# Kurdish Text Segmentation using Projection-Based Approaches

**Tofiq Ahmed Tofiq, Jamal Ali Hussien**

*Department of Computer Science, College of Science, University of Sulaimani, Sulaimani, Iraq*

**ABSTRACT**

An optical character recognition (OCR) system may be the solution to data entry problems for saving the printed document as a soft copy of them. Therefore, OCR systems are being developed for all languages, and Kurdish is no exception. Kurdish is one of the languages that present special challenges to OCR. The main challenge of Kurdish is that it is mostly cursive. Therefore, a segmentation process must be able to specify the beginning and end of the characters. This step is important for character recognition. This paper presents an algorithm for Kurdish character segmentation. The proposed algorithm uses the projection-based approach concepts to separate lines, words, and characters. The algorithm works through the vertical projection of a word and then identifies the splitting areas of the word characters. Then, a post-processing stage is used to handle the over-segmentation problems that occur in the initial segmentation stage. The proposed method is tested using a data set consisting of images of texts that vary in font size, type, and style of more than 63,000 characters. The experiments show that the proposed algorithm can segment Kurdish words with an average accuracy of 98.6%.

**Index Terms:** Optical character recognition, Character segmentation, Kurdish text segmentation, Projection-based approach, and Cursive writing optical character recognition

## 1. INTRODUCTION

Optical character recognition (OCR) is an application for image recognition that can read text from images. This can be achieved by taking an image that includes text, written in a specific language to be understood by the computer, and get the final computer representation for the text. OCR techniques may vary according to the nature of the language and the purpose of the OCR application [1]. Emulating the human ability in associating symbolic identities with images of characters at a fast rate is the main goal of OCR.

Kurdish is one of the languages that present many challenges to OCR. The main challenge in Kurdish is that it is mostly cursive. Kurdish is written by connecting characters together to produce words or parts of words, as shown in Fig. 1. Kurdish text is written from right to left. The Kurdish language has 34 basic characters, of which 14 have from one to three dots and four of them have diacritics.

Kurdish characters have many shapes and sizes depending on their position in the word. For example, the character "ک" is written in the form of "کـ" at the start, "ـکـ" in the middle, and "ـک" at the end of a word but the separated form of this character is "ک." Kurdish characters vary with relevance to their position in the word, representing a great challenge for OCR [1].

Based on the nature of Kurdish fonts, characters may overlap vertically to produce certain compounds of letters at certain positions of the word segments, such as "لا, حم and نج," which can be represented by single atomic graphemes called ligatures [3], [4].

Access this article online

| DOI: 10.21928/uhdjst.v5n1y2021.pp56-65 | E-ISSN: 2521-4217 P-ISSN: 2521-4209 |
|---|---|

**Corresponding author's e-mail:** Tofiq Ahmed Tofiq, Department of Computer Science, College of Science, University of Sulaimani, Sulaimani, Iraq. E-mail: Tofiq.ahmad@univsul.edu.iq

Received: 23-01-2021 Accepted: 12-05-2021 Published: 16-05-2021

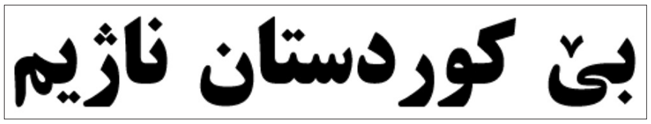
بێ کوردستان ناژیم

Fig. 1. The character connectivity of Kurdish text.

Some Kurdish letters have single dots such as ن, ج and ب, other letters have double dots, such as ش and ق and other have triple dots, such as ش and ڤ. Furthermore, some letters have diacritics such as ۆ, ڵ, and ێ. Besides, the same shape with different dots and diacritics is used to represent different letters, such as, ح, ج, چ, ڵ,ل, ۆ, and و. The doted characters and letters with diacritics present a big challenge while being processed.

This paper presents a Kurdish text segmentation algorithm. The proposed algorithm uses the projection-based approach concepts to separate lines, words, and letters.

The rest of the paper is organized as follows: Section 2 segmentation-based methods, section 3 related works with different segmentation techniques. Section 4 presents the proposed algorithm. Section 5 demonstrates the results and performance analysis. Section 6 concludes this paper.

## 2. SEGMENTATION-BASED METHODS

In this part, the proposed algorithm for the segmentation of cursive text such as Arabic, Persian, and English handwriting text is discussed. The segmentation-based methods can be classified into:

a. Projection profile
b. Character skeleton based
c. Contour tracing based
d. Template matching based
e. Morphological operations based
f. Segmentation based on neural networks (NNs) [2], [5].

