Faculty of Engineering and Technology

Master of Computing

مرحلة التجزئة في أنظمة التعرف الضوئي على حروف اللغة العربية

Arabic Optical Character Segmentation Stage

Submitted By

Muna Abdulfattah Ayyash

Supervised By

Dr. Khader Mohammad

2016 – 1437

Arabic Optical Character Segmentation Stage

Submitted By

Muna Abdulfattah Ayyash

Supervised By

Dr. Khader Mohammad

A thesis submitted in partial fulfillment of requirement for the degree of master of Computing - Faculty of Engineering and Technology - Graduate Studies

Birzeit University
2016 – 1437

Faculty of Engineering and Technology

Master of Computing

Thesis Approval

Arabic Optical Character Segmentation Stage

Submitted by: Muna Abdulfattah Ayyash

Student Number: 1105387

Approved by the thesis committee:

Dr. Khader Mohammad _____

Dr. Majdi Mafarja _____

Dr. Ahmad Alsadeh _____

Abstract

A considerable progress in recognition techniques for many non-Arabic characters has been achieved. In contrary, few efforts have been put on the research of Arabic characters. In any Optical Character Recognition (OCR) system the segmentation step is usually the essential stage in which an extensive portion of processing is devoted and a considerable share of recognition errors is attributed. In this research, a novel segmentation approach for machine Arabic printed text for different segmentation stages; line segmentation, word and sub-word segmentation, and character segmentation including diacritics with different font types, styles, and sizes are proposed. The proposed approach based on profile projection techniques, finding global maximum peak, connected components, region properties, and extracting the word's contour. These methods reduce computation, errors, and give a clear description for the sub-word. Both of evaluation and testing of the proposed methods have been developed using MATLAB and shows nearly 98% accuracy for different segmentation stages.

ملخص

تقدم كبير تم تحقيقه في مجال تقنيات التعرف الضوئي على الحروف في اللغات غير العربية، في المقابل جهود قليلة ومحدودة النتائج قد تم التوصل إليها في اللغة العربية، تعد مرحلة التجزئة في هذه الأنظمة خاصة في مجال اللغة العربية أحد أهم المراحل، إذ يعد الحصول على تجزئة صحيحة مرحلة اساسية ومهمة للحصول على تعرف صحيح على الحروف، وقد جاء هذا البحث للتركيز على مرحلة التجزئة ومعالجة مشاكل مختلفة ومتنوعة يمكن أن تظهر خلال هذه المرحلة، بدءاً من مرحلة تقسيم النص الى أسطر، ثم الى كلمات واجزاء الكلمات والتشكيل، انتهاءً بالحصول على الحروف.

عدد من الطرق والأساليب تم استخدامها في هذا البحث، كتحديد خط الأساس والقيم القصوى والمكونات المتصلة وبعض الخصائص المتعلقة بها، بالإضافة الى استخراج الشكل الخارجي لأجزاء الكلمات الذي يعطي وصفا واضحا لشكل الكلمة، فيسهّلُ عملية التقسيم، واعتمد استخدام برنامج ماتلاب (MATLAB) لإنجاز هذا العمل، وتم الحصول على نتائج تصل تقريبا الى 98% لمراحل التجزئة المختلفة.

Table of Contents

ABSTRACT **1**

ملخص **2**

CHAPTER 1 INTRODUCTION **12**

1.1 INTRODUCTION AND MOTIVATION 12
1.2 RESEARCH OBJECTIVES AND PROBLEM STATEMENT 19
1.3 CONTRIBUTION 20
1.4 RESEARCH METHODOLOGY 22
1.5 THESIS OUTLINE 22

CHAPTER 2 BACKGROUND AND LITERATURE REVIEW. **24**

2.1 BACKGROUND 24
2.2 LINE SEGMENTATION RELATED WORK 25
2.3 WORD / SUB-WORD SEGMENTATION RELATED WORK 28
2.4 CHARACTER SEGMENTATION RELATED WORK 30
2.5 CONCEPTS BACKGROUND 37
2.6 STATE OF ART 38
2.7 THE PROPOSED APPROACHES FEATURES OVER THE PREVIOUS APPROACHES 40

CHAPTER 3 PROPOSED WORK **43**

3.1 TEXT TO LINE SEGMENTATION: ..... **ERROR! BOOKMARK NOT DEFINED.**

3.1.1 Input Image *45*
3.1.2 Page Layout Detection or Segmentation *45*
3.1.3 Generating Segments Consist of Lines with Equal Width ... *51*
3.1.4 Initial Lines Count Estimation Inside the Segment *54*
3.1.5 Split Each Segment to Individual lines in Case of Overlapping *57*
3.1.6 Over Segmentation Problem Detection and Solving *60*

3.2 LINE TO WORDS, SUB-WORDS AND DIACRITICS SEGMENTATION: 61

3.2.1 Text Line Image Processing *62*
3.2.2 Words and Sub-words Extraction *64*
3.2.3 Sub-words Overlapping Case and Diacritics Extraction: .. *67*

3.3 CHARACTER SEGMENTATION 75
3.3.1 Contour Extraction 76
3.3.2 Up-Contour Extraction 77
3.3.3 Splitting Regions Extraction 82
3.3.4 Character Segmentation post processing: 84
CHAPTER 4 TESTING AND RESULTS 93
4.1 TOOLS 93
4.1.1 Software 93
4.1.2 Hardware 93
4.2 DATASETS DESCRIPTION 93
4.2.1 APTID / MF Dataset 93
4.2.2 Thesis (Local) Dataset 94
4.3 PERFORMANCE METRICS 95
4.4 TESTING RESULTS 95
4.4.1 Line Segmentation Results 95
4.4.2 Word/Sub-word Segmentation Results 98
4.4.3 Character Segmentation Results 101
4.5 SOURCE OF FAILURE 106
CHAPTER 5 CONCLUSION AND FUTURE WORK 108
5.1 CONCLUSION 108
5.2 FUTURE WORK 109
REFERENCES 110

List of Figures

Figure 1-1 OCR Development Process 15

Figure 1-2 Baseline connects characters in the same sub-word 17

Figure 1-3 Ligature or compound character (multiple characters overlapped vertically) 17

Figure 1-4 Diacritics in Arabic script 17

Figure 1-5 Different shapes for ‘ayn (ع) character according to its location in the word (at beginning, middle, end, and isolated form) 19

Figure 2-1 OCR segmentation stages 24

Figure 2-2 Splitting areas extracted from the up-contour 34

Figure 2-3 Binary image representation 37

Figure 2-4 Pen size = 1 38

Figure 3-1 OCR segmentation steps 43

Figure 3-2 Arabic text with diacritics 43

Figure 3-3 Arabic text without diacritics 44

Figure 3-4 Line segmentation algorithm stages 45

Figure 3-5 Separation regions where the vertical projection is equal to zero 48

Figure 3-6 Page layout detection for page consists of two columns 49

Figure 3-7 Page layout detection flow chart 50

Figure 3-8 Generating segments with equal lines’ width inside 52

Figure 3-9 Generating segments consist of lines that have equal width 53

Figure 3-10 The drawn line at the global maximum peak (baseline) 56

Figure 3-11 The selection operation for the line passes through the global max peak 56
Figure 3-12 Valid peaks detected in the segment 59
Figure 3-13 Overlapped lines segmentation result 59
Figure 3-14 Word/sub-word segmentation algorithm 62
Figure 3-15-1 Original line image (main bodies + diacritics + dotting) 63
Figure 3-15-2 The drawn line at the location of the baseline 63
Figure 3-15-3 Main body line version with horizontal line 63
Figure 3-15-4 Main body line version 63
Figure 3-15-5 Diacritics and dotting line version 63
Figure 3-15 Example of the three lines generation 63
Figure 3-16 Pen size calculation 66
Figure 3-17 Distance between two sub-words in the same word and different words 67
Figure 3-18-1 Overlapped sub-words 68
Figure 3-18-2 Two overlapped main bodies of sub-words 68
Figure 3-18-3 Diacritics image of overlapped sub-words 69
Figure 3-18-4 Sub-words and diacritics segmentation result in case of overlapping 70
Figure 3-19 Ratio between major axis length and minor axis length is approximately equal 1 73
Figure 3-20 Two touching dots 73
Figure 3-21-1 Input image (line) 74

Figure 3-21-2 Main body line version 74

Figure 3-21-3 Diacritics and dotting line version 74

Figure 3-21-4 Splitting result after applying vertical projection to the main body line version 74

Figure 3-21-5 Diacritics and doting version after splitting at the same locations where the splitting regions locates in the main body line version 74

Figure 3-21-6 Main bodies' separation after applying the connected component concept 74

Figure 3-21-7 Diacritics and dots separation after applying the connected component concept and computing the overlapping percentage 74

Figure 3-21-8 Sub words' separation result after merging the previous two output images 74

Figure 3-21-9 Words separation result after computing the pen size between every two consequent sub-words 74

Figure 3-21 An example of word/sub-word and diacritics segmentation 74

Figure 3-22 Character segmentation algorithm *stages* 75

Figure 3-23 Contour extraction steps 77

Figure 3-24 Last and first parts of the up-contour 78

Figure 3-25 Start and end points election for tracing and extracting the up-contour 79

Figure 3-26 Tracing in counterclockwise direction to extract the up-contour 82

Figure 3-27 Up-contour splitting areas 83

Figure 3-28 Splitting area reference points 84

Figure 3-29 Valid character's part to be merged 86

Figure 3-30 SHEEN case is detected, so the character's part is merged with previous two parts to perform one character 88

Figure 3-31 Space counter = 3, SEEN case is detected and the three parts are merged to perform one character 88

Figure 3-32 Space counter = 1, so SAD or DAD case is detected and the part is merged with previous one to perform one character 88

Figure 3-33 Splitting region is above the baseline so it should be ignored (TAA and THAA cases) 88

Figure 3-34 Last splitting region is below the baseline so it is ignored 89

Figure 3-35 Last splitting region is below the baseline so it is ignored, also the last part is merged after it is detected as part of other character like SEEN, SHEEN, SAD, or DAD. 89

Figure 3-36 SHEEN case detection by one character's part that satisfies the specifications and three overlapped separated dots. 90

Figure 3-37 SHEEN case detection by one character's part that satisfies the specifications and three overlapped dots (one single dot and two touching dots). 91

Figure 4-1 An example of line segmentation output 97

Figure 4-2-1 Simplified Arabic regular style as input 100

Figure 4-2-2 Simplified Arabic regular style word segmentation result. 100

Figure 4-2-3 Simplified Arabic bold style as input 100

Figure 4-2-4 Simplified Arabic bold style word segmentation result. 100

Figure 4-2-5 Simplified Arabic italic style as input 100

Figure 4-2-6 Simplified Arabic italic style word segmentation result. 100

Figure 4-2 Simplified Arabic font type word segmentation result with different styles (Regular, Bold, and Italic) 100

Figure 4-3-1 Input image. 105

Figure 4-3-2 Line segmentation output. 105

Figure 4-3-3 Word segmentation output. 105

Figure 4-3-4 Character segmentation output for line 1 105

Figure 4-3 Example of AOCR output in different stages 105

Figure 4-4 Line segmentation failure due to the variance in font sizes with no spacing 106

Figure 4-5 Line segmentation failure due to the touching with other line for the line locates at global maximum peak 106

Figure 4-6 Word segmentation error due to the pen-size calculation error 106

Figure 4-7 Character segmentation error due to an error in dots detection 107

Figure 4-8 Character segmentation error due to small parts from other line 107

Figure 4-9 Character segmentation error due to the absence of the splitting region 107

List of Tables

Table 1-1 Arabic characters all possible shapes according to their location in the word 18

Table 2-1 Line and word/sub-word segmentation previous work approaches 30

Table 2-2 Segmentation previous work approaches, segmentation accuracy and main disadvantages 35

Table 2-3 Line segmentation proposed approach features over other approaches 40

Table 2-4 Word/Sub-word segmentation proposed approach features over other approaches 41

Table 2-5 Character segmentation proposed approach features over other approaches 42

Table 4-1 Font types and styles that have been used in the local dataset 94

Table 4-2 Line segmentation results for regular style with different font sizes (local dataset). 95

Table 4-3 Line segmentation results for bold style with different font sizes (local dataset). 96

Table 4-4 Line segmentation results for italic style with different font sizes (local dataset). 96

Table 4-5 Line segmentation results summary (local dataset). 97

Table 4-6 Word segmentation results for regular style with different font sizes (local dataset). 98

Table 4-7 Word segmentation results for bold style with different font sizes (local dataset). 98

Table 4-8 Word segmentation results for italic style with different font sizes (local dataset). 99

Table 4-9 Word segmentation results summary (local dataset). 99

Table 4-10 Word segmentation results (APTID / MF dataset). 99

Table 4-11 Character segmentation results for regular style with different font sizes (local dataset) 101

Table 4-12 Character segmentation results for bold style with different font sizes (local dataset) 101

Table 4-13 Character segmentation results for italic style with different font sizes (local dataset) 102

Table 4-14 Character segmentation results summary (local dataset). 102

Table 4-15 Character segmentation results (APTID / MF dataset) 103

Table 4-16 Execution Time for different segmentation stages with and without diacritics. 103

Table 4-17 Character segmentation sample images. 104

Chapter 1 Introduction

This chapter highlights Arabic OCR usage in many applications, the development process and brief description for each one, OCR types, the motivation of this research, and the reason of why the segmentation stage is chosen from all Arabic OCR development process to be worked on. Arabic text characteristics, research objectives, and problem statement will be discussed, also a brief description of the proposed algorithms, and what have been done.

