Figure 12 shows the splitting areas and their related reference points.

After identifying the splitting areas, each character will locate between two consecutive splitting regions. Figure 13 shows the cutting points over splitting regions for some regular characters. However, there are some special cases where

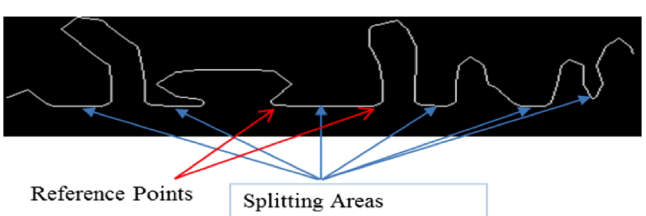

Fig. 12 Up-contour splitting areas with reference points.

Fig. 13 Splitting regions for some regular characters.

the splitting area locates within a character like "ص SAAD," "ض DAD," "س, SEEN," "ش SHEEN," "ط TAA," and "ظ THAA." In addition, some splitting areas locate within character when the position of the character is at the end of the word or exists independently in the text like "ب BAA" and "ي YAA." So a postprocessing step for character segmentation is necessary to ignore these spatial splitting areas.

3.3.4 *Postprocessing*

In the postprocessing step, character segmentation algorithm performs two major steps, which are detailed in Algorithm 3. In the first step, the segment is defined as a part of the up-contour that locates between two splitting areas. The up-contour part might represent a full character or a part of a character. Thus further checking for some requirements is needed as a second step to decide whether this part must be merged or to consider it as a standalone character. As an Example of further checking to decide whether the part must be merged or to consider it as a standalone, the last part "ش SHEEN" character, the three parts in "س, SEEN" character, the second part in "ض DAD" and "ص SAAD" characters. The requirements that must satisfy are summarized in:

- *Euler number*. This is one of the region properties that can be estimated. It must be equal to or greater than one, meaning that no holes exist in this part.
- *Character's part height*. Based on experimental results, the height should be less than the pen size multiplied by two.
- The part has no dots above or below.

Figure 14(a) shows an example of a segment that satisfies the requirements, which should be merged later. So if the segment satisfies the requirements, then checking for "ش SHEEN" character case is taken a place. Thus if the segment is part of SHEEN character (only the last part of the SHEEN character satisfies the conditions), then the previous two splitting areas are ignored. This means that the segment is merged with the two previous segments to form the SHEEN character. If it is not the SHEEN character, then the space counter is incremented by one.

If the character's part does not satisfy the conditions, then two cases should be handled by checking the space count and the location of the segment with respect to its position in the word. In the first case, the count of spaces is checked. Thus if the space count equals one, then the segment is merged with previous one to form the "ط TAA," "ظ THAA," "ض DAD," or "ص SAD" characters. Otherwise, if the counting space equals two, in this case the segment is merged with previous two parts to perform the "س, SEEN" character. Figure 14(c) shows "س, SEEN" case detection when the space counter equals three (three consecutive

Algorithm 3 Character segmentation postprocessing.

```
1.  Input: LSR List Splitting Regions
2.  Result: Each splitting area is assigned by a flag to be ignored or
        not
3.  Vertical Projection VP ← [];
4.  for i←1 to length(LSR) do
5.    if LSR(i).locatesAbove(baseline) then
6.      LSR(i).ignore←true;
7.      return;
8.    end if
9.    CharacterSegment←line.extract(LSR(i),LSR(i+1);
10.   if (CharacterSegment.eulerNumber() >= 1
11.   and CharacterSegment.hasDot() and
      CharacterSegment.height() < 2*penSize) then
12.     if CharacterSegment is sheen then
13.       LSR(i–1).ignore←true;
14.       LSR(i–2).ignore←true;
15.     else
16.       spaceCounter++;
17.     end if
18.   else if LSR(i).isLastRegion() then
19.     if SLR(i).locateUnderBaseline() then
20.       SLR(i).ignore←true;
21.       if CharacterSegment is sheen then
22.         LSR(i–1).ignore←true;
23.         LSR(i–2).ignore←true;
24.       else if spaceCounter == 2 then
25.         LSR(i–1).ignore←true;
26.         LSR(i–2).ignore←true;
27.       else if spaceCounter == 1 then
28.         LSR(i–1).ignore←true;
29.       end if
30.     else if spaceCounter == 1 then
31.       LSR(i–2).ignore←true;
32.       LSR(i–3).ignore←true;
33.     else if spaceCounter == 1 then
34.       LSR(i–2).ignore←true;
35.     end if
36.   end if
37. end for
38. OUTPUT:LSR
```