Projection profiles methods [9]-[11] are usually used to aim for lines, words, sub-words, and characters segmentation specifically when there is a certain gap between lines, words, sub-words, and characters. Indeed, horizontal projection is used for line segmentation, while vertical projection is usually used for word, sub-word, and character segmentation. In the skeleton method, different thinning techniques are engaged for this goal [7], [12]. In many cases, the shape of the characters is different from the main character after thinning, making the splitting process more difficult. In contour tracking [13]-[16] methods, pixels that represent the external shape of a character or word are extracted. Researchers used many methods to determine the cut points in the contour. In general, the contour-based technique avoids the issues that seem once applying to the thinning methods because they depend on extracting the structure of the word, which provides an obvious description of it. This type of method is sensitive to noise, which needs one to perform some preprocessing steps. Morphological methods [17]-[19] use a set of morphological operations for segmentation. In general, closing and opening operations are applied. These methods are dependent, meaning that other techniques must be used in addition to segmentation. Template matching methods [20], [21] often apply a sliding window over baselines. If any match is found, then the center pixel in the sliding window is considered as a cutting point. The main limitation of this method is when the cutting point locates under the baseline. Finally, in 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 [22].

## 3. RELATED WORKS

Zheng *et al.* [10] proposed a machine-printed Arabic character segmentation algorithm that uses a vertical projection method and some rules or features, such as structural characteristics between background area and character components and the specification of isolated Arabic characters to find segmentation points.

Cheung *et al.* [6] proposed a segmentation algorithm that uses a technique wherein the overlapping Arabic words/sub-words are horizontally separated, extensively utilizing a feedback loop among the character segmentation and final recognition stages. In the segmentation stage, a series of experimental lines have been produced in two processes, the first process uses Amin's character segmentation algorithm [21] and the second procedure use the convex dominant points detection algorithm developed by Bennamoun and Boashash [23].

Shaikh *et al.* [7] propound an algorithm for Sindhi text segmentation. The height profile vector (HPV) was used for the character extraction. More analysis was done over HPV to detect the locations of the possible segmentation points (PSPs), in some cases, the algorithm failed by performing under or over-segmentation.

Yeganeh *et al.* [13] introduced a segmentation algorithm for up and down contours based on qualified labeling, and the algorithm was developed for multifont Farsi/Arabic texts. The contour of the sub-word is measured using a convolution kernel with a Laplacian edge recognition-based segmentation detection method. The algorithm goes through several stages including contour labeling of each sub-word, contour curvature grouping to improve the segmentation results, character segmentation, adaptive local baseline, and post-processing, the results showed that 97% of characters of the printed Farsi texts were segmented correctly.

Mostafa [24] proposed a segmentation method for printed Arabic text, especially for "simplified Arabic" font with different sizes. Most characters start with and end before a T-junction on the baseline, that is, the main rule used in this. This rule was fine for most characters, except for some special characters such as "س" and "ش," which had a special solution. The algorithm was tested and achieved a 96.5% of good segmentation accuracy.

Alipour [8] presents an improved segmentation technique for Persian text where a few structural features were used to regulate the relevant segment to increase the quality of segmentation. The vertical projection was used to bring out the word segment over the baseline dots and diacritics were not checkout then the segment was regulated in an additional step by merging the small fragments, this step was needful in the cases where one character is isolated into more than one segment such as "س" and "ش."

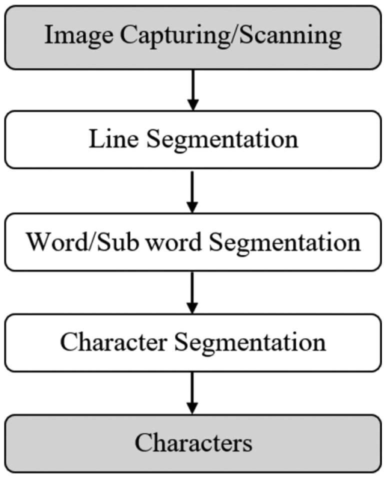

**Fig. 2.** The major steps of the proposed segmentation approach.

## 4. THE PROPOSED ALGORITHM

Our technique is a segmentation-based approach, which contains three main segmentation stages, as shown in Fig. 2. The proposed method takes a binary image that has multiple lines of text and executes several image processing methods to finally segment characters in the image. In this method, the segmentation is performed at three levels: Line segmentation, word/sub-word segmentation, and character segmentation. This work focused only on the line, word/sub-word, and character segmentation steps, considering that the input image has been preprocessed (by applying operations such as binarization, noise removal, and separating text from non-text regions). The image binarization used the below equation.

$$g(x,y)=\begin{cases} \text{üü|}if\ x\ \ y\ <T \\ \text{üü|}if\ x\ \ y\ \geq T \end{cases}$$

Where, (x,y) is the coordinate of the pixels, T represents the threshold value, g(x,y) represents the binary image pixels, and f(x,y) represents gray level image pixels.

### 4.1. Line Segmentation

Line segmentation is achieved by image horizontal projection that calculates the horizontal axis for the binarized image. The horizontal projection matrix $I_j$ is calculated by summing up the pixel values P(i, j) along the X-axis for each y value, as shown in Equation (1):

For each j ∈ 0.m-1

$$I_j=\sum_{i-0}^{n-1}P(i,j) \tag{1}$$

Where, *n* and *m* are, respectively, the width and height of the image and P(*i*,*j*) is the pixel's value at the index (*i*,*j*).

