https://doi.org/10.2352/ISSN.2470-1173.2017.13.IPAS-204

A Robust Line Segmentation Algorithm for Arabic Printed Text with Diacritics

Muna Ayesh, Khader Mohammad, and Aziz Qaroush; Department of Electrical and Computer Engineering of Birzeit University; Ramallah; Palestine
Sos Agaian; University Of Texas; San Antonio
Mahdi Washha; IRIT Laboratory; University of Toulouse ;Toulouse; France
email: ayyash.muna@gmail.com, khamadawwad@birzeit.edu, aqaroush@birzeit.edu, sos.agaian@utsa.edu, mahdi.washha@irit.fr

Abstract

Line segmentation performs a significant stage in the OCR systems; it has a direct effect on the character segmentation stage which affects the recognition rate. In this paper a robust algorithm is proposed for line segmentation for Arabic printed text system with and without diacritics based on finding the global maximum peak and the baseline detection. The algorithm is tested for different font sizes and types and results have been obtained from testing 5 types of fonts with total of 43,055 lines with 99.9 % accuracy for text without diacritics and 99.5% accuracy for text with diacritics.

Introduction

OCR stands for optical character recognition, which converts electronic scanned images into an editable format [1, 2, 3, 4]. The development of an efficient, accurate Arabic OCR systems has become one of the most important and challenging tasks, since it have been used in many applications related to the information retrieval process which has a big revolution recently, to the search engines and data entry operations which can be done in faster and more accurate way [5, 6, 7, 8].

Extracting text and recognizing it from an image in general can be divided into five fundamental steps as shown in figure 1 : image acquisition, preprocessing, segmentation, feature extraction and classification. Image acquisition is the first step in the character recognition process and defined as the action of retrieving an image from the source. According to the way of retrieving the image; OCR can be classified as online OCR systems in which the input is taken by pen writing on a flatbed like tablet computers, or offline OCR systems in which the input is usually an already stored image taken by camera or scanner.

The preprocessing step aims to produce a cleaned up version of the original text image that can be used directly and efficiently by the segmentation methods. The most popular methods used in the preprocessing step are contrast enhancement, binarization, filtering, and thinning [9, 10]. The third step in the OCR is the segmentation which can be classified as:

- **Segmentation-free systems (Holistic Approach):** In this approach, the text is segmented into only lines and words; no need to extract the characters or strokes, the recognition step is applied directly. This method usually uses a lookup dictionary contains all sub-words possibilities to do recognition so it suitable for limited number of words that perform an enumeration like the number and cities names.

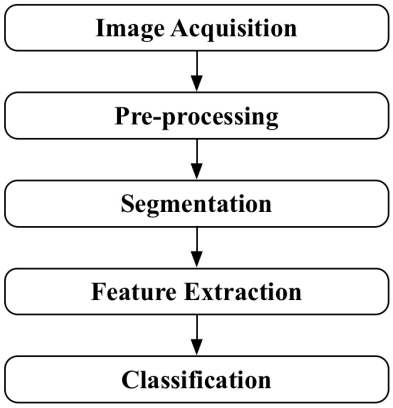

Figure 1. *OCR Development Process.*

- **Segmentation-based systems (Analytical Approach):** In this approach, the text is segmented into small parts that perform characters and strokes that the words or sub-words in the text consist of. This method requires more processing but it gives better results since it can cover most of characters possibilities, this makes the feature extraction and recognition steps easier, so this approach is widely used compared to the previous one.

The segments resulted from the segmentation stage are the input to the feature extraction stage. In this stage statistical or structural features are extracted from the segments to build the feature vector space [11, 4]. The final stage is the classification or recognition which uses the extracted feature to recognize the segments [11, 4]. Many methods can be used at this stage like neural networks, k-nearest- neighbor, and decision tree.