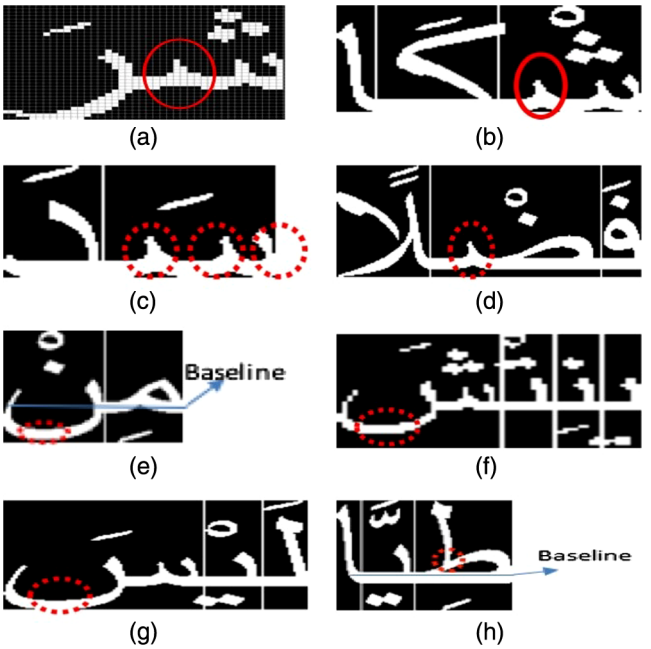

Fig. 14 Character segmentation postprocessing example: (a) segment that satisfies the requirements, (b) SHEEN character case, (c) space counter = 3, SEEN case, (d) space counter =1, DAD case, (e) last splitting region below the baseline, (f) merging the last segment with previous one, (g) merging the last segment with previous one, and (h) splitting region located above the baseline.

segments satisfy the condition) and Fig. 14(d) shows "ص DAD" cases detection when the space counter equals one (one segment satisfies the conditions).

In the second case, if the segment is located within the last character of the word as in Fig. 14(d) and the last splitting region is below the baseline, then the splitting region is ignored. Figure 14(e) shows the last splitting area that should be ignored. Indeed, if the ignored segment has no dots, this means that the segment is not a full character and must be merged with some segments before like in "س, SEEN," "ش SHEEN," "ص SAD," and "ض DAD" characters. So an additional checking must be performed to detect these cases. Thus if the space counter equals one, then "ش SHEEN," "ص SAD," or "ض DAD" cases are detected and the segment must be merged with the previous part. Otherwise, if the space counter equals two then a "س, SEEN" case is detected and the segment must be merged with the previous two parts. Figures 14(e) and 14(f) show merging the last segment with previous part to perform one character according to the above checking case.

Finally, for "ط TAA" and "ظ THAA" characters, the splitting region locates above the baseline. So if this condition is satisfied then the splitting region is ignored as shown in Fig. 14(g). Algorithm 3 summarizes the procedure of character postsegmentation. Figure 15 shows an example of the character segmentation algorithm. Generally, in segmentation-based approach, the main segmentation problems can be classified into over segmentation and under segmentation. In Arabic, the over segmentation can occur mainly in the following characters {س, ش, ص, and ض} and specifically with small font size (e.g., <8) and with low-resolution images. In our method, we efficiently handle this issue especially for moderate and larger font size (>8). On the other hand, the problem showed in segment "التي," in Fig. 15, is called under segmentation, and it is hard to segment in most Arabic character segmentation methods. Indeed, most Arabic OCR systems treated ligatures as new classes since segmenting it is very hard because the characters overlapped vertically and do not touch each other, which leads to an unclear segmentation point.

4 Results and Evaluations

4.1 *Dataset Description*

The available datasets such as INF/ENIF,[51] IFHCDB,[52] and APTI[53] that have been used in evaluating character segmentation methods are not suitable for our target experiments. They are dedicated to Arabic text without diacritics, whereas our method has been designed for Arabic text containing diacritics.