This projection has information about the text lines that are indicated by areas of white intensity, as shown in Fig. 3. White intensities indicate the text area that contains text

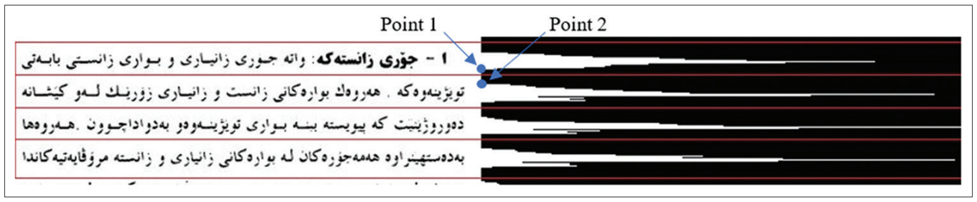

**Fig. 3.** Horizontal projection of input image that contains text line.

and the black intensities show the gap between the text lines.

Fig. 3 shows the line segmentation method that accepts an image of text written in the Kurdish language and separates its lines. The horizontal projection technique does this in two stages. The first one is to locate each group of white intensities in the horizontal projection and the second is to indicate the line position to separate lines from each other. To find the indicator line's position, we used the index of last white intensities (point 1) and the first index of next white intensities(point 2) and calculated the distance between these two points. The line position is in the middle of these two points (point 1 and point 2).

## 4.2. Word/Sub-word Segmentation

After the line segmentation stage, the subsequent stage is word segmentation. The method that is used for word segmentation is based on the connected component method. The algorithm takes a binary text line image of Kurdish text without dots and diacritics as input, and the result is a word/sub-word segmented image as output. The steps of word/sub-word segmentation are described in detail below.

### *4.2.1. Find the connected components*

In this step, the text line image from the previous section is used to find the connected components. A connected component is every component that has adjacent pixels that are connected. Fig. 4(a) shows an example of connected components with boundaries. In the first version of the connected component result, all components are selected but for better word/sub-word extraction dots and diacritics must be ignored. To do so, the baseline of text lines must be found. A baseline is a fictitious line that follows and joins the lower and upper parts of the character bodies [35]. The baseline is the maximum value from the horizontal projection. The index of the max value determines the location of the baseline in the text line image. Fig. 4(a) shows an example of baseline detection that shows by the horizontal line that crosses the whole word. In continuation, using the baseline, the connected components are filtered based on whether these components are intersected with the baseline. Usually, dots and diacritics location are above or below the baseline, so we can ignore connected component that is not on the baseline. Fig. (4a) shows the original image after determining the connected components and the baseline. In Fig. (4b), the dots above the baseline are ignored.

### *4.2.2. Word/sub-words extraction*

For the Kurdish script, a connected component either represents a word or a sub-word. This means that a single word may consist of one or more connected components. For extraction, we applied vertical projection to find the space between the words/sub-words (places that the projection is zero). Projection along the vertical axis is called the vertical projection. Vertical projection is calculated for every column as the sum of all row pixel values inside the column, as shown in Equation 2.

For each i ∈ 0.n-1

$$I_i = \sum_{j-0}^{m-1} P(i,j) \qquad (2)$$

After finding gaps between components, the task is to decide if these gaps are spaces between words or between sub-words. In other words, what is the correct threshold to decide whether the separation space between two sub-words is a gap inside the same word or space between two different words? Although these gaps are not constant and can be vary based on the font sizes, font type, or style. To deal with this issue, first, we find the pen size, which is the pen thickness used for the writing of the adjacent two sequential words/sub-words, and evaluate it with the distance between the current consecutive words/sub-words. Calculating the pen size can handle by taking the most frequent value in the vertical projection applied for each sub-word. However, taking the maximum common value from the vertical projection of some single characters like "ا" gives a wrong conjecture of the pen size. Therefore, the pen size is calculated by considering the most common value calculated from the horizontal projection. Hence, if the maximum frequent value computed from the horizontal projection is

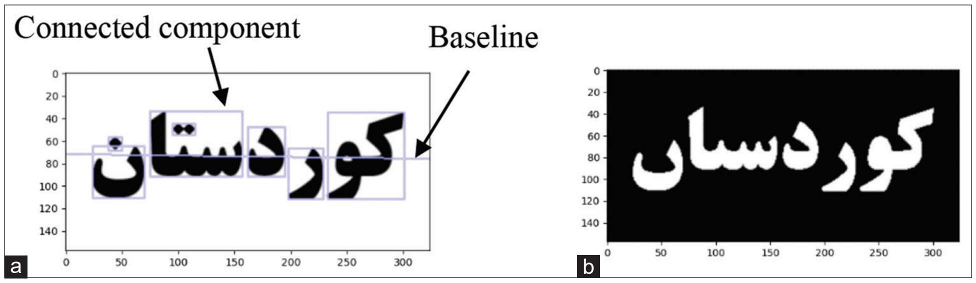

Fig. 4. (a) All connected component. (b) Binary version of ignored dots and diacritics.

greater than the maximum frequent value calculated from the vertical projection, then the pen size is equal to the maximum frequent value calculated from the vertical projection.

Pen size calculation is formally defined as:

$$PS = \begin{cases} \text{üüüüüüü(üüüü}), \mid \quad [\quad (\quad )] > \quad [\quad (\quad )] \\ MFV(HP), \mid \textit{otherwise} \end{cases}$$

Where, PS is the pen size, SW indicates sub-word, HP shows horizontal projection, VP shows vertical projection, and MFV represents the most frequent value. Fig. 5 shows an example of a pen size calculation for two cases.