Segmentation stage and specifically line segmentation is a mandatory step in building any OCR systems. Thus, locating text lines in scanned printed Arabic documents is still not an easy task due to a set of characteristics of the Arabic language, which are summarized as following [12, 2, 13]:

- It has a cursive nature and written/read from right to left.
- The shape of character depends on its location in the word (at beginning, middle, end, or a standalone character).
- The diacritics appear above or below the word which are called sounds or short vowels.
- Each character has different width and height from other characters.
- If the characters overlapped vertically they are perform a ligature or combined character.

The variation of the height and the existence of diacritics which are widely used in holy books produce additional challenges to line segmentations process since most of the work on the line segmentation are based on horizontal projection methods to find the zero crossing area between consecutive lines. In this paper, a novel line segmentation approach for machine printed Arabic documents is proposed. The proposed approach can efficiently segment documents containing text with diacritics independent of the fonts types, sizes, and styles. Our method in segmentation is a top down baseline approach, meaning that the baseline location of each line text in document is determined to find correct segmentation points. The main difference of the proposed method with existed ones is that the proposed method handles the overlapping between consecutive lines due to the existence of diacritics and works independent of font type, style and size. On the other hand, the proposed approach does not address or handle image quality enhancement or character degradation.

Related Work

Line segmentation performs a significant stage in the OCR systems; it has a direct effect on the next stages word, sub-word, and character segmentation which affect the recognition rate directly. Many methods are proposed for line segmentation. These methods can be classified as projection profile method, smearing method, grouping based method, bounding based method, Hough transform method, and thinning based method. Projection profiles methods: Two main types of projection profiles are used; horizontal projection profiles and vertical Projection profiles [14, 12, 15], horizontal projection profile is used for line segmentation by finding the inter line gap and considered as a separation between two consequent lines. The approach is efficient for printed text and when no overlapping or touching is detected between lines [1, 14, 16, 17, 18]. Smearing methods: These methods are usually used for handwritten documents to segment the text into lines, the consequent black pixels in the horizontal direction are smeared, then the white spaces or pixels between them are marked with black if the distance between them and the black pixels is less than a threshold value. The boundaries of smeared region is considered as a line segment, the algorithm fails in case of touching lines, also it is not useful for fully overlapped lines [1, 19, 14, 4, 18]. Grouping methods: In this method the connected components of black pixels are grouped, the grouping operation is based on some properties like continuity, and similarity. This approach is more used for documents analysis [1, 18]. Bounding box based methods: In this method the histogram for the image is generated then the lines have lesser numbers of pixels are determined, then by finding the centroid for each line through measuring the region properties, the boundaries for each line are determined [14]. Hough transforms methods: Hough peaks are determined and according to this peaks the lines are extracted, this method is mainly used for document analysis [1]. Thinning based methods: This method is applied to the background region to detect the boundaries and separation regions; this method is also used to determine text in the documents [1].

Proposed Method

Our line segmentation approach handles the problem of overlapping between text lines and the over segmentation problem which caused by the existence of diacritics [20, 14]. The proposed approach is top down baseline dependent. Meaning that, the algorithm first segments the page into columns and then for each column a recursive method is applied to segment the column into text lines based on finding the baseline using horizontal projection method with the assumption that the lines are not skewed. The algorithm passes through a set of sequenced dependent stages, as shown in algorithm listed in figure 2, with a gray image of a scanned Arabic printed document with/without diacritics as an input and a segmented document lines as an output.

```
Algorithm 1:  Page and Line Segmentation Algorithm
   Input: A grayscle image (Igray)
   Output: List of image lines
1  begin
2      // Convert the input image to cleaned format.
       Icleaned ← Image_preprocessing(Igray)
3      // Segments the page into columns.
       Page_columns_list ← page_layout_segmentation(Icleaned)
4      // segments each column into its separated segments and lines.
5      while Page_columns_list not empty do
6          Page_column ← Next_page_column(Page_columns_list)
7          Column_segments_list ← Column_segmnetation(Page_column)
8          while Column_segments_list not empty do
9              Column_segment ← Next_column_segmnents(Column_segments_list)
10             Segments_line_list ← Line_segmentation(Column_segment)
11         end
12     end
13     return Segments_line_list
14 end
```