Thus we manually built a new dataset from real scanned documents to experiment with our method. To make the dataset generic and comprehensive, the collected dataset includes text content from different sources (e.g., books, magazines, reports, and papers) and topics (e.g., religious, sport, and poetry texts), in addition to a considerable

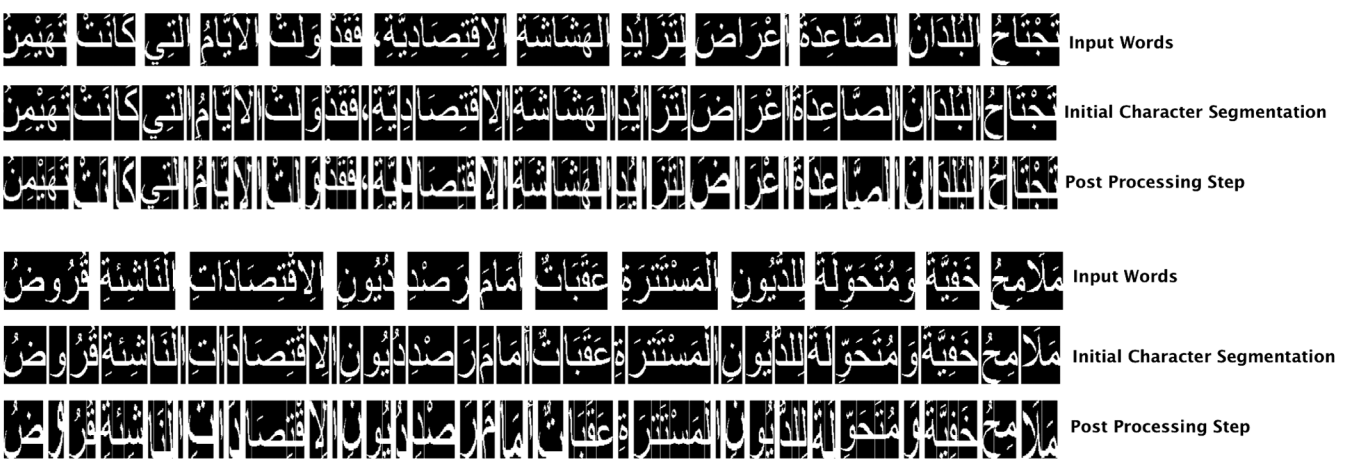

Fig. 15 Example of initial segmentation and postprocessing outputs for two different font types.

variation at font type, size, and style levels. The dataset is available in Ref. 54. We tested the algorithm on scanned images at 300 dpi with multiple font sizes between 8 and 24, and styles including regular, bold, and italic. In addition, we used APTI dataset in order to evaluate and compare the performance of the purpose method with respect to the performance of other related approaches. APTI dataset is a large-scale benchmark for recognition systems in Arabic. It has a variability in the generation procedure of text images including different font types, sizes, and styles. In addition, it includes very large vocabulary, various forms of ligatures, overlaps of characters, and variability of the height of each word image.

To sum up, since the collected texts in the APTI dataset did not contain diacritics, it is not suitable to evaluate the performance of the proposed algorithm in segmenting Arabic text with the existence of diacritics. Therefore, we evaluated our method on our own collected dataset, which includes Arabic text with diacritics along with variation of font types, styles, and size. On the other hand, to compare our proposed method with other related works, we used APTI dataset, which is a standard published benchmark and some related works used it for the purpose of testing.

4.2 *Results*

This section presents the results of the conducted experiments on the generated set of Arabic documents and using APTI dataset. As an implementation platform, we used MATLAB in implementing and then experimenting our proposed character segmentation algorithm where such a commonly known high-level technical computing platform provides well-implemented toolbox in image processing. The performance for segmentation is measured in terms of character segmentation rate, which is computed by the ratio of the number of characters that are correctly segmented to the total number of characters; in this metric, the ligature is considered as one character; also the last character with no splitting area is not considered.

First, the proposed algorithms (line, word, and character segmentation) were experimented and evaluated using our manually created dataset since it includes diacritics. The proposed line segmentation methods were experimented on 43,055 lines and reported excellent results in terms of line segmentation ratio, which computed "the total number of correctly segmented lines over the total number of input lines" with an average of 99.5%. Table 1 below shows the results generated through the testing process over variation of font type, style, and size.