After finding the pen size for each sub-word, it is compared with the spaces between the components. If the gap between two adjacent components (SS) is greater than the mean of the PS value of these two adjacent components, then this gap is a space between two separate words (WS), otherwise, this gap is a space between two sub-words (SWs) that belong to the same word. Figure 6 shows an example that determining the type of gap between the two words/sub-words in the same line. It is defined formally as:

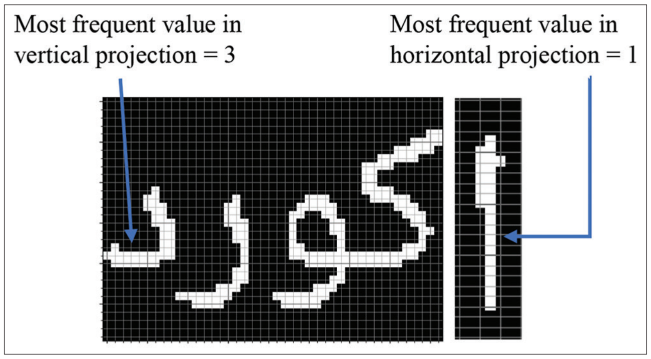
Most frequent value in vertical projection = 3
Most frequent value in horizontal projection = 1

**Fig. 5.** Finding the most frequent value in both VP and HP, this value is the sub-word pen size.

$$SR = \begin{cases} WS, \left| SS[SW(i), SW(i+1)] > \dfrac{PS(i) + PS(i+1)}{2} \right. \\ SWS, \left| \textit{otherwise} \right. \end{cases}$$

Where, SR, WS, SWS, SS, and SW are the separation region, the word separation, the sub-word separation, the separation space, and the sub-word, respectively.

## 4.3. Character Segmentation

The proposed algorithm for character segmentation is based on the vertical projection. The algorithm consists of two stages. The algorithm takes a binary image of word/sub-word and returns a binary image of segmented characters. Each step is explained in detail below:

### *4.3.1. Word/sub-word vertical projection*

Vertical projection can provide a better definition of a letter's shape. This method can give us an accurate result. At this stage, we will once again find a vertical projection for the word. This technique reveals all the convexity and dents in the word. Fig. 7 shows an example of a vertical projection stage.

### *4.3.2. Segmentation areas identification*

In this step, the vertical projection as shown in Figure 8 is examined to identify the segmentation (splitting) areas between letters. The segmentation area between the two letters is an area that ended one letter and started another letter or ended the word and we know that the baseline shared between all letter in a word, this means, letters join by the baseline, and if we find the pixel of baseline, we can find the start and the end of a letter or the area between letters. In the different font sizes and font styles, the baseline height or the pen size can be found by the most frequent data (MFD) in the vertical projection that we discussed previously. The process starts with finding the MFD in the vertical projection array. After this, we compare the other data in the VP array with the MFD. If the

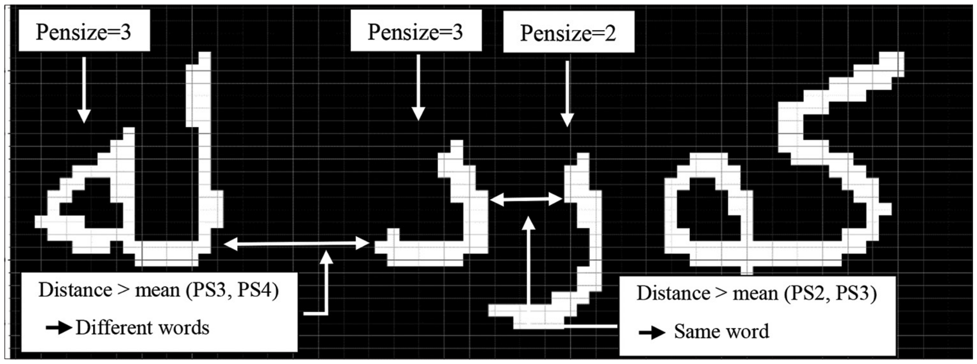

**Fig. 6.** Calculate the distance between two sub-words in the same word and different words.

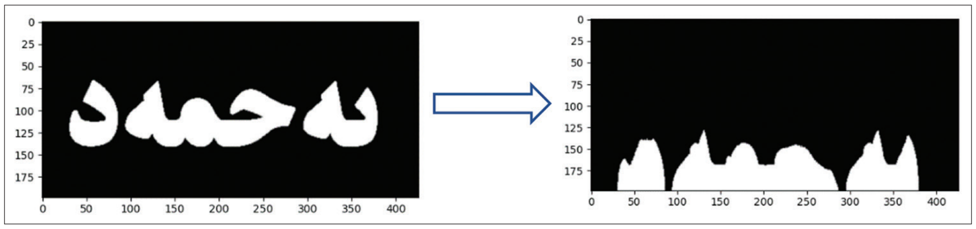

Fig. 7. Vertical projection example.