Figure 2. *Page and line egmentation algorithm.*

```
Algorithm 2:  Preprocessing
   Input: A grayscle image (Igray)
   Output: Cleaned
1  begin
2      // Convert the input image to binary format
       Ibinary ← Convert_to_binary(Igray)
3      // reduce the overlapping and interlacement
       ICleaned ← Morphological_Opening(Ibinary)
4      return ICleaned
5  end
```

Figure 3. *Pre-processing steps.*

Algorithm 3: Page Layout Detection and Segmentation

Input: Binarized cleaned Image (CImage)
Output: List of columns the page layout consist of

```
1  begin
2      // vertical projection is applied.
       V_proj ← Vertical_projection(CImage))
3      // indices where vertical projection is equal
4      to zero are detected and these are the
       separation regions indices .
       Separation_indices ← V_proj==0
5
6          // every set of separation indices are
7      grouped and each group is then considered
       as separation region.
       continuous_regions ←
8       grouped_continuous_indices(Separation_indices)
           // filter separation region.
9      separation_regions ←
10      continuous_regions > Threshold
11     for i ← 1 to length(separation_regions)
        do
12         // page column min limit .
          column_min_index(i) ←
13        min(separation_regions(i))
           // page column max limit.
14        column_max_index(i) ←
15        max(separation_regions(i + 1))
           // page column part .
16        column(i) ← Igray(:,
17        column_min_index(i):
          column_max_index(i))
           // Add the column to columns' list .
18        ListOf(columns).Add(column(i));
19     end
       return ListOf(columns)
   end
```

Figure 4. *Page layout detection and segmentation algorithm.*

Pre-processing

Pre-processinghas a big impact on thequalityof the OCR. It produced a clean-up image that facilitates the processing of the next stages. In this stage, firstly, the Otsus binarization method [12] is applied to convert the image into two levels of pixels values, 0 and 1, aiming at reducing the noise and overcome the illumination that is resulted during the scanning process. Secondly, a morphological opening is applied to text lines that caused by the existence of diacritics. Algorithm as shown in figure 3 summarizes this step.

Page Segmentation

The aim of this stage is to find the number of columns that the page consists of and extract them as individual pieces. The

Figure 5. *Separation regions where the vertical projection equals to zero.*

algorithm as shown in figure 4 starts by applying a vertical projection to the image resulted from the pre-processing stage. Then, the Indices that perform zero vertical projection as shown in figure 5 are maintained, and the continuous white space regions are determined by these indices. The separation will be applied according to these areas if the length of the separation area greater than a certain threshold value determined experimentally. The separation is done by determining the min and max column indices for each column region, so the column is located between current separation region max index and the next separation region min index. Algorithm in figure 4 shows the steps for page layout segmentation algorithm.

Column Segmentation

Each column resulted from the previous step is divided into segments and each segment consists of one or more lines. Horizontal projection is an efficient way for line segmentation when there is inter-line gap between consecutive lines. In the existence of diacritics, not all regions between lines perform a clear gap, so an opening technique was applied in the pre-processing stage to reduce the overlapping and interlacement, also by using this technique, the segments with different font sizes will have a clear gap.

Each image column resulted from the previous step is processed as following: Horizontal projection is applied to the column image, and the continuous indices where the projection equals to zero is determined and grouped. Each group performs a separation region between the segments. Algorithm in figure 6 summarizes the column segmentation stage and figure 7 shows an example of the output of this stage.