The results of the words segmentation stage in terms of word segmentation ratio are reported in Table 2. The proposed word segmentation methods are experimented on about 1500 lines of (23,350 words) with five font types (advertising bold, simplified Arabic, Arial, traditional

Table 1 Line segmentation results for different font styles and types on text with diacritics.

Font	Font type	Total number of input lines	No. of correctly segmented lines	Accuracy (%)
Plain	Advertising bold	3000	2982	99.4
	Diwani	3011	2988	99.2
	Andalus	2520	2499	99.2
	Arabic transparent	2940	2922	99.4
	Naskh	2982	2960	99.3
Bold	Advertising bold	3000	2982	99.4
	Diwani	3011	2988	99.2
	Andalus	2520	2499	99.2
	Arabic transparent	2940	2922	99.4
	Naskh	2982	2960	99.3
Italic	Advertising bold	3034	3020	99.5
	Diwani	3036	3024	99.6
	Andalus	2438	2420	99.3
	Arabic transparent	3051	3038	99.6
	Naskh	3118	3102	99.5
	Total	43,055	43,271	99.5

Table 2 Word segmentation results for different font styles, types, and size between 8 and 24.

Font style	Font type	Total numbers of input words	No. of correctly segmented words	Accuracy (%)
Plain	Simplified Arabic	1572	1564	99.5
	Times New Roman	1554	1544	99.4
	Arial	1550	1541	99.4
	Advertising bold	1566	1333	85.1
	Arabic transparent	1548	1542	99.6
Bold	Simplified Arabic	1571	1563	99.5
	Times New Roman	1556	1550	99.6
	Arial	1549	1543	99.6
	Advertising bold	1546	1336	86.4
	Arabic transparent	1548	1540	99.5
Italic	Simplified Arabic	1572	1558	99.1
	Times New Roman	1554	1538	99
	Arial	1550	1534	99
	Advertising bold	1566	1324	84.5
	Arabic transparent	1548	1534	99.1
	Total	23,350	22,544	96.5

Arabic, and Times New Roman), three styles (plain, italic, and bold) and eight font sizes (8, 9, 10, 12, 14, 16, 18, and 24 points), with an average accuracy of 96.5%. The results show that the algorithm has almost the same performance when varying the font style, type, and size. Also we experimented the character segmentation stage on different font type, style, and size on about 38,763 words. Table 3 shows the performance of the proposed algorithm with an average accuracy of 98.5%. Note that the accuracy was computed as "the total number of correctly segmented characters over the total number of input characters." In this metric, the ligature is considered as one character; also the last character with no splitting area is not considered. The results show that the algorithm has almost the same performance when varying the font style, type, and size. For APTI dataset, the proposed line, word, and character segmentation methods were experimented on the same variation on font types, size, and style with an average accuracy of 99.9% for line segmentation, 98.1% for word segmentation, and 98.2% for character segmentation.

Table 4 shows our results compared with that of previous related works. Note that there is no baseline to compare with because there is no published standard dataset for Arabic text with diacritics to test with. Therefore, most of the authors tested their work on their own collected data and did not make it public. Indeed, it is fairer to compare with using the same dataset. However, the implementation code of the proposed methods in the related works is not available and sometimes it is difficult to rewrite the code since it may depend on parameters, hypothesis, tools, and APIs that are not mentioned clearly in the published paper. Therefore, we tested our method using APTI dataset in order to make some fair comparisons with other related works. As shown from this table, the proposed algorithm outperforms other related works in terms of: (i) experimenting on different font type, size, and style, (ii) handling diacritics, (iii) and in terms of average accuracy.

Table 3 Character segmentation results for different font styles, types, and size between 8 and 24.