1.1 *Introduction and Motivation*

OCR stands for Optical Character Recognition, OCR converts the electronic scanned images into computer digital format which can be edited by any of text applications like Microsoft Office package programs [2]. The development of efficient and accurate Arabic OCR systems has become one of the most important and challenging tasks. Arabic OCR systems 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 from different sources of documents like invoices, receipts, bank statement, and passport document which can be done in faster and more accurate way, so the need of accurate Arabic OCR systems increases day by day [21].

In general, extracting text from an image can be divided into five fundamental steps: image acquisition, preprocessing, segmentation, feature extraction and classification. Figure 1-1 shows OCR development stages.

Image acquisition is the first step in the character recognition process. The goal of this step is to transform the input text into a digitized image. OCR systems are classified according to the way of taking their input; online OCR systems in which the input is taken by a pen writes on a flat bed, so the recognition is done while the writing operation is in progress like some tablet computers with pen and smart phones [25]. The second type is offline OCR systems in which the input is usually an already taken image by camera, scanner - the scanner is most commonly used as it is more convenient -, or other optical devices [5]. Two types of input images according to the way of writing are found: the handwritten text type and the printed text type [8].

The preprocessing step should cover all the functions prior to the feature extraction step so it can produce a cleaned-up version of the original image that can be used directly and efficiently by the segmentation method. The most popular methods used in the preprocessing step are binarization, filtering, smoothing, contrast for noise reduction and thinning. Binarization converts a grayscale image into a bi-level image depending on a certain threshold. Noise reduction helps in removing unwanted variations from the input image. Thinning is the process of reducing the width of the input text from many pixels to just only one pixel. This step aims at enhancing the input scanned image to avoid recognition mistakes due to the processing of irrelevant data [22].

The separation of writing into individual characters or segments is called segmentation.

OCR systems use different approaches for segmentation; the analytical approach and segmentation free approach or holistic approach [14], in the analytical approach each word is segmented to small components or parts that form the word like characters and diacritics[12], in this approach more processing is needed but more accurate results are obtained than using the segmentation free approach in which the recognition is done without segmenting the words to small parts but it uses some patterns and look-up dictionary for a certain or limited number of words like numbers and cities' names [10].

Character segmentation is not always producing separate characters [6]; in some cases, it is difficult to segment to single characters like in ligatures which is a combination of two characters or more like (**محمد**) in word (**محمد**) [9], also some characters are segmented to more than one segment like SEEN(**س**), SHEEN(**ش**), SAD(**ص**), DAD(**ض**), TAA(**ط**), and THAA(**ظ**) characters, in this case post processing step after initial segmentation is applied to get the correct characters.

Diacritics or strokes add additional source of errors in segmentation stage because of overlapping and the need to distinguish between them and dots which they are considered as a part of the sub-word, so extra work should be done.

The segmentation stage is a necessary step in recognizing Arabic characters. An error in segmenting the basic shape of the characters will produce errors in the identification of each character. Segmentation passes through line segmentation, word segmentation and character segmentation stages. Horizontal projection technique can be used for line segmentation, vertical projection technique can be used for word segmentation and over-segmentation with pixel tracking can be used

for character segmentation [23]. For sub words, contours can be used to separate them from the original word. After applying vertical and horizontal projections to segment words into characters, a low-level segmentation is applied for dots and other zigzagging features [24].

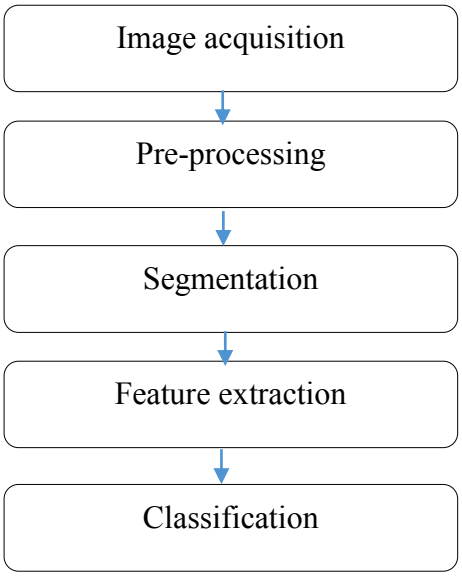


Figure 1-1 OCR Development Process

The segments resulting from the segmentation stage are identified in the feature extraction stage, by selecting unique features from letter or segment [16], statistical, discrete or structural features can be used at this stage; the representation of sequence of chain code is used in the statistical features by using simple vector holds these information, so the comparison can be done easily with candidate letter or character [30]. In the discrete features, there are specific features for Arabic characters are used as pre-classification step. Since they produce huge number of classes, the structural features use significant information about the letter where

each letter has a model holds most letter information according to its structure like using information from its contour [30].

The final stage is the classification which uses the extracted features to recognize the segments [12]. Many methods can be used at this stage like neural networks, k-nearest- neighbor and decision tree.

Arabic language is widely used in the world, it is the spoken language by millions of people, Arabic script is used in many applications and documents [3].

Arabic script has some characteristics which make the development of reliable OCR a challenging task especially the segmentation stage from which most of errors come and affect negatively the recognition rate, these features as following:

- It has a cursive nature and written/read from right to left [29].
- Baseline connects characters horizontally, and connection's points locate on. Figure 1-2 shows the location of the baseline; it is the line where most of the pixels locate horizontally [25].
- Overlapped connected components sometimes make the sub-words [31].
- Each character has different width and height from other characters [9].
- If the characters overlapped vertically they perform a ligature or combined character [9]. Figure 1-3 shows an example of ligature.
- The diacritics appear above or below the word which are called sounds or short vowels [20]. Figure 1-4 shows the diacritics of Arabic script.
- The shape of character depends on its location in the word (at beginning, middle, end or a standalone character) [1]. Figure 1-5 shows an example of

‘ayn (ع) character and how its shape differs according to its location in the word.

- Different font types and styles are found for printed Arabic text [10].

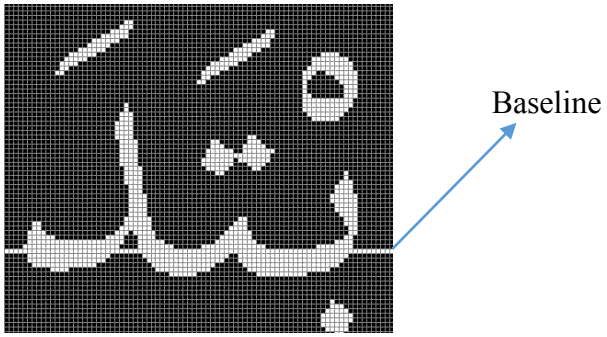


Figure 1-2 Baseline connects characters in the same sub-word

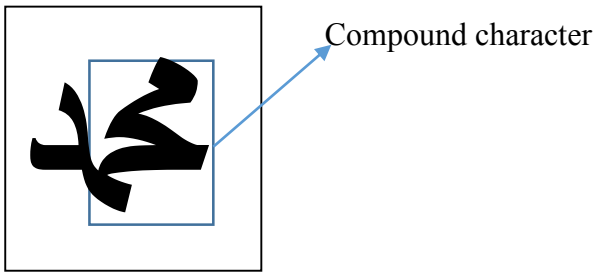


Figure 1-3 Ligature or compound character (multiple characters overlapped vertically)

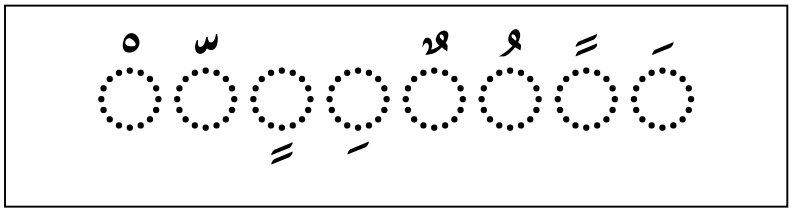


Figure 1-4 Diacritics in Arabic script

Table 1-1 Arabic characters all possible shapes according to their location in the word

Letter Name	Possible shapes			
	Alone	End	Middle	beginning
Alef	ا	ـا		
Ba’a	ب	ـب	ـبـ	بـ
Ta’a	ت	ـت	ـتـ	تـ
Tha’a	ث	ـث	ـثـ	ثـ
Jeem	ج	ـج	ـجـ	جـ
Ha’a	ح	ـح	ـحـ	حـ
Kha’a	خ	ـخ	ـخـ	خـ
Dal	د	ـد		
Thal	ذ	ـذ		
Raa	ر	ـر		
Zain	ز	ـز		
Seen	س	ـس	ـسـ	سـ
Sheen	ش	ـش	ـشـ	شـ
Sad	ص	ـص	ـصـ	صـ
Dad	ض	ـض	ـضـ	ضـ
Taa	ط	ـط	ـطـ	طـ
Thaa	ظ	ـظ	ـظـ	ظـ
Ein	ع	ـع	ـعـ	عـ
Gein	غ	ـغ	ـغـ	غـ
Faa	ف	ـف	ـفـ	فـ
Qaf	ق	ـق	ـقـ	قـ
Kaf	ك	ـك	ـكـ	كـ
Lam	ل	ـل	ـلـ	لـ
Meem	م	ـم	ـمـ	مـ
Noon	ن	ـن	ـنـ	نـ
Haa	ه	ـه	ـهـ	هـ
Waw	و	ـو		
Yaa	ي	ـي	ـيـ	يـ

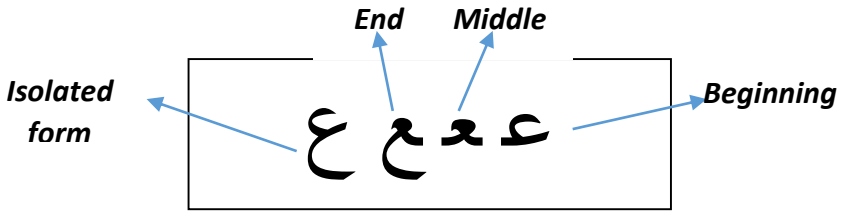


Figure 1-5 Different shapes for 'ayn (ع) character according to its location in the word (at beginning, middle, end, and isolated form)

Having a segmentation algorithm that deals with Arabic text with diacritics is essential for success of Arabic OCR system. The accuracy and the efficiency of OCR applications are dependent upon the quality of the input image and the segmentation stage.

1.2 *Research Objectives and Problem Statement*

Segmentation stage is the main source of errors in segmentation based Arabic OCRs, till now the research in this field is open and needs a lot of work. For example, dictionaries and holy books for Arabic language with diacritics need to be transformed to digital format, and the existence of diacritics on the texts of these books will poses an additional segmentation challenges which haven't been addressed in most of the existing researches, even in the commercial Arabic OCRs. Indeed, a research in Arabic field which is conducted by Birzeit University, needs an editable format of Arabic dictionaries with diacritics. Commercial OCR is used for this purpose but the result was bad. As a solution for this problem, these dictionaries were typed manually into Microsoft Word software by some people and this took a lot of efforts and time. In this research, we handled most of the issues resulted from segmenting Arabic typed text with diacritics through providing a set of solutions for the problems appear during the segmentation stages summarized as the following:

- Handle the overlapping problem between consecutive text lines in the line segmentation stage.
- Handle the overlapping between sub-words in the word/sub-word segmentation stage**.**
- Character segmentation with diacritics existence.

1.3 *Contribution*

Different segmentation approaches for offline recognition of printed text with diacritics for different Arabic OCR segmentation stages are enhanced in this thesis;

- A robust algorithm is proposed for line segmentation for Arabic printed text in the AOCR systems based on finding the global maximum peak and the baseline detection. Overlapping between lines and over-segmentation due to the diacritics existence in the Arabic text are the main problems that cause errors in this stage, and solved by the proposed algorithm. The algorithm is tested for different font sizes and types and encouraged results have been obtained.
- An enhanced method for word, sub-word and diacritics segmentation is also proposed. Each word consists of one sub-word or more. The sub-words are extracted in two ways according to the sub-words situation. Vertical projection is used in case of full separation between sub-words by finding the gaps between them, while the connected component concept is used to find the sub-words in case of overlapping. The overlapped sub-words are related to the same word [8], the connected components concept is used to

extract the diacritics from the diacritic image resulted from the vertical projection separation of the diacritic line image. The proposed method also determines if the sub-words are related to the same word or to different words regardless to the font type or size by estimating the pen size for each sub-word.