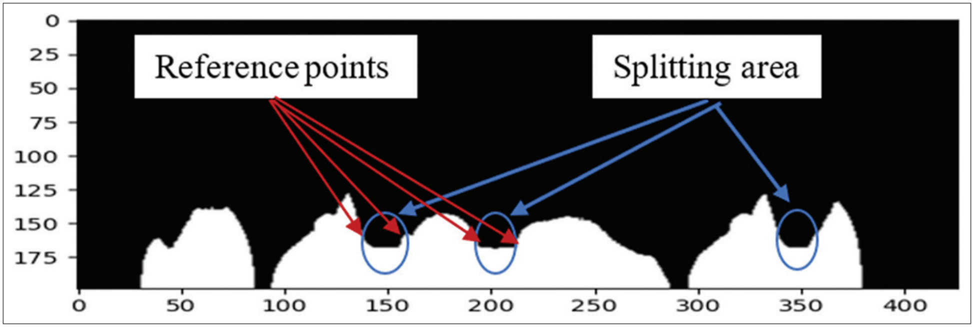

Fig. 8. Splitting areas with reference points.

difference of these data is less than 1, we will change the data of this index to the MFD, but if this difference is more than 1, we change the value of this cell to zero because this index is part of the character, not the splitting area. Now, we have an array consisting of MFD and zero values. Therefore, the MFD data in the array represent the splitting area between the two letters. By finding the start and end index of this group of data in the array, the beginning and end of the splitting area can be specified. After that, the first and last points in the region are considered as splitting area reference points. The start and endpoints can be used to separate the letters. By adding the middle line between these two points (separator line), as well as adding the start and end lines of connected components that are found in previous steps, we now have a separator line between the characters. Fig. 8 shows an example of finding the splitting area and reference points.

After identifying the splitting areas, each character is located between two consecutive splitting area. Fig. 9 shows the separator line over splitting areas for some characters.

However, there are some special cases where the splitting area locates within a character such as the letters "س" and "ش" in all forms. Besides, some splitting areas locate within a character when the position of the character is at the end of the word or exists independently in the text such as "ب" and "ی." Hence, a post-processing step for character segmentation is necessary to ignore these spatial splitting areas. However, in this paper, we do not work on the "س" and "ش," we only worked on the "ب" and "ی." Figure 10 shows some examples of these cases.

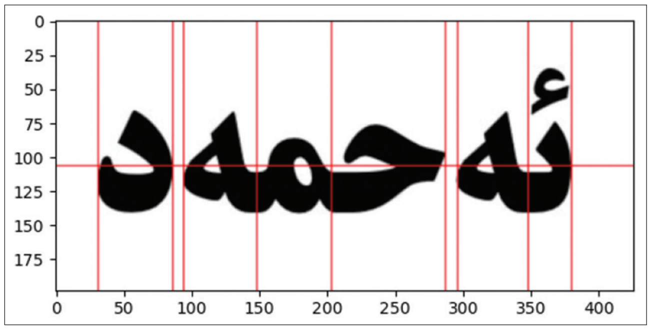

Fig. 9. Splitting regions for some regular characters.

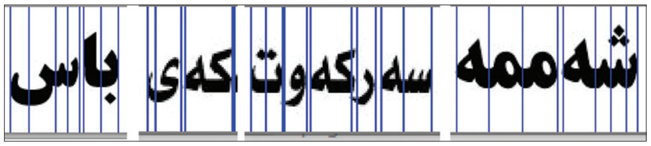

Fig. 10. Example of special cases that must ignored.

### *4.3.3. Post-processing*

In the post-processing phase, we will take a step. In this main step, we can eliminate some of these special cases using the baseline that we found earlier, as well as the separator line that we found in the previous step. To do this, for any separation lines find the intersection point with the baseline. After that, check the value of this point (intersection point) in the image

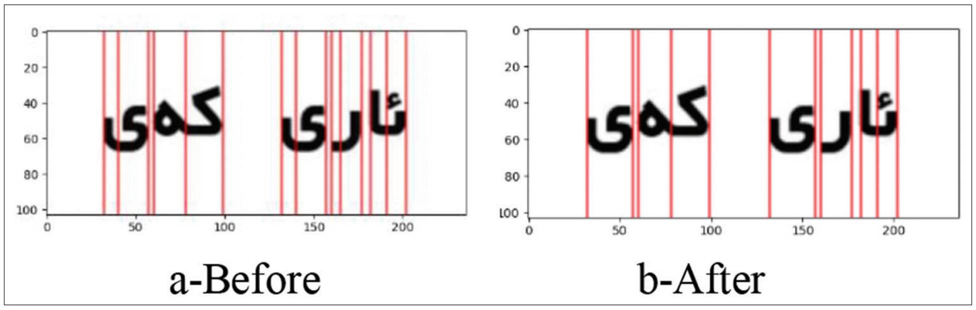

**Fig. 11.** Example of post-processing before and after applying.

binary data and if this value is zero its means this point not on the letter and we know that the separation line between the adjacent latter must be on the letters. Finally, these separation lines can be removed from the list. In this way, we can solve some of these situations using this technique. Fig. 11 shows some example after applying the post-processing step.