Line Segmentation

To get the approximated lines count inside the segment extracted from the column image we need to calculate the single

Algorithm 4: Column Segmentation

Input: Column Image (CImage)
Output: List of column Segments

```
1  begin
2      // horizontal projection is applied.
       H_proj ← horizontal_projection(CImage))
3      // indices where horizontal projection is
       equal to zero are detected .
4      Separation_indices ← H_proj(i, j) == 0
5      // every set of continuos separation indices
       are grouped to form the separation region.
6      separation_regions ←
       grouped_continuous_indices(Separation_indices)
       // extract column segmnts and add them to
       the column segment list .
7      for i ← 1 to length(separation_regions)
8       do
9          // segment row start index .
           segment_row_start_index(i) ←
           min(separation_regions(i))
10         // segment row end index .
           segment_row_end_index(i) ←
           max(separation_regions(i + 1))
11
12         // crop the segment using start and end
           indices segment(i) ← Crop(CImage,
           segment_row_start_index(i),
13         segment_row_end_index(i))
14                                  // Add
           the cropped segment to the segments' list
           Column_Segments_list.Add(segment(i));
       end
       return Column_Segments_list)
15 end
```

Figure 6. *Column segmentation algorithm.*

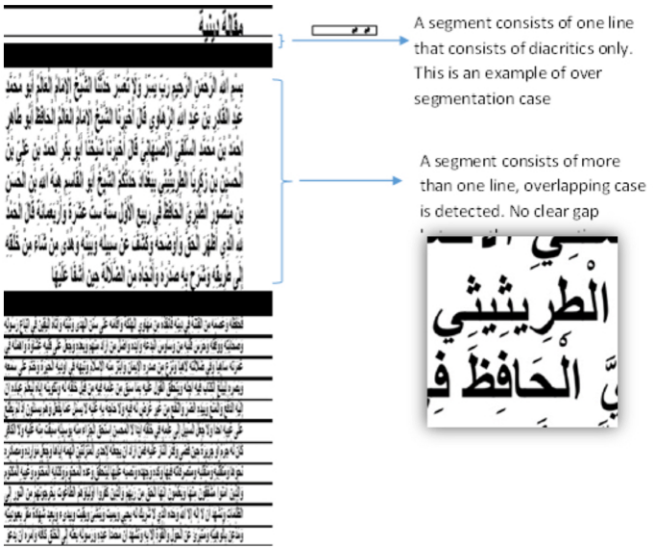

Figure 7. *Column segmentation example.*

Figure 8. *Column segmentation example.*

Figure 9. *Column segmentation example.*

Figure 10. *Overall line segmentation example.*

line width after that the count is easily calculated by dividing the width of the segment by the width of the single line.

To estimate the line width, horizontal projection method is applied to the segment to find the global maximum peak and its location. The location of the global maximum peak performs the baseline for single line in the segment; the baseline is then marked by marking the pixels pass through it with white color.

Figure 8 shows the line drawn at global maximum peak, this drawn line makes the main body of the line as one connected component and extracted by selecting the connected components that passes the baseline after applying connected component algorithm on the segment. The obtained lines from the selection operation are without diacritics, and its width is less than the actual width because it hasnt diacritics. Figure 9 shows the selection operation for the line located at a global maximum peak (baseline). To give better result for the line width, the line width assigned by dividing width of the segment by the approximated count of lines. The estimated number of lines in the segment is then determining by dividing the segment width by the line width. If the resulted count of lines inside the segment greater than one, then the seg-

Algorithm 5: lines segmentation

Input: Column segment (CSegment)
Output: List of Column segment lines

```
1  begin
2      // Horizontal projection. H_proj ← Horizontal_projection(CSegment)
3
4      // Get global maximum peaks. max_peaks ← global_peaks(H_proj)
5
6      // get location of the maximum peaks. max_peaks_location ← get_location(max_peaks)
7
8      // draw a horizontal line at the global maximum peak location so the line will perform one object. CSegment ← draw_line_at_max_peak_location(CSegment, max_peaks_location)
9      // select the line that locates at the global maximum peak, the selected object performs a single line without its diacritics . lines_at_max_peak ← Extract_line_at_max_peak(CSegment)
10     // gets the estimated number of lines in the segment.
       estimated_line_counts ← (Csegment_width
11     lines_at_max_peak_width )
12     if estimated_line_counts == 1 then
13         List_of_lines← over_segmentation_checking(CSegment, List_of_lines)
14     else
15         List_of_lines ← overlap_handling(CSegment)
16     end
17     return List_of_lines
18 end
```

Figure 11. *Line segmentation algorithm.*

ment 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 10 shows an example result of line segmentation and figure 11 shows the algorithm that summarize the steps of line segmentation stage.