- Another issue is solved which is the diacritic segmentation of overlapped sub-words, the diacritics in the overlapped region between sub-words are also overlapped by column indices with the two sub-words so the diacritic is related to the sub-word with higher overlapping percentage.
- For the last segmentation stage, which is the character segmentation, another enhanced method is proposed based on contour extraction technique which has many advantages over other methods like having a clear description for character shape and details even for small fonts, also the errors in extracting the baseline are eliminated and no need to adjust the baseline many times [14]. The algorithm extracts the up-contour part then finds the splitting regions locate over it, the splitting regions perform local minimum areas in the up-contour part which are considered as initial cutting points on the sub-word. Over segmentation problem appears at this step, so post processing phase is needed to ignore some splitting regions which cause the over segmentation problem. Ignore cases checking algorithm is developed in easy and reliable way that can fit many font types and styles, based on the specifications of the character's part resulted from initial segmentation process, the location of the splitting region above or below the

baseline included the distance from it (to ignore last splitting region which is not a cutting point plus solving the over segmentation in TAA (ط) and THAA (ظ) characters), the space counter which is incremented in case of the specifications are satisfied and the character's part has no dots above or below (this detects the SEEN (س), SAD (ص), DAD (ض) and helps to detect SHEEN(ش) case which needs extra checking for having three dots). This gives more generality for the algorithm especially the shapes of characters differ according to the used font type. The algorithm shows good results up to 98%.

1.4 *Research Methodology*

The methodology of this research goes through two main steps:

- Data collection: two main sources for the data (datasets) are used for testing; the first one is APTID / MF dataset and the second one is generated locally.
- Data analysis: MATLAB software program is used for code implementation and obtaining the results.

The structure of the datasets and the used tools are described in details in chapter 4.

1.5 *Thesis Outline*

Chapter 2 discusses the pertinent literature and some approaches that already have been done on the Arabic text segmentation, the state of arts, and why this research is a long term one.

Chapter 3 outlines the research methodology which followed and detailed steps that have been done in the proposed algorithm.

Chapter 4 shows the testing and results including the tools, datasets description and different segmentation stages (line, word/sub-word, and character) results.

Chapter 5 shows the conclusion and future work.

Chapter 2 Background and Literature Review

2.1 *Background*

For the past couple of years, there has been increasing interest among researchers in problems related to the text recognition in general. Intensive research has been carried out in this area with a large number of technical papers and reports in devoted to character recognition. This subject has attracted research interest not only because of the very challenging nature of the problem, but also because it provides the means for automatic processing of large volumes of data in dictionary, books and postal code reading [2].

The segmentation research area is still open and still not matured, good segmentation process leads to a good recognition rate. Segmentation passed through several steps or stages shown in figure 2-1.

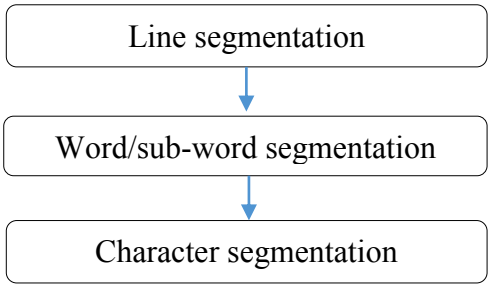


Figure 2-1 OCR segmentation stages

2.2 *Line Segmentation Related Work*

Line segmentation performs a significant stage in the AOCR systems; it has a direct effect on the next stages –word and sub-word segmentation- which affect the recognition rate directly.

The diacritics existence in the Arabic script causes the major problems in this step. For small fonts, the overlapping between lines problem appears while both over-segmentation and overlapping problems appear for large font sizes. In the over-segmentation case diacritics perform an independent line, these are the main problems which are highlighted and discussed.

Many methods are proposed for text to line segmentation and they are grouped to the following:

- Projection profiles methods

  Two main types of projection profiles; horizontal projection profiles and vertical Projection profiles [7], horizontal projection profile is used for line segmentation by finding the inter line gap which is considered as a separation region between two consequent lines. This way is efficient for printed text and when no overlapping or touching is detected between lines [15].

- 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 [28], then the white spaces or pixels between them are marked with black if the distance between them and the black pixels are less than a threshold value, the

boundaries of the 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 [17].

- Grouping methods

In this method, the connected components of black pixels are grouped, the grouping operation based on some properties like continuity, and similarity, it is more used for documents analysis [2].

- 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 by measuring the region properties, the boundaries for each line are determined [7].

- Hough transform methods

Hough peaks are determined and according to those peaks the lines are extracted. This method is mainly used for document analysis [2].

- 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 [2].

- Others

In 2003, Nawaz, Sarfraz, Zidouri and Al-Khatib proposed an algorithm for line segmentation using horizontal projection technique, the line is divided into three

zones; upper, lower and the baseline zone in which the black pixels perform the highest density [5].

In 2012, Elaiwat and Abu-zanona proposed an algorithm for line segmentation based on the summation of each row, where the summation is greater than zero it is considered as part of the line, while the rows that have the summation equals to zero it is considered as the end of the line, so the line locates between two rows that have the summation equals to zero [5]. In this assumption, the overlapping problem between lines is not taken into consideration, so the inter line gap should be clear.

In 2012, M. Alrefai et al. proposed an algorithm for line segmentation based on horizontal projection technique, this method based on detecting the small parts like dots and Hamza to be removed. In this method, processing is intensive and it doesn't deal with diacritics [4].

In 2013, M. Alipour. proposed a line segmentation algorithm based on the horizontal projection profile and used a predefined threshold to detect the separation between two lines, if black pixels' difference between the two lines is less than this threshold, the region is considered as a separation between two lines [14]. The problem in this method is the using of a predefined threshold, so it works for predefined and special cases.

The proposed algorithm in this research used the horizontal projection profile technique, since the documents that are used in this research are printed type documents and the assumption that there is no skew in the input image, so we can benefit from the horizontal projection technique to avoid much processing that is

needed in the document analysis techniques, the proposed algorithm needs extra processing just when the overlapping case is detected. This method can be used for different font types, sizes and styles.

2.3 *Word / Sub-word Segmentation Related Work*

Word/sub-word segmentation is an important step in the segmentation phase for segmentation based AOCR systems. In this step, each word and its parts (sub-words) are extracted.

In 2000, A. Cheung, et al. introduced a new segmentation algorithm that uses a technique in which the overlapping Arabic words/sub-words is horizontally separated, it also uses 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 and the second process uses the convex dominant points (CDPs) detection algorithm developed by Bennamoun. The recognition accuracy reached up 90% [16].

In 2008, Jawad H AlKhateeb, et al. proposed a new method for baseline detection and extracting the connected components for the sub-word. The baseline is located below the middle line of an image, after that the peak was checked to determine the baseline. An iterative process was used to detect the connected components based on the connected black pixels in the sub-word and the results were very encouraging; more accurate results for baseline detection and word segmentation were achieved [26].

In 2009, Noor Ahmed Shaikh, et al. suggested an algorithm for Sindhi text segmentation, the text was thinned using Shaikh, Z.A. thinning algorithm. Horizontal projection was used for text to line segmentation, then the baseline was detected using the horizontal projection of the extracted line of text, finally the sub-word was extracted using connected components extraction method. The algorithm failed in case of overlapped characters.

In 2012, M. Alrefai et al. proposed an algorithm for word/sub-word segmentation based on vertical projection technique, the small parts like dots and Hamza have to be removed and this is computationally expensive, also it doesn't deal with diacritics and different font sizes, the threshold value should be determined before [4], the proposed algorithm in this thesis works for different font types and sizes dynamically.

In 2013, M. Alipour. proposed an algorithm for word/sub-word segmentation based on the vertical projection profile, a predefined constant k is used which holds the value equals half of the line height, the separation region holds the value of k consecutive zero's [14]. The problem in this method is the using of predefined value which will be not suitable for all different word sizes and styles.

The notice from these works that the word and sub-word segmentation cannot come as independent work in most cases, it comes as pre-stage for character segmentation. Table 2-1 shows some approaches for line and word/sub-word segmentation. Many of these methods failed in solving the overlapping between sub-words, and some depend on the line height to come with equations that

determine the threshold value between two independent words, in this case one font size is used in the input line; on the contrary, the proposed algorithm in this research generates the threshold value regardless the font size, type or style used in the input line.

Table 2-1 Line and word/sub-word segmentation previous work approaches

Year/Reference	**Method based on**	**Accuracy on line segmentation**	**Word / sub-word segmentation accuracy**
2010/ [2]	Horizontal and vertical projection, Recursive middle line extraction, Component contour tracing	99.5%	99.54%
2012/ [5]	Horizontal projection	97.3%	96.3%
2006/ [6]	Modified horizontal projection method	97.85%	Not Reported

2.4 *Character Segmentation Related Work*

Many methods are proposed for Arabic OCR character segmentation and they are classified into: projection profile methods, character skeleton based methods, contour tracing based methods, template matching based methods, morphological operations based methods and recognition based segmentation methods, and each of these methods has advantages and disadvantages [1].

- Methods based on projection profiles are usually used for lines, words and sub-words segmentation, when a clear gap is found between them. Horizontal projection is used for line segmentation and vertical projection is usually used for word and sub-word segmentation. When this method is used for character segmentation, the segmentation region is thinner than around regions by applying the vertical projection [19].
- Methods based on contour tracing: in this method, 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 appear in the thinning because it depends on extracting the structure of the word, which gives a clear description for it, but they are affected by the noise, so some enhancements also should be applied [1].
- Methods based on morphological operation: in this method, morphological operations are used for segmentation, usually closing followed by opening operations are applied. This method is not an independent mothed because other techniques should be used beside for segmentation. Little number of researchers used this method [30].
- Methods based on template matching: in this method, usually a sliding window slides over the baseline is used, if any match is noticed then the center pixel in the sliding window is considered as the cutting point. The problem in this method is if the cutting point locates under the baseline, a segmentation failure will be occurred [1].

In 2000, A. Cheung, et al. introduced a new segmentation algorithm that uses a technique in which the overlapping Arabic words/sub-words 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, and the second process uses the convex dominant points (CDPs) detection algorithm developed by Bennamoun. The recognition accuracy reached up 90% [16].

In 2004, Mostafa G. Mostafa developed a new segmentation approach for printed Arabic text especially for "Simplified Arabic" font with different sizes. The main rule used is that "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 [12].

In 2005, M. Omidyeganeh, et al. presented a new segmentation algorithm based on conditional labeling for up and down contours. The algorithm was developed for multi font Farsi/Arabic texts. The contour of sub-word is measured by using convolution kernel with Laplacian edge recognition-based segmentation detection method. The algorithm goes through Contour labeling of each sub-word and 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 [11].

In 2009, Noor Ahmed Shaikh, et al. suggested an algorithm for Sindhi text segmentation, the text was thinned using Shaikh, Z.A. thinning algorithm. Horizontal projection was used for text to line segmentation, then the baseline was detected using horizontal projection of the extracted line of text, finally the sub-word was extracted using connected components extraction method. The algorithm failed in case of overlapped characters. Height Profile Vector (HPV) used in [13] for characters' extraction. 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 over segmentation, these faults in the algorithm made it not solid and need extra work to solve the under and over segmentation problems.

In 2010, Sobia T. Javed, et al. developed a free segmentation approach for Urdu script, different pattern matching techniques were used to classify the pattern. The features were extracted from the image and fed them to HMM recognizer, which has an ability to perform recognition with great ease and efficiency. The algorithm was tested giving an accuracy of 92% for total 3655 ligatures, 3375 ligatures are accurately identified [27].

In 2013, Mohammad Alipour improved segmentation method of the Persian script. Some structural features were used to adjust the 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 extra step by merging the small fragments, this step was necessary in the cases of one character is segmented into more one part like SEEN (س), SHEEN

(ش), SAD (ص) and DAD (ض). The algorithm achieved 98.02% as an average segmentation accuracy [14].

Table 2-2 summarizes the previous work approaches, recognition rates and main disadvantages. The proposed approach benefits directly from the contour method by using only the up-contour part as shown in Figure 2-2.