## 5. PERFORMANCE ANALYSIS

In this section, the results of testing our approach on a collection of images that contain Kurdish text are shown. We use the Python programming language to implement and then test our proposed character segmentation algorithm since it is a commonly known high-level programming language that provides well-implemented packages for image processing. The performance of line segmentation is measured by computing the ratio of the number of lines that are correctly segmented to the total number of inputted lines. The same measurement is used for each of word and character segmentations.

$$Accauracy = \frac{Num.of\ corrected\ segment}{Total\ of\ segments}$$

The proposed algorithms (line, word, and character segmentations) were experimented and evaluated using a manually created dataset. We develop a software to generate a dataset with the ground truth from the random Kurdish text that we collected. 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 variation at font type, size, and style levels. These texts are converted to image word by word and add some noise to every image and saved all images. The proposed line segmentation methods were tested on 6099 lines and reported excellent results in terms of line segmentation ratio, which computed with an average of 99.9%. Table 1 shows the results generated through the testing process using different font types, styles, and sizes.

**TABLE 1: Line segmentation results for different font styles and types on text**

| Font | Font type | Total number of input lines | No. of correctly segmented lines | Accuracy (%) |
|---|---|---|---|---|
| Plain | UniQAIDAR_ Blawkrawe 004 | 497 | 497 | 100 |
| | Unikurd Web | 507 | 506 | 99.9 |
| | Noto Naskh Arabic UI | 525 | 524 | 99.9 |
| | UniQAIDAR_ JWNEYD | 504 | 504 | 100 |
| Bold | UniQAIDAR_ Blawkrawe 004 | 497 | 497 | 100 |
| | Unikurd Web | 507 | 506 | 99.9 |
| | Noto Naskh Arabic UI | 525 | 524 | 99.9 |
| | UniQAIDAR_ JWNEYD | 504 | 504 | 100 |
| Italic | UniQAIDAR_ Blawkrawe 004 | 497 | 497 | 100 |
| | Unikurd Web | 507 | 506 | 99.9 |
| | Noto Naskh Arabic UI | 525 | 524 | 99.9 |
| | UniQAIDAR_ JWNEYD | 504 | 504 | 100 |
| | Total | 6099 | 6093 | 99.9 |

**TABLE 2: Word segmentation results for different font types and size between 24 and 48**

| Font | Font type | Total number of input word | No. of correctly segmented word | Accuracy (%) |
|---|---|---|---|---|
| Plain | UniQAIDAR_ Blawkrawe 004 | 2218 | 2180 | 98.28 |
| | Unikurd Web | 2218 | 2112 | 95.22 |
| | Noto Naskh Arabic UI | 2218 | 2150 | 96.93 |
| | UniQAIDAR_ JWNEYD | 2218 | 2119 | 95.53 |
| | Total | 8872 | 8561 | 96.5 |

TABLE 3: Character segmentation results for different font types and font size between 24 and 48

| Font | Font type | Total number of input character | No. of correctly segmented character | Accuracy (%) |
|---|---|---|---|---|
| Plain | UniQAIDAR_ Blawkrawe 004 | 15,887 | 15,842 | 99.7 |
| | Unikurd Web | 15,887 | 16,101 | 98.7 |
| | Noto Naskh Arabic UI | 15,887 | 15,636 | 98.4 |
| | UniQAIDAR_ JWNEYD | 15,887 | 15,495 | 97.5 |
| | Total | 63,548 | 63,074 | 98.6 |

The results of the word segmentation stage in terms of word segmentation ratio are reported in Table 2. The proposed word segmentation methods are experimented on about 8872 words with four font types (Noto Naskh Arabic UI, Unikurd Web, UniQAIDAR_JWNEYD and UniQAIDAR_Blawkrawe 004) and five font sizes (24, 26, 28, 36, and 48 points), with an average accuracy of 96.5%. The results show that the algorithm has almost the same performance when changing the font size. Furthermore, we experimented with the character segmentation stage on different font types and sizes on about 63,548 characters. Table 3 shows the performance of the proposed algorithm with an average accuracy of 98.6%. The results show that the algorithm has almost the same performance regardless of the font type, style, and size.