Evaluations and Results

The algorithm is tested on scanned images at 300 dpi with multiple font sizes between 8-24, and styles includes regular, bold and italic, also different types of documents are used like magazines, newspapers, and reports the algorithm gives excellent results for both document with diacritics and without diacritics, it reaches up to 99.9 % without diacritics, and up to 99.5% for documents with diacritics, the used data set are text based documents And line spacing which is used to generate them is zero. Table 1 below shows the results generated through the testing process over variation of font type, style and size.

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

Font	Font Type	Total Number of Lines	No. of Lines Correctly Segmented	Accuracy
Regular	Advertising Bold	3,000	2,982	99.4%
	Diwani	3,011	2,988	99.2%
	Andalus	2,520	2,499	99.2%
	Arabic Transparent	2,940	2,922	99.4%
	Naskh	2,982	2,960	99.3%
Bold	Advertising Bold	3,000	2,982	99.4%
	Diwani	3,011	2,988	99.2%
	Andalus	2,520	2,499	99.2%
	Arabic Transparent	2,940	2,922	99.4%
	Naskh	2,982	2,960	99.3%
Italic	Advertising Bold	3,034	3,020	99.5%
	Diwani	3,036	3,024	99.6%
	Andalus	2,438	2,420	99.3%
	Arabic Transparent	3,051	3,038	99.6%
	Naskh	3,118	3,102	99.5%
	Total	**43,055**	**43,271**	**99.5%**

Conclusion

Line segmentation acts a critical step in the segmentation stage in the OCR systems, this paper addresses the main problems in Arabic OCR line segmentation which appear due to the diacritics existence which causes lines overlapping specially for small font sizes, and the over segmentation problem mainly for large fonts, in this case the diacritics performs independent lines, these problems are covered and solved in efficient way, the proposed algorithm shows encouraging results for Arabic script documents with diacritics and excellent results for documents without diacritics. The proposed algorithm covers almost all cases for Arabic script and tested in more than 43000 lines with different font type, size and style. As the propose algorithm shows excellent result, we still need to do more evaluation on other Arabic font and extend this to Farsi and other languages.

References

[1] A. Cheung, M. Bennamoun, and N. W. Bergmann, "An arabic optical character recognition system using recognition-based segmentation," *Pattern recognition*, vol. 34, no. 2, pp. 215–233, 2001.

[2] S. Naz, A. I. Umar, S. H. Shirazi, S. B. Ahmed, M. I. Razzak, and I. Siddiqi, "Segmentation techniques for recognition of arabic-like scripts: A comprehensive survey," *Education and Information Technologies*, pp. 1–17, 2015.

[3] M. Omidyeganeh, K. Nayebi, R. Azmi, and A. Javadtalab, "A new segmentation technique for multi font farsi/arabic texts." in *ICASSP (2)*, 2005, pp. 757–760.

[4] D. Brodić and Z. N. Milivojević, "Text line segmentation with the algorithm based on the oriented anisotropic gaussian kernel," *Journal of Electrical Engineering*, vol. 64, no. 4, pp. 238–243, 2013.

[5] M. G. Mostafa, "An adaptive algorithm for the automatic segmentation of printed arabic text," in *17th National Computer Conference*. Citeseer, 2004, pp. 437–444.

[6] M. G. Kchaou, S. Kanoun, and J.-M. Ogier, "Segmentation and word spotting methods for printed and handwritten arabic texts: A comparative study." in *ICFHR*, 2012, pp. 274–279.