Font	Font type	No. of input characters	No. of correctly segmented characters	Accuracy (%)
Regular	Simplified Arabic	2790	2835	98.4
	Traditional Arabic	2935	2976	98.6
	Times New Roman	2812	2844	98.9
	Arial	2585	2620	98.7
	Advertising bold	2058	2084	98.8
Bold	Simplified Arabic	2550	2599	98.1
	Traditional Arabic	2587	2643	97.9
	Times New Roman	2841	2873	98.9
	Arial	2615	2646	98.8
	Advertising bold	2615	2491	98.9
Italic	Simplified Arabic	2549	2606	97.8
	Traditional Arabic	2584	2629	98.2
	Times New Roman	2484	2516	98.7
	Arial	2454	2490	98.6
	Advertising bold	2455	2491	98.6
	Total	39,353	38,763	98.5

5 Conclusion

In this paper, an efficient offline contour-based character segmentation algorithm for printed Arabic text is proposed based on contour extraction segmentation technique. Our algorithm is able to segment characters of words consisting of diacritics. Also the algorithm can handle some special complex cases occurring because of the over-segmentation problem. We experimented the algorithm on dataset collected and built manually, with making different versions of the documents in terms of style, size, type, and font variation. The experimental results show the reliability of our algorithm in segmenting correctly more than 38,700 out of 39,353 words.

Segmentation of Arabic text is error-prone. It is the stage where most of the errors occur and where the error in segmentation will result in classification errors. In this paper, a new scheme is investigated and developed such that the segmentation is done in such a way to minimize errors and maximize the recognition rate. Different algorithms are proposed for different segmentation stages (a word/subword and diacritics segmentation and character segmentation). Results using the proposed scheme show that promising results of the Arabic character are achieved. Also an enhanced method for word, subword, and diacritics segmentation is proposed, the subwords are extracted in two ways according to the subwords situation, vertical projection is used in the case of full separation between subwords by finding the gaps between them, the concept of connected component concept is used to find the subwords in case of overlapping, the connected components concept is also used to extract the diacritics, the proposed method also determines if the subwords are related to the same word or to different words regardless of the font type or size by estimating the pen size for each subword, the algorithm shows promising results. For character segmentation stage, an enhanced algorithm is proposed, based on contour extraction technique, which has many advantages over other methods like having a clear description of character shape and details even for small fonts, also the errors in extracting the baseline is eliminated since there is no need to adjust the baseline many times. A postprocessing step is needed to solve over segmentation problem; ignore cases checking algorithm is developed in an easy and reliable way that can fit many font types and styles, character segmentation algorithm shows good results up to 98.5%.

As a future work, our plan is to reduce the under-segmentation problem as much as possible specifically for small font size. Also to enhance the rules, it can be generalized to handle more complex fonts. In addition, we want to extend the work to extract diacritic from the characters to facilitate the recognition stage.

Table 4 Comparing with other related work.

	Year	Segmentation method	Dataset	Font variation			Handling diacritics	Average accuracy (%)
				Type	Size	Style		
Zheng[19]	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	No	94.8
Javed[4]	2010	Pattern matching techniques	A total of 1282 unique ligatures are extracted from the 5000 high frequency words in a corpus-based dictionary	Noori Nastalique font	36	Plain	No	92
Saabni[55]	2014	Partial segmentation and Hausdorff distance	APTI	Different fonts to cover different complexity of shapes of Arabic printed characters	10 different sizes	Plain	No	96.8
Anwar et al.[35]	2015	Projection-based	127 sentences composed of 1061 letters	Traditional Arabic	70	Plain	Yes	97.5
Marwa et al.[38]	2016	Histogram and contextual properties	APTI	Different font types	Different sizes	Plain, italic, and bold	No	85.6
Radwan[37]	2016	Multichannel neural networks	APTI	Arial, Tahoma, Thuluth, and Damas	18	Plain	No	95.5
Fitriyatul et al.[39]	2017	Interests points, contour-based	10 lines of 30 subwords	Not reported	Not reported	Plain	No	86.5
Fakhry et al.[40]	2017	Connected component	5 lines 15 words	Not reported	Not reported	Plain	Yes	80.2
Marwa et al.[41]	2017	Projection profile, SVM	APTI	Advertising bold	6,8,10, 12	Plain, italic, and bold	No	98.24
Abdelhay et al.[38]	2018	Contour-based and template matching	83 lines of 984 words	34 different fonts	Different font sizes	Plain	No	94.7
Our approach	2019	Contour-based method	1500 lines of (23,350 words)	Advertising bold, simplified Arabic, Arial, traditional Arabic, and Times New Roman	8, 9, 10, 12, 14, 16, 18 and 24	Plain, italic, and bold	Yes	98.5
Our approach	2019	Contour-based method	APTI	Advertising bold, simplified Arabic, Arial, traditional Arabic, and Times New Roman	8, 9, 10,12, 14, 16, 18 and 24	Pain, italic, and bold	Yes	98.2