TABLE 4: Comparing with other related work

| Articles | Year | Segmentation method | Dataset | Font type | Font size | Font style | Average accuracy (%) |
|---|---|---|---|---|---|---|---|
| Zheng *et al*. [10] | 2004 | Vertical histogram and some structural characteristics rules | 500 samples of Arabic text | Simplified Arabic and Arabic transparent | 12, 14, 16, 18, 20, and 22 | Plain | 94.8 |
| Javed *et al*. [27] | 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 | 92 |
| Saabni [26] | 2014 | Partial segmentation and Hausdorff distance | APTI | Different fonts to cover different complexity of shapes of Arabic printed characters | 10 different sizes | Plain | 96.8 |
| Anwar *et al*. [28] | 2015 | Projection-based | 127 sentences composed of 1061 letters | Traditional Arabic | 70 | Plain | 97.5 |
| Amara *et al*. [29] | 2016 | Histogram and contextual properties | APTI | Different font types | Different sizes | Plain, italic, and bold | 85.6 |
| Radwan *et al*. [32] | 2016 | Multichannel neural networks | APTI | Arial, Tahoma, Thuluth, and Damas | 18 | Plain | 95.5 |
| Qomariyah *et al*. [33] | 2017 | Interests points, contour-based | 10 lines of 30 sub-words | Not reported | Not reported | Plain | 86.5 |
| Fakhry [30] | 2017 | Connected component | 5 lines 15 words | Not reported | Not reported | Plain | 80.2 |
| Amara *et al*. [31] | 2017 | Projection profile, SVM | APTI | Advertising bold | 6,8,10, 12 | Plain, italic, and bold | 98.24 |
| Zoizou *et al*. [34] | 2018 | Contour-based and template matching | 83 lines of 984 words | 34 different fonts | Different font sizes | Plain | 94.7 |
| Mohammad *et al*. [25] | 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 | 98.5 |
| Our approach | 2020 | Projection based | 6099 line of (63,548) letters | UniQAIDAR_004, Unikurd_ Web, Noto_Naskh Arabic UI, UniQAIDAR_JWNEYD | 24, 26, 28, 38, and 48 | Plain | 98.6 |

Table 4 shows our results compared with some previous related works. As shown in the table, the proposed algorithm performs better in comparison with other related works in: (i) Using more font types, sizes, and styles than the other approaches (ii) and recording higher average accuracies.

## 6. CONCLUSION

In this paper, line, word, and character segmentation algorithms are proposed for Kurdish printed text based on projection-based segmentation methods. The proposed algorithm can segment the characters of words. The algorithm can also handle certain complex cases that occur due to over-segmentation problems. We tested the algorithm on the manually created dataset by creating different versions of the same text using different font types, styles, and sizes. Experimental results show the reliability of our algorithm in performing a correct segmentation of more than 63,074 out of 63,548 words without the س and ش letter.

The segmentation of the Kurdish text is prone to errors, which leads to classification errors. The proposed segmentation algorithms are capable of minimize errors and maximize the classification rate. An advanced method is proposed for word/sub-word segmentation. Horizontal and vertical segmentations are used to distinguish between words and sub-words based on the size of the gaps that separate the connected components in comparison to the pen size.

For the character segmentation step, an advanced projection-based algorithm is proposed. The proposed algorithm is built easily and reliably that can fit a variety of fonts and styles, the character segmentation algorithm shows good results up to 98.6%.

For future work, we plan to find the correct segmentation for characters, such as "س" and "ش," by ignoring the over-segmentation part that occurs in these two special characters cases. Furthermore, we want to extend the work to extract all characters more accurately to facilitate the recognition stage.

## REFERENCES

[1] H. Althobaiti and C. Lu. "*A survey on Arabic Optical Character Recognition and an Isolated Handwritten Arabic Character Recognition Algorithm Using Encoded Freeman Chain Code*". 2017 51st Annual Conference on Information Sciences and Systems (CISS), Baltimore, MD, pp. 1-6, 2017.

[2] A. Lawgali. "A survey on Arabic character recognition". *International Journal of Signal Processing, Image Processing and Pattern Recognition*, vol. 8, no. 2, pp. 401-426, 2015.

[3] S. Elaiwat and M. A. Abu-Zanona. "A three stages segmentation model for a higher accurate off-line arabic handwriting recognition. *World of Computer Science and Information Technology Journal*, vol. 2, no. 3, pp. 98-104, 2012.

[4] M. A. Abdullah, L. M. Al-Harigy and H. H. Al-Fraidi. "Off-line arabic handwriting character recognition using word segmentation". *Journal of Computing*, vol. 4, pp. 40-44, 2012.

[5] Y. M. Alginahi. "A survey on Arabic character segmentation". *International Journal on Document Analysis and Recognition*, vol. 16, no. 2, pp. 105-126, 2013.

[6] 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.

[7] 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.

[8] M. M. Alipour. "A new approach to segmentation of Persian cursive script based on adjustment the fragments". *International Journal of Computers and Applications*, vol. 64, no. 11, pp. 21-26, 2013.

[9] S. N. Nawaz, M. Sarfraz, A. Zidouri and. W. G. Al-Khatib. "*An Approach to Offline Arabic Character Recognition Using Neural Networks*". In: 10th IEEE The IEEE International Conference on Electronics, Circuits, and Systems, IEEE, vol. 3, pp. 1328-1331, 2003.

[10] L. Zheng, A. H. Hassin and X. Tang. "A new algorithm for machine printed Arabic character segmentation". *Pattern Recognition Letters*, vol. 25, no. 15, pp. 1723-1729, 2004.

[11] A. Zidouri and K. Nayebi. "*Adaptive Dissection Based Subword Segmentation of Printed Arabic Text*". In: 9th International Conference on Information Visualisation (IV), IEEE, pp. 239-243, 2005.

[12] J. Ahmad. "Optical character recognition system for Arabic text using cursive multi-directional approach". *Journal of Computational Science*, vol. 3, pp. 549-555, 2007.

[13] M. Omidyeganeh, K. Nayebi. "*A New Segmentation Technique for Multi font Farsi/Arabic Texts*". In: IEEE International Conference on Acoustics Speech, and Signal Process., IEEE, vol. 2, 2005.