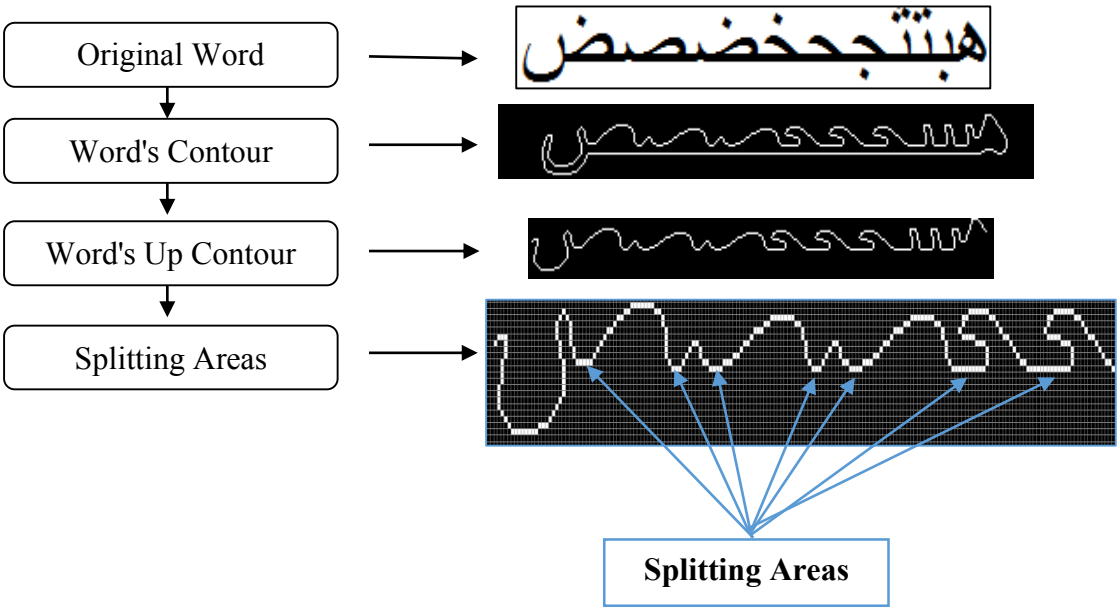


Figure 2-2 Splitting areas extracted from the up-contour

Table 2-2 Segmentation previous work approaches, segmentation accuracy and main disadvantages

Year/Reference	Approach	Segmentation Accuracy	Disadvantage
2004 - [16]	Segmentation Based	96.5%	The algorithm was used only for 'Simplified Arabic' font type
2005 - [11]	Segmentation Based	97%	- A pre-processing technique was used to adjust the local base line for each sub-word. - A post processing stage was used to adjust the segmented characters
2008- [26]	Segmentation Based	Not reported	An iterative process was used to detect the connected components based on the connected black pixels in the sub-word - overhead-
2009 - [13]	Segmentation Based	Not reported	Some faults in the algorithm were registered, and extra work was needed to solve the under and over segmentation problems

2010 - [27]	Holistic Based	92%	Each time the features were extracted from the image they fed to some recognizer for identification purpose, and this led to extra processing.
2013 - [14]	Segmentation Based	98.02%	-The word fragmentation was done without considering dots and other signs of the word's characters - Adjustment is done for merging small fragments that are parts of a character

2.5 *Concepts Background*

Binary image: a black white image form, where the data is represented by 0 value (black, no data) and 1 value (white). The binary image in MATLAB program is represented by matrix or two-dimensional array, as shown in figure 2-3.

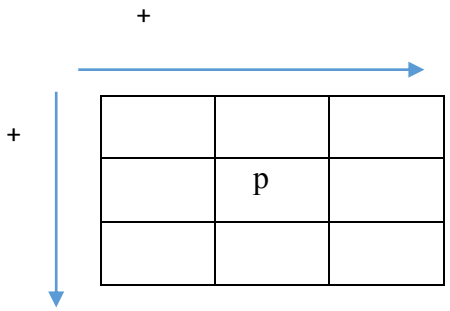


Figure 2-3 Binary image representation

Pixel: the smallest part of the image that represented by one location in the image' matrix. Each pixel in the binary image has 8 faces or edges to be connected with other pixels.

Horizontal projection: the summation values of data in the matrix rows that represents the image. The inter gap represents zero output for the summation.

$$\boldsymbol{H_{Proj}} = \sum_{\boldsymbol{j}} \boldsymbol{p(i,j)}\,; \qquad (2\text{-}1)$$

Vertical projection: the summation values of data in the matrix columns that represents the image, the inter gap represents zero output for the summation.

$$\boldsymbol{V_{Proj}} = \sum_{\boldsymbol{i}} \boldsymbol{p(i,j)}\,; \qquad (2\text{-}2)$$

Connected components: groups or clusters of pixels that connected to each other in a certain way. The connectivity value is determined by the number of faces or edges

for the pixel inside the group, for example, the pixels inside one connected components can connect with each other using 8-edge connectivity, in which one of edges for every pixel should be connected to another.

Sub-word: part of a word that performs one connected component.

Word: part of text line that consists of one or more sub-words.

Sub-word's main body: the sub-word without diacritics that performs the largest connected component in the sub-word.

Pen size: the thickness of pen for writing. It usually performs the most frequent value of the vertical projection, as shown in figure 2-4.

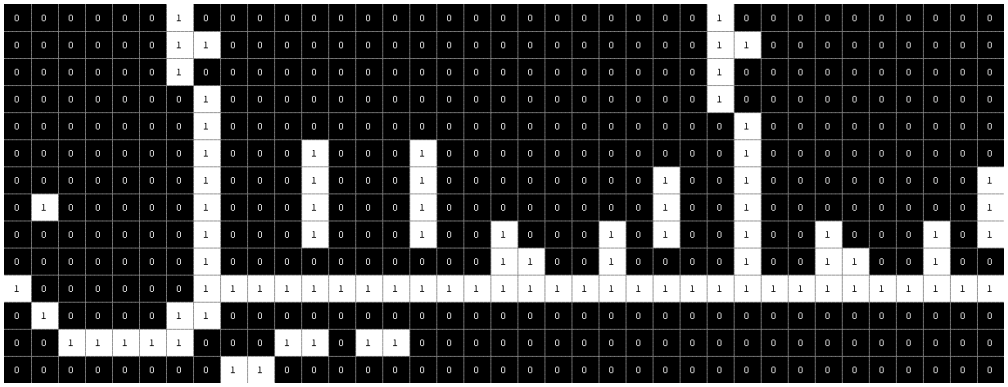


Figure 2-4 Pen size = 1

2.6 *State of Art*

Segmentation algorithms are developed for different segmentation stages (line segmentation, word/sub-word segmentation, diacritics segmentation and character segmentation). Each one of them has advantages over the previous approaches. For line segmentation algorithm, it uses the horizontal projection technique with additional steps to detect the overlapping cases by respectively estimating the global

maximum peak, finding the expected line height to find the lines count, and splitting the segment into lines. This method avoids much computation time which is found in document analysis methods. Some of these methods failed in case of full overlapping between two lines like the run length smearing method. Most of the methods that used horizontal projection assume the inter line gap between lines is clear, or a pre-define threshold value which can't be suitable for different font types, styles and sizes.

For word/sub-word segmentation the proposed method used the vertical projection technique for separation, then it computes the pen size to determine if the separation region locates between two sub-words in the same word or in different words. The algorithm used the connected component technique to extract the overlapped sub-words. After that, the diacritics are extracted and assigned to its related sub-word even in case of overlapping by estimating the overlapping percentage at the column level between each diacritic and the overlapped sub-words. The diacritic belongs to the sub-word with max overlapping percentage.

For character segmentation, the proposed algorithm uses the contour finding approach. This approach has many advantages over finding the skeleton of the word, in which word's information can be lost, and that leads to less recognition rate.

Imcontour () MATLAB function is used with some enhancements, then the up-contour is extracted by electing two points as beginning and end points on the original contour. The up-contour is formed by tracking the path from the beginning point to the end point in counterclockwise direction. After extracting the up-

contour, it goes for extra processing and new algorithm is applied to find the splitting areas over the up-contour. Some of these splitting areas should be ignored like in case of SAD (ص), DAD (ض), SEEN (س) and SHEEN (ش). For this reason, these splitting areas are passed to a new algorithm to determine whether to be ignored or not. In this algorithm, some rules are applied to determine the SAD (ص), DAD (ض), SEEN (س) and SHEEN (ش) characters.

2.7 *The Proposed Approaches Features over the Previous Approaches*

Table 2-3 Line segmentation proposed approach features over other approaches

Proposed Approach	**Previous Approaches**
Benefits from the horizontal projection technique then some enhancements are added to get the wanted results.	Horizontal projection technique is used with assumption that no overlapping no touching found.
Less computation time is needed than the document analysis techniques, since the documents contain printed text only.	The document analysis methods that used for line segmentation like the grouping, thinning and run length smearing needs much computation time.
No need for a predefined threshold value to segment the text into lines like line height or the distance between two adjacent lines. The document may	Many techniques that used the horizontal projection for overlapped lines use a predefined value like line width or the distance between two

contain different lines height and different font types and styles.	consequent lines, and that doesn't work for documents that include many font sizes and styles.

Table 2-4 Word/Sub-word segmentation proposed approach features over other approaches

Proposed Approach	**Previous Approaches**
It solves the problem of overlapping between sub-words using the connected component technique.	Many of the proposed approaches fail in solving the connected component problem.
It uses the vertical projection technique to separate the words and sub-words that have a clear gap between them, and determines which sub-words are related to the same word without using a predefined or a threshold value.	Many of these techniques uses a predefined threshold value to determine if the splitting region is between two different words or between two sub-words related to the same word.
It is used for different font types, styles, and sizes by calculating the pen size for each sub-word.	Most of the used algorithms are proposed for a certain font type or style.

Table 2-5 Character segmentation proposed approach features over other approaches

Proposed Approach	Previous Approaches
Gives a clearer and better description for the structure of the sub-word.	In the skeleton approach the data and the structure of the sub-word can be lost in some cases, since the shape is reduced to one-pixel width.
The splitting is done without the need to compute the baseline. This approach overcomes this problem by finding the splitting areas in the up-contour, and then applying some rules to ignore the extra ones.	The up and down contours were extracted to calculate the baseline, and sometimes the algorithm failed in determining the baseline. In this case, the baseline was adjusted in extra processing step.
There is no need for extra processing to be done to label each pixel on the contour, and no need to apply the full state diagram to label it. Also, there is no need to do the contour curvature grouping.	Up contour labeling was performed to determine each pixel in the contour if it is above or below the extracted baseline, and this took much processing.
It solves many problems like the touching between sub-words. In this case, the region of touching performs a splitting area.	This cannot be done by other approaches.

Chapter 3 Proposed Work

In this chapter the proposed methodologies for different segmentation stages - for segmentation based OCRs - are discussed in details. Figure 3-1 shows different segmentation stages.

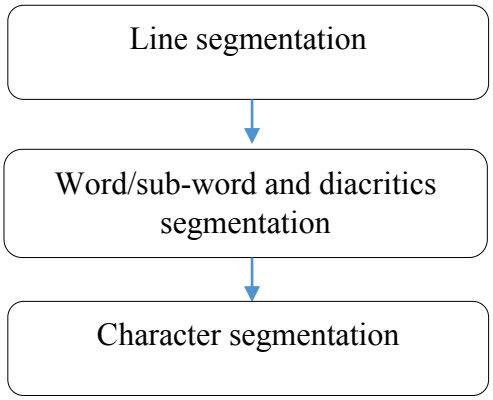


Figure 3-1 OCR segmentation steps

The used documents are classified into two groups according to the existence of the diacritics:

- Documents with diacritics (tashkel), as shown in figure 3-2.

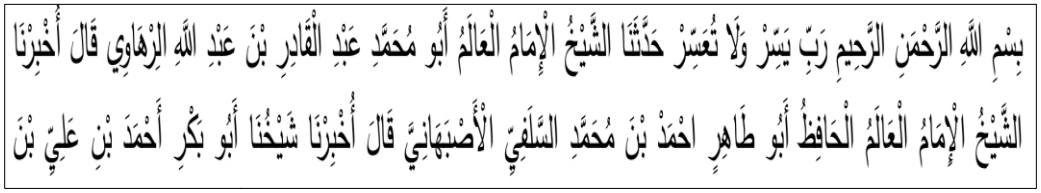

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ رَبِّ يَسِّرْ وَلَا تُعَسِّرْ حَدَّثَنَا الشَّيْخُ الْإِمَامُ الْعَالِمُ أَبُو مُحَمَّدٍ عَبْدِ الْقَادِرِ بْنَ عَبْدِ اللَّهِ الرِّهَاوِي قَالَ أُخْبِرْنَا
الشَّيْخُ الْإِمَامُ الْعَالِمُ الْحَافِظُ أَبُو طَاهِرٍ احْمَدْ بْنَ مُحَمَّدٍ السَّلَفِيِّ الْأَصْبَهَانِيَّ قَالَ أُخْبِرْنَا شَيْخُنَا أَبُو بَكْرٍ أَحْمَدَ بْنِ عَلِيِّ بْنَ

Figure 3-2 Arabic text with diacritics

- Documents without diacritics (tashkel), as shown in figure 3-3.