References

1. N. at Accredited Language, "The 10 most common languages," *Blog* (2019).
2. A. Lawgali, "A survey on Arabic character recognition," *Int. J. Signal Process. Image Process. Pattern Recognit.* **8**(2), 401–426 (2015).
3. G. Meijer, *Smart Sensor Systems*, Wiley, Delft, The Netherlands (2008).
4. S. T. Javed et al., "Segmentation free Nastalique Urdu OCR," *World Acad. Sci. Eng. Technol.* **4**(10), 456–461 (2010).
5. T. Islam, S. C. Mukhopadhyay, and N. K. Suryadevara, "Smart sensors and Internet of Things: a postgraduate paper," *IEEE Sensors J.* **17**(3), 577–584 (2017).
6. L. M. Lorigo and V. Govindaraju, "Offline Arabic handwriting recognition: a survey," *IEEE Trans. Pattern Anal. Mach. Intell.* **28**(5), 712–724 (2006).
7. N. Sabbour and F. Shafait, "A segmentation-free approach to Arabic and Urdu OCR," *Proc. SPIE* **8658**, 86580N (2013).
8. S. Naz et al., "An OCR system for printed Nasta'liq script: a segmentation based approach," in *IEEE 17th Int. Multi-Top. Conf. (INMIC)*, IEEE, pp. 255–259 (2014).
9. P. Ahmed and Y. Al-Ohali, "Arabic character recognition: progress and challenges," *J. King Saud Univ.-Comput. Inf. Sci.* **12**, 85–116 (2000).
10. A. M. Zeki and M. S. Zakaria, "Challenges in recognizing Arabic characters," International Islamic University Malaysia (IIUM), Kuala Lumpur, Malaysia, National University of Malaysia (UKM), Bangi, Selangor, Malaysia (2004).
11. A. Mahmood, "Arabic & Urdu text segmentation challenges and techniques," *Int. J. Comput. Sci. Technol.* **4**, 32–34 (2013).
12. Y. M. Alginahi, "A survey on Arabic character segmentation," *Int. J. Doc. Anal. Recognit.* **16**(2), 105–126 (2013).
13. A. Cheung, M. Bennamoun, and N. W. Bergmann, "An Arabic optical character recognition system using recognition-based segmentation," *Pattern Recognit.* **34**(2), 215–233 (2001).
14. J. H. AlKhateeb et al., "Component-based segmentation of words from handwritten Arabic text," *Int. J. Comput. Syst. Sci. Eng.* **5**(1), 54–58 (2009).
15. N. A. Shaikh, G. A. Mallah, and Z. A. Shaikh, "Character segmentation of Sindhi, an Arabic style scripting language, using height profile vector," *Aust. J. Basic Appl. Sci.* **3**(4), 4160–4169 (2009).
16. I. Aljarrah et al., "Automated system for Arabic optical character recognition with lookup dictionary," *J. Emerg. Technol. Web Intell.* **4**(4), 362–370 (2012).
17. M. M. Alipour, "A new approach to segmentation of Persian cursive script based on adjustment the fragments," *Int. J. Comput. Appl.* **64**(11), 21–26 (2013).
18. S. N. Nawaz et al., "An approach to offline Arabic character recognition using neural networks," in *10th IEEE Int. Conf. Electron., Circuits and Syst. (ICECS)*, IEEE, Vol. 3, pp. 1328–1331 (2003).
19. L. Zheng, A. H. Hassin, and X. Tang, "A new algorithm for machine printed Arabic character segmentation," *Pattern Recognit. Lett.* **25**(15), 1723–1729 (2004).
20. A. Zidouri et al., "Adaptive dissection based subword segmentation of printed Arabic text," in *Ninth Int. Conf. Inf. Visualisation (IV)*, IEEE, pp. 239–243 (2005).
21. J. Ahmad, "Optical character recognition system for Arabic text using cursive multi-directional approach," *J. Comput. Sci.* **3**, 549–555 (2007).
22. M. Omidyeganeh et al., "A new segmentation technique for multi font Farsi/Arabic texts," in *IEEE Int. Conf. Acoust., Speech, and Signal Process.*, IEEE, Vol. 2 (2005).
23. T. Sari, L. Souici, and M. Sellami, "Off-line handwritten Arabic character segmentation algorithm: ACSA," in *Proc. Eighth Int. Workshop Front. Handwriting Recognit.*, IEEE, pp. 452–457 (2002).
24. R. Mehran, H. Pirsiavash, and F. Razzazi, "A front-end OCR for Omnifont Persian/Arabic cursive printed documents," in *Digital Image Computing: Techniques and Applications (DICTA)*, IEEE, pp. 56–56 (2005).
25. B. Bushofa and M. Spann, "Segmentation of Arabic characters using their contour information," in *Proc. 13th Int. Conf. Digital Signal Process.*, IEEE, Vol. 2, pp. 683–686 (1997).
26. K. Romeo-Pakker, H. Miled, and Y. Lecourtier, "A new approach for Latin/Arabic character segmentation," in *Proc. 3rd Int. Conf. Doc. Anal. and Recognit.*, IEEE, Vol. 2, pp. 874–877 (1995).
27. M. M. Altuwaijri and M. A. Bayoumi, "A thinning algorithm for Arabic characters using art2 neural network," *IEEE Trans. Circuits Syst. II* **45**(2), 260–264 (1998).
28. D. Motawa, A. Amin, and R. Sabourin, "Segmentation of Arabic cursive script," in *Proc. Fourth Int. Conf. Doc. Anal. and Recognit.*, pp. 625–628 (1997).