[7] J. Ahmad, "Optical character recognition system for ara-

bic text using cursive multi-directional approach," *Journal of Computer Science*, vol. 3, pp. 549–555, 2007.
[8] "Optical character recognition," https://www.alsintl.com/blog/most-common-languages/, 2016, [Online; accessed 20-September-2016].
[9] T. X.-l. Abbas H Hassin, HUANG Jian-hua, "Offline arabic recognition system," *Journal of Harbin Institute of Technology (New Series)*, vol. 10, no. 1, pp. 80–88, 2003.
[10] P. Xiu, L. Peng, X. Ding, and H. Wang, "Offline handwritten arabic character segmentation with probabilistic model," in *International Workshop on Document Analysis Systems*. Springer, 2006, pp. 402–412.
[11] M. G. Mostafa, "An adaptive algorithm for the automatic segmentation of printed arabic text," in *17th National Computer Conference*. Citeseer, 2004, pp. 437–444.
[12] A. M. Zeki, M. S. Zakaria, and C.-Y. Liong, "Segmentation of arabic characters: A comprehensive survey," *International Journal of Technology Diffusion*, vol. 2, no. 4, pp. 48–82, 2011.
[13] K. Mohammad, M. Ayyesh, A. Qaroush, and I. Tumar, "Printed arabic optical character segmentation," in *SPIE/IS&T Electronic Imaging*. International Society for Optics and Photonics, 2015, pp. 939 911–939 911.
[14] P. Soujanya, V. K. Koppula, K. Gaddam, and P. Sruthi, "Comparative study of text line segmentation algorithms on low quality documents," *CMR College of Engineering and Technology Cognizant Technologies, Hyderabad, India*, 2010.
[15] A. Cheung, M. Bennamoun, and N. W. Bergmann, "An arabic optical character recognition system using recognition-based segmentation," *Pattern recognition*, vol. 34, no. 2, pp. 215–233, 2001.
[16] N. A. Shaikh, G. A. Mallah, and Z. A. Shaikh, "Character segmentation of sindhi, an arabic style scripting language, using height profile vector," *Australian Journal of Basic and Applied Sciences*, vol. 3, no. 4, pp. 4160–4169, 2009.
[17] M. G. Kchaou, S. Kanoun, and J.-M. Ogier, "Segmentation and word spotting methods for printed and handwritten arabic texts: A comparative study." in *ICFHR*, 2012, pp. 274–279.
[18] S. Hussain, S. Ali *et al.*, "Nastalique segmentation-based approach for urdu ocr," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 18, no. 4, pp. 357–374, 2015.
[19] I. Aljarrah, O. Al-Khaleel, K. Mhaidat, M. Alrefai, A. Alzu'bi, M. Rabab'ah *et al.*, "Automated system for arabic optical character recognition with lookup dictionary," *Journal of Emerging Technologies in Web Intelligence*, vol. 4, no. 4, pp. 362–370, 2012.
[20] I. Abuhaiba, "Segmentation of discrete arabic script document images," *Al Azhar University*, vol. 8, no. 1810-6366, pp. 85–108, 2006.

Author Biography

Muna Ayyash: Graduate student at Birzeit University.

Dr. Khader Mohammad: he is currently working as 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.

Mr. Aziz Qaroush: Master of Computer Systems Engineering, working as lecturer in the Engineering and Technology collage at Birzeit University where he teaches graduate and undergraduate level courses in image processing and machine learning.

Mr. Mahdi Washha: He received the Bachelor degree in Computer Systems Engineering, with high distinction, from university of Birzeit in 2012. He was employed then as a teaching assistant in the department of Computer Systems Engineering at the university of Birzeit from 2012 to 2013. Since the end of 2013, he completed a master degree in Artificial Intelligence and Robotics in the department of Information Engineering, Computer Science, and Statistics atthe University of Roma La Sapienza. Currently, he is a candidate Phd student at IRIT lab. He works in detecting spam content and anomaly behaviors on social networks such as Twitter and Facebook, targeting to improve data quality. His research interests include email spam filtering, web spam filtering, information retrieval systems, robot programming, visual learning, machine learning algorithms, and Arabic optical character recognition (OCR) systems.