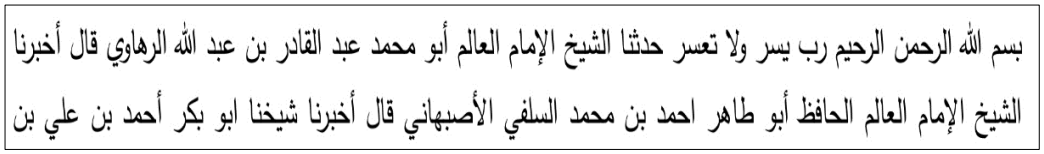


Figure 3-3 Arabic text without diacritics

3.1 Text to Line Segmentation

Line segmentation is a common step between segmentation based and free based OCRs, the diacritics existence makes the segmentation to individual lines a complicated task; it causes two main problems:

- Overlapping between lines [6]. In this case the inter line gap disappear.
- Over segmentation problem, in which the line appears consisting of diacritics only, especially for large fonts [28].

The proposed algorithm overcomes these problems, the assumption is that lines are not skewed, the algorithm based on the horizontal projection technique, the base line detection (global maximum peak), and finding the local maximum peaks. Figure 3-4 shows the text to lines segmentation algorithm stages.

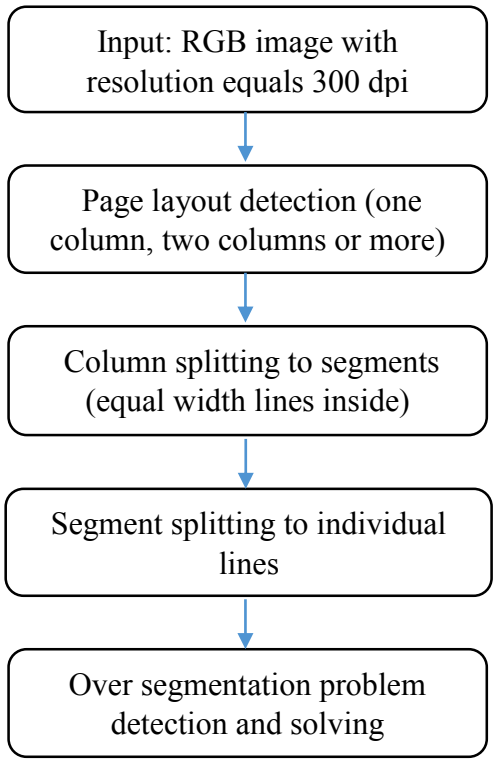


Figure 3-4 Line segmentation algorithm stages

3.1.1 *Input Image*

The input is an RGB image, which is converted to grayscale format [18] using rgb2gray () MATLAB function, in this function, the brightness information is obtained by merging RED, GREEN and BLUE in the true color image, with ratio reach up to 30%, 60% and 11% for each one respectively, then the image is cropped and passed to the next stage to detect the page layout.

3.1.2 *Page Layout Detection or Segmentation*

Page layout detection performs a pre-processing step for extracting the document's lines. The page organization is detected by finding the columns that the page

consists of (one column, two columns or more). Figure 3-7 shows the flowchart for page layout detection, and these steps are clarified by algorithm 1 as the following:

- Image binarization - image conversion to black white image (binary image) -. The size of the data to be processed is being minimized for ease of estimations.
- Enhance the image using an opening technique to reduce the interlacement and the overlapping between the lines that caused by the diacritics.
- Vertical projection is then applied – In this step the regions with no data are detected by finding the indices with zero projection – , these indices are the initial separation regions between page's columns; then the size of each region is calculated and compared with a threshold value to be considered as a separation region (the length of the separation region is greater than a certain threshold value). Figure 3-5 shows the separation regions where the vertical projection equals to zero.
- The page's columns are located between these separation regions.

The output is a list of columns that the layout of the page consists of. Figure 3-6 shows the output of this stage.

Algorithm 1: Page Layout Detection and Segmentation

Input: Binarized cleaned Image (CImage)
Output: List of columns the page layout consist of
begin
  //vertical projection is applied
  $V\_proj \leftarrow Vertical\_projection(CImage))$

  //indices where vertical projection is equal to zero are detected and these are the separation regions indices .
  $Separation\_indices \leftarrow V\_proj==0$

  //every set of separation indices are grouped and each group is then considered as separation region.
  $continuous\_regions \leftarrow grouped\_continuous\_indices(Separation\_indices)$

  //filter separation region.
  $separation\_regions \leftarrow continuous\_regions > Threshold$
  for $i \leftarrow 1$ ***to*** $length(separation\_regions)$ **do**
    //page column min limit
    $column\_min\_index(i) \leftarrow min(separation\_regions(i))$

    //page column max limit
    $column\_max\_index(i) \leftarrow max(separation\_regions(i+1))$

    //page column part
    $column(i) \leftarrow Igray(:, column\_min\_index(i): column\_max\_index(i))$

    //add the column to columns' list
    $ListOf(columns).Add(column(i));$
  end
  return $ListOf(columns)$
end

لماذا تونس ليست مصر؟

الثلاثاء 30/12/1436 هـ - الموافق 13/10/2015 م) آخر تحديث (الساعة 0:02
بمكة المكرمة)، 21:02 غرينتش)

فهمي هويدي

قول المجتمع المدني في تونس بجائزة نوبل للسلام يثير عديدا من الأسئلة حول أزمة المجتمع المدني في مصر وكونه حاضرا وغائبا طول الوقت. منذ قامت الثورة في عام ٢٠١١ ألقت الناس في مصر والعالم كله صور السياسيين التونسيين الذين تصدروا واجهات العمل العام في الرئاسة والحكومة والأحزاب والمجلس التشريعي. وابتداء من يوم السبت الماضي انضافت إلى الذاكرة البصرية صور أخرين لم نسمع بهم من قبل، نشرت جريدة" الأهرام "حوارا مع وجه غير مألوف لقارئ المصري كان منتوب الجريدة قد اجرى معه حوارا في العام الماضي. عرفنا أن اسمه حسين العبيدي، وأنه كان محاميا سابقا في إحدى المدارس التونسية لكنه انتخب رئيسا لاتحاد الشغل المقابل لاتحاد العمال عندنا .(وانشرت صحيفة "الحياة "اللندنية على صفحتها الأولى صورة لم نرها من قبل لسيدة عرفنا أن اسمها وداد بوشماوي الشهيرة فاضحكة وهي تلوح بقبضة يدها بعد إعلان خبر الجائزة .وفهمنا أنها رئيسة الاتحاد التونسي للصناعة والتجارة الذي كان ضمن الرباعية الفائزة .تناقلت مواقع التواصل الاجتماعي صور الثنين أخرين رأيناهما لأول مرة. أحدهما عبد الستار بن موسى رئيس الرابطة التونسية لحقوق الإنسان وبشير الصيد رئيس نقابة المحامين .وعين النصف الأربعة التي قائمة نجوم المرحلة، فإن ذلك أثار الاهتمام بالتعرف على الجهات التي يمثلونها، ليس فقط بدافع الفضول، ولكن أيضا لأننا انتبهنا إلى أن لدينا منظمات مماثلة ولكننا لا نتمس لها أثرا فلا نعرف ماذا تفعل، فضلا عن أننا نجهل أسماء وصور القائمين عليها .

استثنى من ذلك المجلس القومي لحقوق الإنسان الذي عينته الحكومة وله دوره المشهود في تجميل المؤسسة الأمنية، استثنى أيضا بعض المنظمات الحقوقية المستقلة التي لا نقرأ عنها إلا أخبارا بعضها ينتقد الانتهاكات والبعض الاخر يتعلق بالتحقيقات التي تجرى مع أعضائها .

أكثر ما أثار انتباهي حين تتبعت خلفيات المنظمات التونسية الأربع عدة أمور، أولها أن الثنين منها تشكلت قبل استقلال تونس في عام ١٩٥٦ ،فاتحاد الشغل تشكل في عام ١٩٤٦ ، واتحاد الصناعة والتجارة تأسس في عام ١٩٤٧ .أما نقابة المحامين فقد تأسست بعد الاستقلال، في عام ١٩٥٩ (ورابطة حقوق الإنسان تم تأسيسها عام ١٩٧٧ .وفي تفسير ذلك قال الكاتب والناشط التونسي صلاح الدين الجورشي إن المنظمتين الأوليتين تشكلتا في مرحلة الصراع ضد الاستعمار الفرنسي ،فالأولى كانت وعاء جمع العمل والثانية اجتمع فيها التجار واصحاب الأعمال .أما المنظمتان الأخيرتان فقد برزتا في طور الدفاع عن الحقوق واستقلالها من السلطة الوطنية .من الملاحظات الأخرى الجديرة بالذكر أن المنظمات الأربع قلت أغلب الوقت وعاء اجتمعت فيه مختلف القوى السياسية ،رغم التقلبات التي طرأت على قيادتها، إلى جانب أنها حافظت على استقلالها عن السلطة وأصرت على ذلك، حتى أن اتحاد الشغل رفض هيمنة الحزب الدستوري عام ١٩٧٨ واصطدم بالسلطة في عهد الرئيس الأسبق الحبيب بورقيبة، ولهذا السبب أمر الرئيس بنزول الجيش ومواجهة المتظاهرين في مواجهة أسفرت عن مقتل ١٧٨ شخصا، في واقعة ما زالت حاضرة بقوة في الذاكرة الجمعية التونسية .

مما ساعد على قوة تلك المنظمات أن نظامي الرئيسين بورقيبة وبن علي كان تركيزهما أكبر على مواجهة تحدي الأحزاب السياسية واعتبرا أن معظمها الأكبر ضد حركة الاتجاه الإسلامي) النهضة حاليا (إلى جانب المناضلين الوطنيين .وكانوا ما سعى إلى احتواء منظمات المجتمع المدني التي استمدت قوتها من شعبيتها من ناحية، والدعم الأوروبي من ناحية ثانية .لذلك أن أغلب تلك المنظمات احتفظت بعلاقات وثيقة مع نظيراتها الأوروبيين لاسباب جغرافية مفهومة. الأمر الذي جعل السلطات المحلية تتردد كثيرا في محاولة الإطاحة بها .

ثورة ٢٠١١ اعتبرت أن منظمات المجتمع المدني شريكة فيها ،إذ كان أول رئيس منتخب بعد الثورة، الدكتور منصف المرزوقي، ناشطا حقوقيا بارزا وربما سابقا للمجلس الوطني للحريات. كما استحدث منصب الوزير المكلف بالعلاقة مع الهيئات الدستورية والمجتمع المدني وعين له الدكتور كمال جندوبي الذي بدوره من نشطاء حقوق الإنسان وكان عضوا في الشبكة الأوروبية المتوسطة لحقوق الإنسان منذ عام ٢٠٠٣ .ولم يكن مستغربا والأمر كذلك أن يصدر بعد الثورة قانون يطلق حرية العمل لكافة منظمات المجتمع المدني .

الدور الوقائي الذي قامت به منظمات المجتمع المدني برز بشدة حين حدث الاستقطاب السياسي في تونس أسوة بما حدث في مصر، إذ انقسم الفاعلون السياسيون إلى معسكرين ،احدهما تمثل في معسكر الترويكا التي تحكم(حركة النهضة والمؤتمر من أجل الجمهورية، والتكتل الديمقراطي ،(المعسكر المقابل ضم احزاب المعارضة التي كانت خليطا من الليبراليين واليساريين والقوميين) ١١ حزبا مثلت في البرلمان .(انتهب الموقف بعد اغتيال اثنين من المعارضين) شكري بلعيد ومحمد البراهمي (فانتهزت المعارضة الفرصة وسعت إلى استثمار صدمة الشارع التونسي من خلال الدعوة إلى التظاهر والاعتصام .ونادى بعض الغلاة الذين انخرطوا فيما سمي الجبهة الشعبية بقيادة حزب العمال الشيوعي باستدعاء الجيش للتدخل بحجة حسم الصراع وحماية الثورة .

في الوقت ذاته لوحت الترويكا باعتقال الداعين الى الانقلاب على الشرعية .وبدا في الأفق أن الموقف على وشك الانفجار وأن دائرة العنف مرشحة للاتساع .وفي حين كانت اتصالات السياسيين مستمرة لإنقاذ الموقف، تحركت منظمات المجتمع المدني الرئيسية لكي تستخدم رصيدها التاريخي وثقلها الاجتماعي لإنجاز الدستور والبحث عن مخرج من الأزمة، من خلال اقتراحات المطالب والمبادئ على أطراف المعادلة السياسية .وفيما علمت فإن الفكرة لم يكن داخليا فقط ولكنه كان خارجيا أيضا .لذلك أن السفراء الغربيين أعربوا عن مخاوفهم من أن ينتكس الوضع في تونس، وأن تلقى تجربة التحول الديمقراطي المصير المحزن الذي منيت به في أقطار أخرى .وقبل أمي ان جميع السفراء الأوروبيين ذهبوا بصحبة السفير الأمريكي في تونس إلى مقر الاتحاد التونسي للشغل للاعراب عن دعمهم وتاييدهم لإنجاح المسار الديمقراطي .النتيجة أن جهد الرباعية وتجاوب احزاب الأغلبية السياسية وتشجيع الجماهير الغربية، ذلك كله اسهم في تغليب الوفاق واستبعاد عوامل الانفجار واستمرار مسيرة الثورة .لذلك استحقت الرباعية جائزة نوبل علما بأن الرئيس التونسي الباجي قايد السبسي كان أحد الثنين وشعود المنظمات الأربع لتنيل جائزة السلام .