29. B. Al-Badr and R. M. Haralick, "Segmentation-free word recognition with application to Arabic," in *Proc. 3rd Int. Conf. Doc. Anal. and Recognit.*, IEEE, Vol. 1, pp. 355–359 (1995).
30. B. Bushofa and M. Spann, "Segmentation and recognition of Arabic characters by structural classification," *Image Vision Comput.* **15**(3), 167–179 (1997).
31. Y. Zhang, Z. Q. Zha, and L. F. Bai, "A license plate character segmentation method based on character contour and template matching," *Appl. Mech. Mater.* **333**, 974–979 (2013).
32. A. Amin, "Recognition of Arabic handprinted mathematical formulas," *Arabian J. Sci. Eng.* **16**(4), 531–542 (1991).
33. M. Bennamoun and B. Boashash, "A structural-description-based vision system for automatic object recognition," *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **27**(6), 893–906 (1997).
34. M. Mostafa, "An adaptive algorithm for the automatic segmentation of printed Arabic text," in *17th Natl. Comput. Conf.*, International Society for Optics and Photonics, Saudi Arabia, pp. 437–444 (2004).
35. K. Anwar, Adiwijaya, and H. Nugroho, "A segmentation scheme of Arabic words with Harakat," in *IEEE Int. Conf. Commun., Networks and Satell. (COMNESTAT)*, pp. 111–114 (2015).
36. M. A. A. Mousa, M. S. Sayed, and M. I. Abdalla, "Arabic character segmentation using projection-based approach with profile's amplitude filter," arXiv 1707.00800 (2013).
37. M. A. Radwan, M. I. Khalil, and H. M. Abbas, "Predictive segmentation using multichannel neural networks in Arabic OCR system," *Lect. Notes Comput. Sci.* **9896**, 233–245 (2016).
38. M. Amara et al., "New rules to enhance the performances of histogram projection for segmenting small-sized Arabic words," in *Int. Conf. Hybrid Intell. Syst.* (2016).
39. F. Qomariyah, F. Utaminingrum, and W. F. Mahmudy, "The segmentation of printed Arabic characters based on interest point," *J. Telecommun. Electron. Comput. Eng.* **9**(2–8), 19–24 (2017).
40. F. I. Firdaus, A. Khumaini, and F. Utaminingrum, "Arabic letter segmentation using modified connected component labeling," in *Int. Conf. Sustainable Inf. Eng. and Technol. (SIET)*, pp. 392–397 (2017).
41. M. Amara, K. Zidi, and K. Ghedira, "An efficient and flexible knowledge-based Arabic text segmentation approach," *Int. J. Computer Sci. Inform. Security* **15**(7), 25–35 (2017).
42. A. Zoizou et al., "A new hybrid method for Arabic multi-font text segmentation, and a reference corpus construction," *J. King Saud Univ.* (2018).
43. S. Naz et al., "Urdu Nasta'liq text recognition using implicit segmentation based on multi-dimensional long short term memory neural networks," *SpringerPlus* **5**(1), 2010 (2016).
44. F. Nashwan et al., "A holistic technique for an Arabic OCR system," *J. Imaging* **4**(1), 6 (2018).
45. I. U. Din et al., "Segmentation-free optical character recognition for printed Urdu text," *EURASIP J. Image Video Process.* **2017**(1), 62 (2017).
46. S. Rawls et al., "Combining deep learning and language modeling for segmentation-free OCR from raw pixels," in *1st Int. Workshop Arabic Script Anal. and Recognit. (ASAR)*, pp. 119–123 (2017).
47. B. Su and S. Lu, "Accurate recognition of words in scenes without character segmentation using recurrent neural network," *Pattern Recognit.* **63**, 397–405 (2017).
48. M. Namysl and I. Konya, "Efficient, lexicon-free OCR using deep learning," (2019).
49. MATLAB, Filled 2-D contour plot, https://www.mathworks.com/help/matlab/ref/contourf.html (2010).
50. K. Mohammad et al., "Printed Arabic optical character segmentation," *Proc. SPIE* **9399**, 939911 (2015).
51. M. Pechwitz et al., "IFN/ENIT-database of handwritten Arabic words," in *Proc. CIFED*, Vol. 2, pp. 127–136, Citeseer (2002).
52. S. Mozaffari et al., "A comprehensive isolated Farsi/Arabic character database for handwritten OCR research," in *Tenth Int. Workshop Front. Handwriting Recognit.*, Suvisoft (2006).
53. F. Slimane et al., "A new Arabic printed text image database and evaluation protocols," in *10th Int. Conf. Doc. Anal. and Recognit.*, IEEE, pp. 946–950 (2009).
54. K. Mohammad, "BZU OCR research group," http://sites.birzeit.edu/bzuocr/data-sets.
55. R. Saabni, "Efficient recognition of machine printed Arabic text using partial segmentation and Hausdorff distance," in *6th Int. Conf. Soft Comput. and Pattern Recognit. (SoCPaR)*, pp. 284–289 (2014).