[14] T. Sari, L. Souici, and M. Sellami. "*Off-line Handwritten Arabic Character Segmentation Algorithm: ACSA*". In: Proceeding 8th International Workshop Front Handwriting Recognit., IEEE, pp. 452-457, 2002.

[15] R. Mehran, H. Pirsiavash and F. Razzazi. "*A Front-end OCR for Omni-font Persian/Arabic Cursive Printed Documents*". In: Digital Image Computing: Techniques and Applications (DICTA), IEEE, pp. 56-56, 2005.

[16] A. Al-Nassiri, S. Abdulla and R. Salam. "The segmentation of off-line arabic characters, categorization and review". *International Journal on Media Technology*, vol. 1, no. 1, pp. 25-34, 2017.

[17] M. M. Altuwaijri and M. A. Bayoumi. "A thinning algorithm for Arabic characters using art2 neural network". *IEEE Transactions on Circuits and Systems*, vol. 45, no. 2, pp. 260-264, 1998.

[18] A. A. A. Ali and M. Suresha. Survey on segmentation and recognition of handwritten arabic script. *SN Computer Science*, vol. 1, p. 192, 2020.

[19] I. Aljarrah, O. Al-Khaleel, K. Mhaidat, M. Alrefai, A. Alzu'bi and M. Rabab'ah. 2012. *Automated System for Arabic Optical Character*

*Recognition*. In: Proceedings of the 3rd International Conference on Information and Communication Systems(ICICS'12).

[20] Y. Alginahi. "A survey on Arabic character segmentation". *International Journal on Document Analysis and Recognition*, vol. 16, pp. 105-126, 2013.

[21] Y. Zhang, Z. Q. Zha and L. F. Bai. "A license plate character segmentation method based on character contour and template matching". *Applied Mechanics and Materials*, vol. 333, pp. 974-979, 2013.

[22] I. Ahmed, M. Sabri and P. Mohammad. *Printed Arabic Text Recognition*. Guide to OCR for Arabic Scripts, 2012.

[23] M. Bennamoun and B. Boashash. "A structural-description-based vision system for automatic object recognition". *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 27, no. 6, pp. 893-906, 1997.

[24] M. Mostafa."*An Adaptive Algorithm for the Automatic Segmentation of Printed Arabic Text*". In: 17th National Computer Conference, International Society for Optics and Photonics, Saudi Arabia, pp. 437-444, 2004.

[25] K. Mohammad, A. Qaroush, M. Ayesh, M. Washha, A. Alsadeh and S. Agaian. Contour-based character segmentation for printed Arabic text with diacritics. *Journal of Electronic Imaging*, vol. 28, no. 4, p. 1, 2019.

[26] R. Saabni. "*Efficient Recognition of Machine Printed Arabic Text Using Partial Segmentation and Hausdorff Distance*". In: 6th International Conference Soft Computing and Pattern Recognition (SoCPaR), pp. 284-289, 2014.

[27] S. T. Javed, S. Hussain, A. Maqbool, S. Asloob, S. Jamil and H. Moin. "Segmentation free nastalique urdu OCR". *World Academy of Science, Engineering and Technology*, vol. 4, no. 10, pp. 456-461, 2010.

[28] K. Anwar, Adiwijaya and H. Nugroho. "*A Segmentation Scheme of Arabic Words with Harakat*". In: IEEE International Conference on Communications, Networks and Satellite (COMNESTAT), pp. 111-114, 2015.

[29] M. Amaram, K. Zidi, G. Ghedira and S. Zidi. "*New Rules to Enhance the Performances of Histogram Projection for Segmenting Small-sized Arabic Words*," In: International Conference on Hybrid Intelligent Systems, 2016.

[30] F. I. Firdaus, A. Khumaini and F. Utaminingrum. "*Arabic Letter Segmentation Using Modified Connected Component Labeling*". In: International Conference on Sustainable Information Engineering and Technology (SIET), pp. 392-397, 2017.

[31] M. Amara, K. Zidi and K. Ghedira. "An efficient and flexible knowledge- based Arabic text segmentation approach". *The International Journal of Computer Science and Information Security*, vol. 15, no. 7, pp. 25-35, 2017.

[32] M. A. Radwan, M. I. Khalil and H. M. Abbas. "Predictive segmentation using multichannel neural networks in Arabic OCR system". *Lecture Notes in Computer Science*, vol. 9896, pp. 233-245, 2016.

[33] F. Qomariyah, F. Utaminingrum and W. F. Mahmudy. "The segmentation of printed Arabic characters based on interest point". *The Journal of Telecommunication, Electronic and Computer Engineering*, vol. 9, no. 2-8, pp. 19-24, 2017.

[34] A. Zoizou, A. Zarghili and I. Chaker. "A new hybrid method for Arabic multi-font text segmentation, and a reference corpus construction". *Journal of King Saud University Computer and Information Sciences*, vol. 32, no. 5, pp. 576-582, 2018.

[35] A. Fawzi, M. Pastor and C. D. Martínez-Hinarejos. "*Baseline Detection on Arabic Handwritten Documents*". P Proceedings of the 2017 ACM Symposium on Document Engineering, pp. 193-196, 2017.