ولان" الشيخين "السبسي زعيم حزب النداء وراشد الغنوشي زعيم حركة النهضة لعبا الدور الأكبر بين السياسيين في تحقيق الوفاق المنشود فان مجموعة الأزمات الدولية منحتهما جائزة السلام للعام الحالي، التي تعد قريبة من نوبل .ولكن رجح الخير الأول حجب خبر جائزة الشيخين فلم يتل حظه من الذيوع والانتشار .كما ان المقادير أن يتزامن إرسال الرئيس التونسي خطاب ترشيح الرباعية لجائزة نوبل، مع صدور تعليمات في مصر بمنع عدد من العاملين بمنظمات المجتمع المدني من السفر إلى الخارج .التعليمات جاءت بناء على قرار اصدره قاضي التحقيق في قضية التمويل الأجنبي المفتوحة في مصر منذ عام ٢٠١١ ، ولم يفك حتى الوقت الراهن، لذلك اصدرت ١٩ منظمة حقوقية مستقلة بيانا في يناير الماضي ادان حظر سفر النشطاء بناء على إجراءات غير معروفة ودون معرفة الاتهامات الموجهة اليهم .كما انتقد البيان الإجراءات التعسفية التي اتخذت بحق بعض المنظمات الأهلية، التي كان منها اقتحام مقراتها واستدعاء العاملين فيها للتحقيق واصدار احكام بسجن بعضهم مدة خمس سنوات مع إيقاف التنفيذ .

هذا الإجراء مجرد نموذج واحد لسبل التضييقات القانونية والأمنية التي لاحقت المنظمات الحقوقية المستقلة بعد قيام الثورة في عام ٢٠١١ .وقد بدأت التضييقات بحملة تشهير روجت للادعاء بان الثورة كانت مؤامرة أسهم الخارج في تدبيرها من خلال تلك المنظمات .وهو ما فتح الباب لتشكل لجنة لتقصي حقائق التمويل الاجنبي للمنظمات الأهلية، وادى الى إحالة ٣٥ منظمة حقوقية إلى التحقيق في شهر مارس/آذار عام ٢٠١١ .واستمرت الحملة من خلال التعديل الذي ادخل على قانون العقوبات باضافة المادة ٧٨ التي سميت قانون" الاشياء الأخرى "وهي التي فتحت الباب واسعا لمعاقبة نشطاء المنظمات الحقوقية بالسجن المؤبد وغرامة نصف مليون جنيه إذا ما تلقوا اي عون يمكن أن يؤدي إلى الإضرار بالمصلحة الوطنية، وهو التعبير الفضفاض الذي يمكن أن يشمل اي انتقاد لانتهاكات حقوق الانسان .

القوانين التي صدرت بعد ذلك كرست التصنيف وكبلت المنظمات الحقوقية بمختلف القيود .وكان ذلك أوضح ما يكون في قانوني الكيانات الإرهابية ومكافحة الإرهاب . بل إن تحقيق قضائي اجري مع بعض النشطاء والباحثين لانهم تلقوا مشروعا لإصلاح التشريعات لمنع التعذيب في مصر، وهو التحقيق الذي تعرض له الناشط نجاد البرعي، مدير المجموعة الدولية للمحامين .مع الثنين من القضاة وكانت نتيجة تلك التصنيفات أن قلصت بعض المنظمات الحقوقية من انشطتها، كما نقل مركز القاهرة لدراسات حقوق الإنسان مقره الرئيسي من القاهرة إلى تونس .في السياق الذي نحن بصدده فان الفرق الجوهري بين تونس ومصر أنهم هناك تعاملوا مع المنظمات الحقوقية باعتبارها شريكة في الثورة .اما في مصر فقد اتهمت تلك المنظمات أنها تآمرت في عملية اغتيال الثورة، وهذا الامر يعني أنهم في تونس تعاملوا مع المنظمات الحقوقية من منظور معاكس تماما لتوضع في مصر، لذلك كان طبيعيا أن ينتشل الشريك هناك، وأن يخبو دوره وينكمش نظيره في مصر .هذا رأي اثنين من النشطاء هما الأستاذان نجاد البرعي، مدير المجموعة الدولية للمحامين .ومحمد زارع، مدير مكتب مركز دراسات حقوق الإنسان بالقاهرة .يضيف الأستاذ جمال عيد، مدير الشبكة العربية لحقوق الإنسان . أن الشطة في مصر ليست ضد المنظمات الأهلية بالمطلق، وانما تقبل وتشجع انشطتها في مجالات الخدمات والأعمال الخيرية فقط .ومشكلتها الحقيقية مع المنظمات الحقوقية .ورغم أن المنظمات الأخيرة في الظروف العادية تقف في نقطة وسط بين السلطة والمجتمع، فانها في ظل الأوضاع غير الديمقراطية تفرض عليها مسؤولياتها أن تقف في صف المعارضة .وهو يلخص رايه في أن مشكلة المنظمات الحقوقية تكمن في انه لا توجد ارادة سياسية للاصلاح .وغياب تلك الإرادة هو العنصر الحقيقي لأزمة المجتمع المدني في مصر، الأمر الذي يعني أن الأزمة جزء من مشكلة أكبر تعاني منها مصر، لا توحي الدلائل بامكانية حلها في الأجل المنظور .

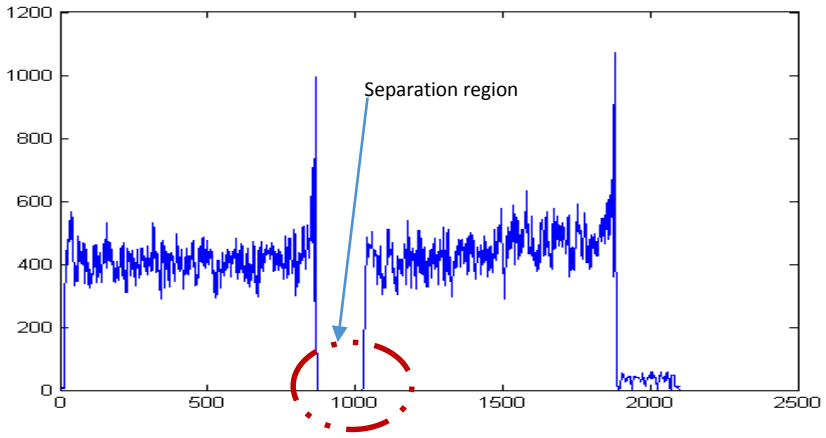
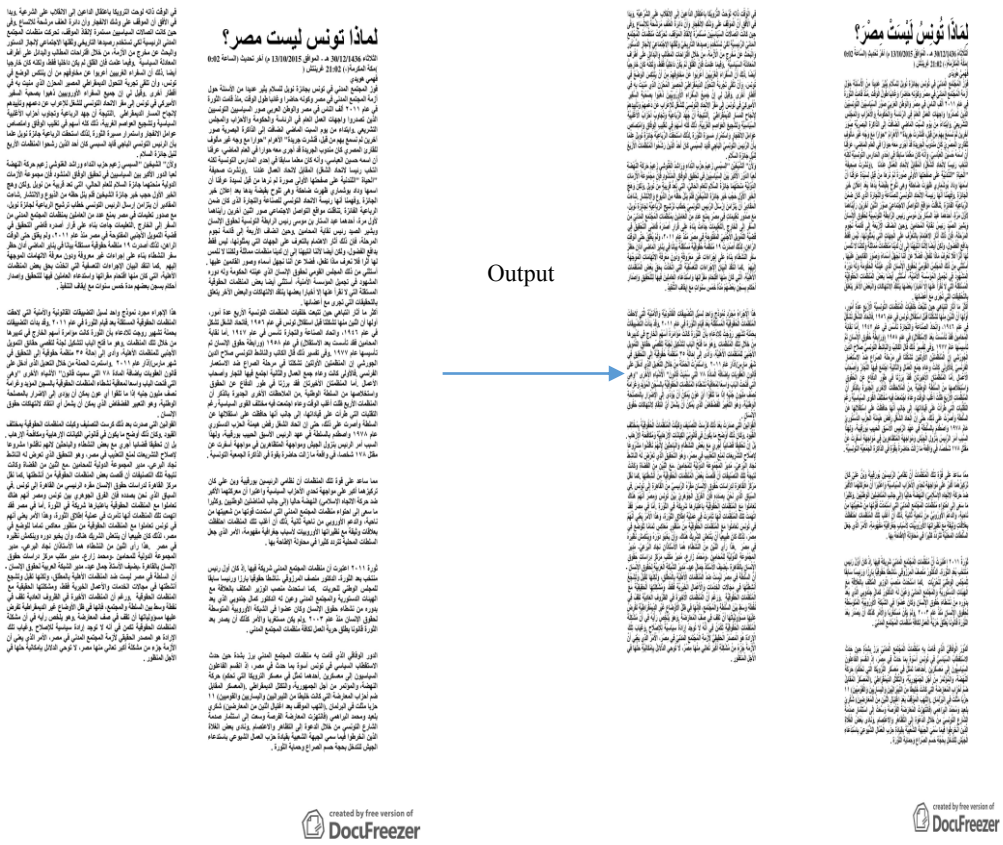


Figure 3-5 Separation regions where the vertical projection is equal to zero

Image of two columns page layout

Column 1

Column 2



Output

Figure 3-6 Page layout detection for page consists of two columns

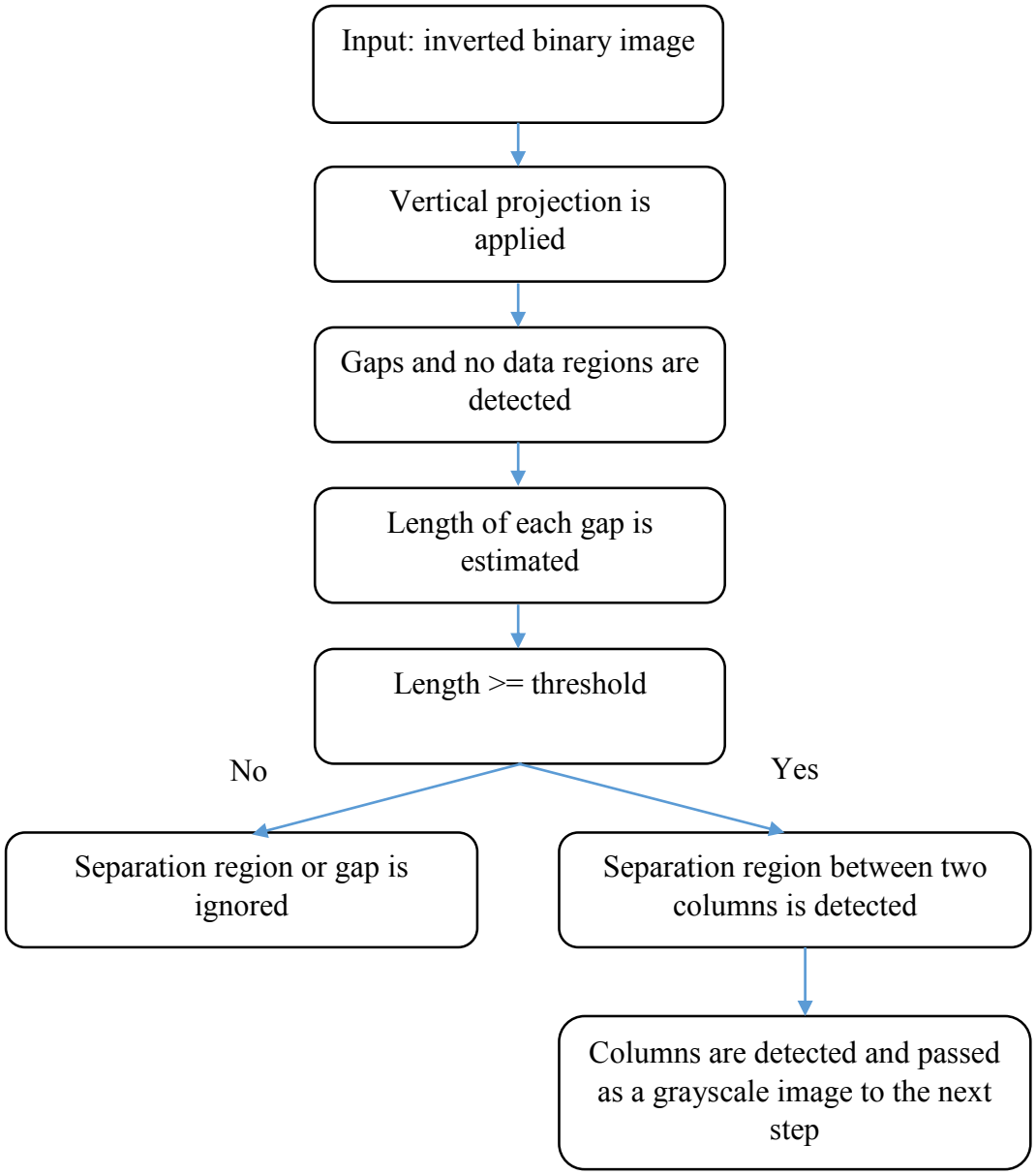


Figure 3-7 Page layout detection flow chart

3.1.3 *Generating Segments Consist of Lines with Equal Width*

Each column resulted from the previous step is divided into segments (each segment consists of one or more lines that have equal width). Opening technique plus horizontal projection are applied in this step.