Khader Mohammad is currently working as an assistance professor in the Engineering and Technology collage at Birzeit University, where he teaches graduate and undergraduate level courses in hardware design, computer vision, system-on-chip design and technical leadership. His current interests include research on the broad areas of soc-chip design and verification, VLSI design, image processing, computer, multimedia, mobile imaging, image forensics, and methodologies to improve design and verification productivity in system design.

Aziz Qaroush received his bachelor's and master's degrees in computer engineering from Jordan University of Science and Technology, Irbid, Jordan, in 2003 and 2006, respectively. Currently, he is a lecturer with the Department of Electrical and Computer Engineering, Birzeit University, Birzeit, Palestine. His research interests include

machine learning, image processing and computer vision. He published many articles in these areas.

Muna Ayesh graduated with a master's degree from Birzeit University.

Mahdi Washha received his bachelor's degree in computer systems engineering, with high distinction, from University of Birzeit in 2012. He completed a master's degree in artificial intelligence and robotics in the Department of Information Engineering, Computer Science, and Statistics at the University of Roma La Sapienza. Currently, he completed a PhD degree from IRIT.

Ahmad Alsadeh is an assistant professor in the Electrical and Computer Engineering Department of Birzeit University. He was awarded the degree of Doctor of Engineering in network security from the Hasso Plattner Institute at the University of Potsdam in Germany in October 2013. Prior to that, he received his bachelor's degree in electrical engineering with emphasis on telecommunication in 2002, and his master's degree in scientific computing from Birzeit University in 2007.

Sos Agaian is a distinguished professor of computer science at College of Staten Island and the Graduate Center, CUNY. Prior to joining the City University of New York, he was a Peter T. Flawn Professor of Electrical and Computer Engineering with the University of Texas at San Antonio.