Horizontal projection is an efficient way for segmentation when the inter-lines gap is clear. In case of diacritics existence, the separation between lines is not clear due to the overlapping, so an opening technique is applied to reduce the overlapping and interlacement. Also by using this technique, the segments with different lines width will have a clear gap in most cases, then the horizontal projection is applied to get these segments [18]. Each segment has one line or multiple overlapped lines with the same width inside.

The number of lines in the segment is then detected, if it consists of more than one line, then the overlapping case is detected, and 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 to be merged or not depending on its width if it exceeds a certain threshold value. Figure 3-8 shows the flow chart for generating these segments.

Each column (grayscale image) resulted from the page layout detection step is processed as the following:

- The interlacement caused by diacritics is reduced by using bwareaopen() MATLAB function using a certain threshold which is determined through the testing process.

- Horizontal projection is then applied, and the rows' indices where the projection equals to zero is determined, then the indices are grouped. Each group performs a separation region between two consequent segments.
- Each segment is determined by the separation region boundaries.
- The resulted segments consist of one or more lines, if more than one line inside the segment then the overlapping problem is detected and it is solved in the next stage, the lines inside this segment have the same width.
- In the last step, the segment is passed for extra processing to determine the overlapping case to be solved.

Algorithm 2 shows these steps, and figure 3-9 shows an example of the output segments generated from this stage.

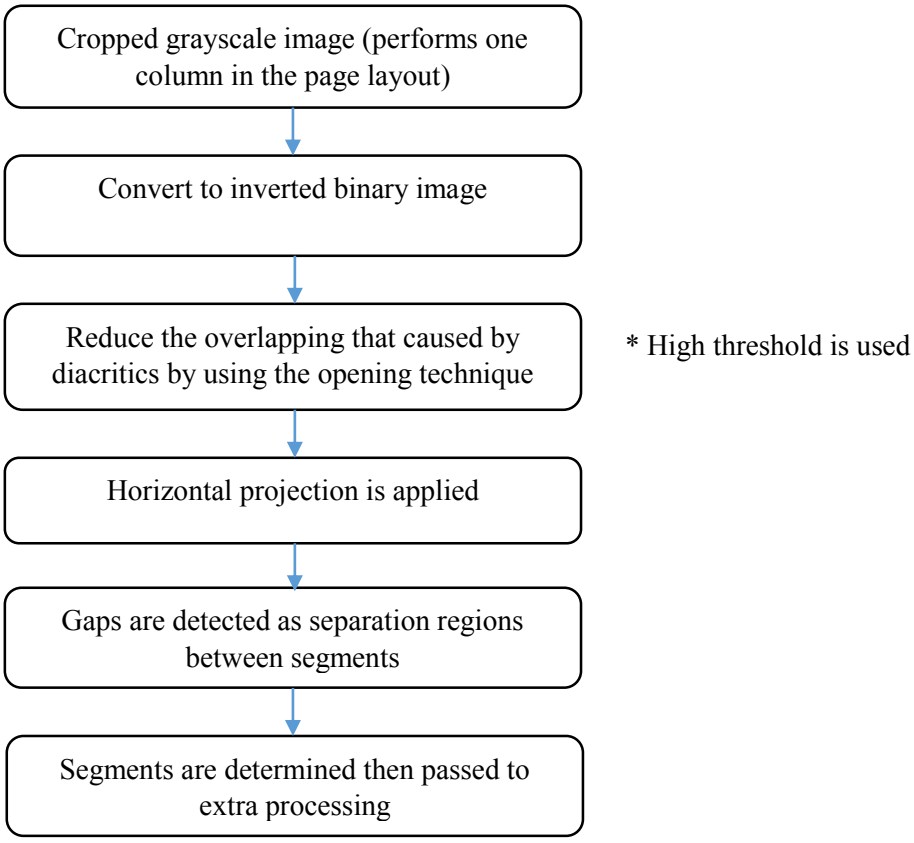


Figure 3-8 Generating segments with equal lines' width inside

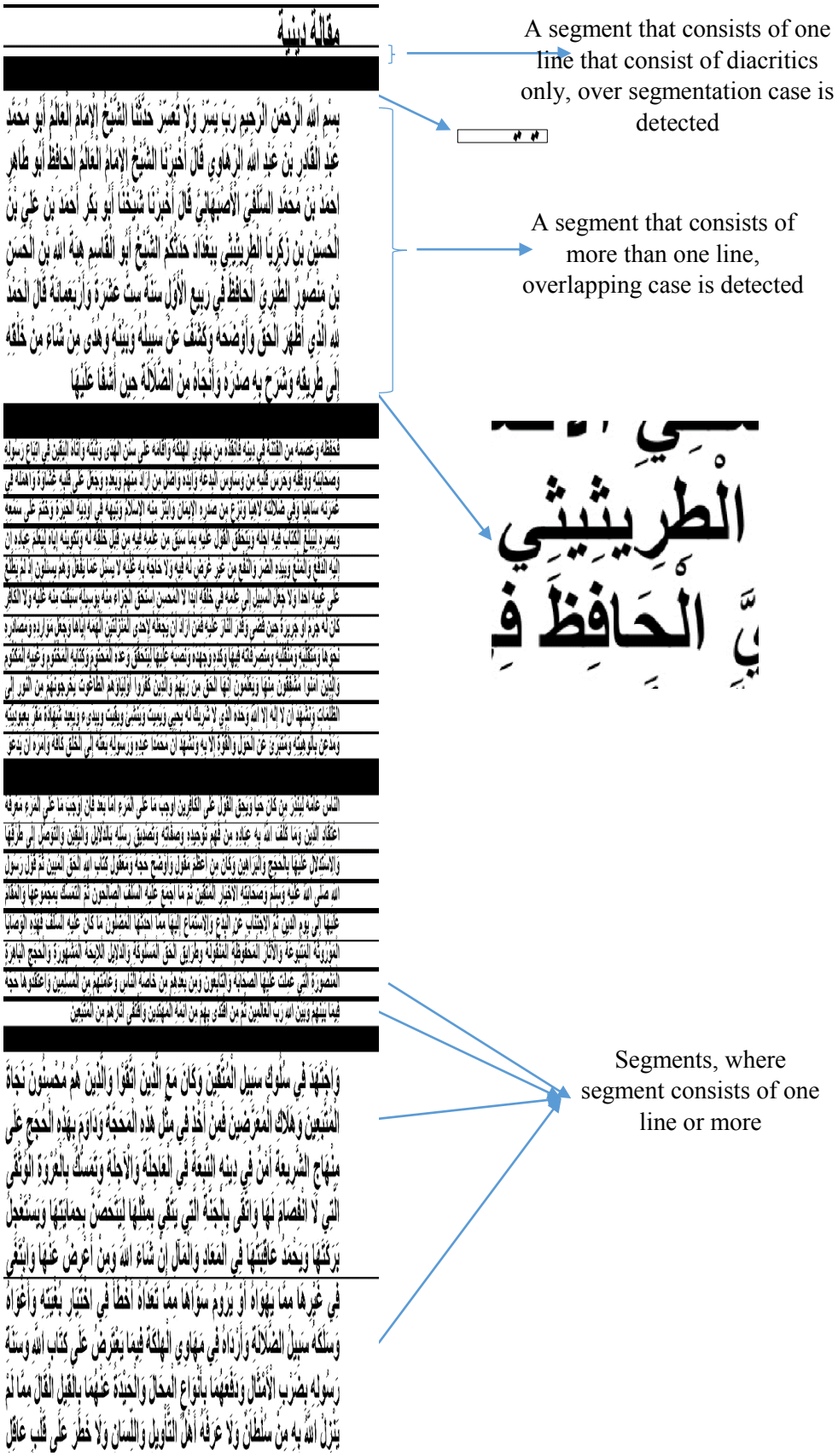


Figure 3-9 Generating segments consist of lines that have equal width

Algorithm 2: Column Segmentation

```
Input: Column Image (CImage)
Output: List of column Segments
begin
    //horizontal projection is applied
    H_proj ← horizontal_projection(CImage))

    //indices where horizontal projection is equal to zero are detected .
    Separation_indices ← H_proj(i، j) == 0

    //every set of continuos separation indices are grouped to form the separation region.
     separation_regions ← grouped_continuous_indices(Separation_indices)
     //extract column segments and add them to the column segment list .
    for i ← 1 to length(separation_regions) do
        //segment row start index
        segment_row_start_index(i) ← min(separation_regions(i))

        //segment row end index
        segment_row_end_index(i) ← max(separation_regions(i + 1))

        //crop the segment using start and end indices
        segment(i) ← Crop(CImage, segment_row_start_index(i), segment_row_end_index(i))

        //add the cropped segment to the segments' list
        Column_Segments_list.Add(segment(i));
    end
    return Column_Segments_list
end
```

3.1.4 *Initial Lines Count Estimation Inside the Segment*

The resulted segments from the previous step is passed to another stage to detect the overlapping by finding the approximated lines count inside the segment, if the count is more than one then the overlapping is detected, otherwise, the over segmentation checking is taken a place.

To get the approximated lines count inside the segment, a single line width and the segment width are calculated, then the count is easily calculated by dividing the width of the segment by the width of a single line, but how to get a single line width? The horizontal projection is applied to the segment, then the global maximum peak and its location are determined. The location of the global

maximum peak performs a baseline for a single line in the segment [25], then the baseline is marked with white color (assigning the value of pixels passed through with value equals one in white black image). Figure 3-10 shows the line drawn at global maximum peak. This drawn line makes the main body of the sub-word in the line as one connected component (it passes though the main bodies without diacritics), and can be selected by applying the selection operation. The obtained line from the selection operation is without diacritics, and its width is less than the actual width because it hasn't diacritics. Figure 3-11 shows the selection operation for the line located at global maximum peak (baseline).

$$\textbf{Approximated lines' count} = \frac{\textbf{Segment width}}{\textbf{Line main\_body width}} \quad (3\text{-}1)$$

To give better result for the line width, the segment's width is divided by the approximated count of lines, if the resulted count of lines inside the segment equals to one then the line (the input segment in this case) is added to the lines. Algorithm 3 defines the steps' sequence to split segment into lines.

$$\textbf{Approximated line width} = \frac{\textbf{Segment width}}{\textbf{Approximated lines count}} \quad (3\text{-}2)$$

Figure 3-10 The drawn line at the global maximum peak (baseline)

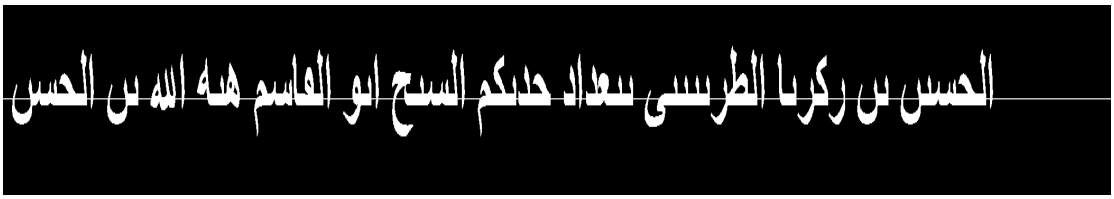


Figure 3-11 The selection operation for the line passes through the global max peak

Algorithm 3: Splitting Segment into Lines

Input: A grayscale image -segment- that consist of one or more lines that have equal width (Igray)
Output: List of lines
begin

```
//complementary of the gray-scale image
Igray ← Complement(Igray)

//compute the horizontal projection
h_proj ← sum(p(j^c i))

//local maximum peaks in the horizontal projection values
local_peaks ← find_peaks(h_proj)

// the rows indices for the peaks, every baseline locates at one of these indices
 peaks_locations ← location(local_peaks)

//calculate global maximum peak
max_peak ← max(local_peaks)

//the location of the global max peak
max_peak_location ← location(max_peak)

//draw line at the global max peak location, the line will perform one object
Ibinary ← draw_line_at_max_peak_location(segment)

//select the line that its baseline locates at global maximum peak
 line_at_max_peak ← select_line(max_peak)

//calculates an approximation for lines count inside the segment
approximated_lines_count ← segment_width / selected_line_width
if approximated_lines_count > 1 then
    // overlapping case
    ListOf(Lines) ← split_segment_to_individual_lines(Igray)
else
    //over segmentation checking
    ListOf(Lines) ← over_segmentation_checking(Igray)
end
return ListOf(Lines)
```

end

3.1.5 *Split Each Segment to Individual Lines in Case of Overlapping*

If the approximated lines count inside the segment is more than one, then the overlapping between lines is detected to get all lines inside the segment -which has lines with equal width inside-. Another stage of processing is wanted to avoid the error in calculating lines count that caused by diacritics in the document, so the lines count is recalculated to get more accurate result for the lines count.

In this stage, the segmentation of the overlapped lines is started by taking 35% of local maximum peaks resulted from the previous step and sort them in descending order, the locations of these peaks perform the baselines of many lines in the segment, these locations are sorted in descending order, and the distance between every two consequent locations is then calculated, the distance should be greater than or equal to the approximated line width obtained in the previous step to consider it as a valid distance. Figure 3-12 shows the locations of the valid peaks in the segment that perform the baselines for many lines. After the valid distances are calculated, the average value of them is calculated, which performs a single line width inside the segment. To get better estimation for single line width, the count of lines is then recalculated - calculated by dividing the width of the segment by the estimated line width -, then the width of the line is re-evaluated by dividing the segment's width by the count of lines. In the last step after finding the line width, the lines are determined by splitting the segment into lines' images. Figure 3-13 shows the segmentation result of overlapped lines inside the segment, and algorithm 4 describes the overlapped lines segmentation.

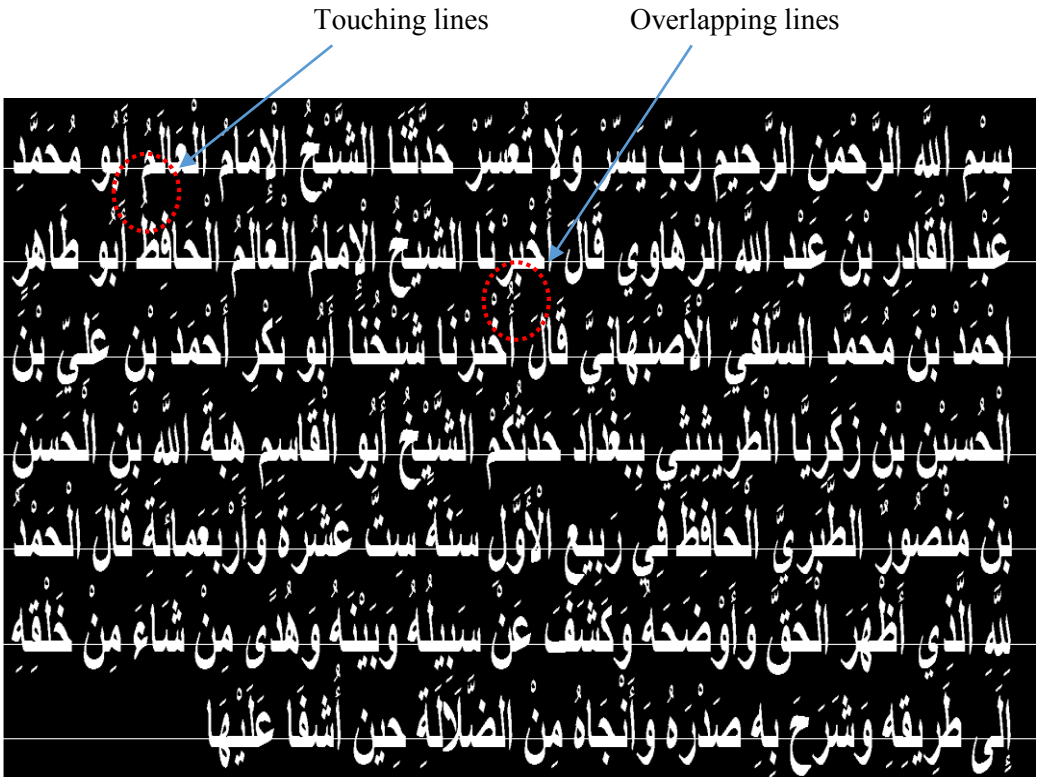


Figure 3-12 Valid peaks detected in the segment

Figure 3-13 Overlapped lines segmentation result

Algorithm 4: Segment Splitting to Individual Lines in Case of Overlapping

Input:

- Segment with overlapped lines inside (Segment)
- Approximated or expected line width

Output: List of equal width lines that locates inside the segment

```
begin
    // find local maximum peaks for the horizontal projection of the segment
    local_maximum_peaks ← find_peaks(H_proj(Segment))

    //sort peaks in descending order
    sorted_peaks ← sort_descending(local_maximum_peaks)

    //part of the sorted peaks are taken
    selected_peaks ← take_part(sorted_peaks)

    //the rows indices of the selected peaks are calculated
    peaks_locations ← location(selected_peaks)

    //sort locations in descending order
    sorted_locations ← sort_desc(peaks_locations)

    for i ← 1 to length(local_peaks) do
        //the distance between each two consequent locations is calculated
        distance ← distance(local_peaks(i), local_peaks(i + 1))
        if distance < expected_line_width then
            distance.ignore()
        else
            ListOf(distances).Add(distance)
        end
    end
    //the line width is the average distance
    initial_line_width ← mean(ListOf(distances))
    //calculate lines count
    lines_count ← segment_width / initial_line_width
    //recalculate line width
    line_width ← segment_width / lines_count
    ListOf(Lines) ← divide_segment_into_lines(line_width)
    return ListOf(Lines)
end
```

3.1.6 *Over Segmentation Problem Detection and Solving*

If a single line is detected after calculating the approximated lines count inside the segment, the line is passed to check for over segmentation case, if the line's width is less than a threshold value [28], then it will be merged with the previous line or next segment (still not segmented, which will pass for detecting lines count stage). In this step, the distance between the current line and the previous one is measured

then compared with the distance between the current line and the next segment, if it is smaller, then it will be merged with previous one, otherwise, it will be merged with the next segment.

To merge the current line with previous line, the minimum row index value for the previous line will be the minimum index of the current line; the image field will also be modified by applying the new boundaries for the cropping operation, which applied on the original grayscale image.

$$\boldsymbol{i}: \boldsymbol{index}$$

$$\boldsymbol{lines}(\boldsymbol{i}-\mathbf{1}).\boldsymbol{max\_index} = \boldsymbol{lines}(\boldsymbol{i}).\boldsymbol{max\_index}; \quad (3\text{-}3)$$

$$\boldsymbol{lines}(\boldsymbol{i}-\mathbf{1}).\boldsymbol{img} = \boldsymbol{im\_gray}(\boldsymbol{lines}(\boldsymbol{i}-\mathbf{1}).\boldsymbol{min\_row\_index} : \boldsymbol{lines}(\boldsymbol{i}-\mathbf{1}).\boldsymbol{max\_row\_index}, :);$$

To merge with next segment, the splitting region boundaries for the next segment are modified by replacing the current splitting region by the previous one.

$$\boldsymbol{current\_splitting\_region} = \boldsymbol{previous\_splitting\_region}. \quad (3\text{-}4)$$

3.2 *Line to Words, Sub-words and Diacritics Segmentation*

For Arabic OCR system, the next stage after line segmentation is mainly words/sub-words segmentation. The proposed methodology is mainly based on the projection profile approach and consists of set of sequence stages, as shown in figure 3-14, a

binary text line image of printed Arabic text with/without diacritics performs the input, and a segmented words/sub-words as an output.

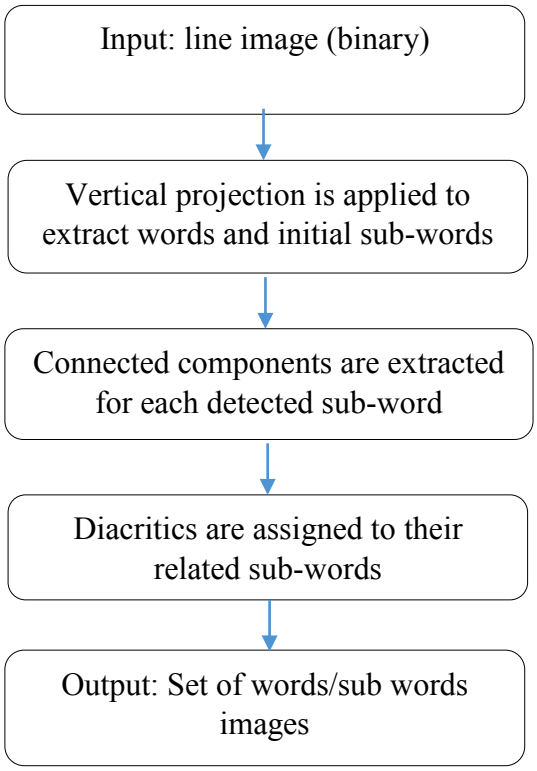


Figure 3-14 Word/sub-word segmentation algorithm

3.2.1 *Text Line Image Processing*

In this stage, to facilitate 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 dotting, while the second one contains only the diacritics and dotting.

To generate these versions, a horizontal projection method is applied to the original text line input image to find the global maximum peak and its location, which represent the baseline of the input text line. Then, a horizontal line is drawn at the

location of the baseline as shown in figure 3-15-2. To generate the first version, the connected component algorithm is applied to the image obtained from the previous stage and select the largest component which represents the body of the text without diacritics and dotting as in figure 3-15-3 To remove the horizontal line, we made a logical AND operation between the resulted image and the original one as in figure 3-15-4 The second version of the line image is obtained by subtracting the original input text line image (figure 3-15-1) from the generated first version (figure 3-15-4) as shown in figure 3-15-5, and algorithm 5 shows these steps in details.

Figure 3-15-1 Original line image (main bodies + diacritics + dotting)

Figure 3-15-2 The drawn line at the location of the baseline

Figure 3-15-3 Main body line version with horizontal line

Figure 3-15-4 Main body line version

Figure 3-15-5 Diacritics and dotting line version

Figure 3-15 Example of the three lines generation

Algorithm 5: Words/Sub-words Segmentation Pre-processing Step

Input: A binary image which is a line (*im_line*)
Result: im_line (full line) , im_main_body (line image without diacritics), im_diacritics (line's diacritics)
begin
//compute the horizontal projection for the input line image
$h\_proj \leftarrow sum(p(i, j))$

//calculate global maximum peak
$max\_peak \leftarrow max(local\_peaks)$

//get the location of the global maximum peak.
$max\_peak\_location \leftarrow get\_location(max\_peak)$

//draw line at the global maximum peak to make the main bodies as one connected component.
$Draw\_line(max\_peak)$

//extract the main-body image from the input line
$im\_main\_body \leftarrow select\_line(max\_peak)$

//extract the diacritics image from the input line
$im\_diacritics \leftarrow$ im_line.$Subtract(im\_main\_body)$
end

3.2.2 *Words and Sub-Words Extraction*

For Arabic script, each word consists of one or more sub-words [30]. To extract words/sub-words, a vertical projection is applied to the line's versions extracted from text line image processing as an initial step to find the gap between them (where the projection equals to zero) [18], so three versions of the words are available now (original word, word's main body and word's diacritics). The challenge after this step is to know that the text between two gaps is a word or a sub-word, and if it is a sub-word to which word belong. In other words, what is the suitable threshold to determine the separation space between the sub-words as an intra-space in the same word or a separation space between two distinct words? Moreover, the gap between two consecutive words or sub-words 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 [11] of the current two

consecutive words'/sub-words' main bodies, and compare it with length of the separation space between the current two 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, 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 the most frequent value calculated from the vertical projection. That means, if the sub-word consists of more than one character, then the pen size is the thickness of the baseline, and vice versa. Pen size calculation if formally defined as:

$$
\begin{aligned}
&\boldsymbol{SW}: \boldsymbol{Sub-Word} \\
&\boldsymbol{HP}: \ \boldsymbol{Horizontal\ Projection} \\
&\boldsymbol{VP}: \ \boldsymbol{Vertical\ Projection} \\
&\boldsymbol{MFV}: \ \boldsymbol{Most\ Frequent\ Value} \\
&\boldsymbol{PS}: \ \boldsymbol{Pen\ Size}
\end{aligned}
$$

$$
\begin{aligned}
&\boldsymbol{if}\ \boldsymbol{max}\big(\boldsymbol{HP}(\boldsymbol{SW})\big) > \ \boldsymbol{max}(\boldsymbol{VP}(\boldsymbol{SW})) \\
&\quad \boldsymbol{PS} = \boldsymbol{MFV}\ (\boldsymbol{VP})\,; \\
&\boldsymbol{else} \\
&\quad \boldsymbol{PS} = \boldsymbol{MFV}\ (\boldsymbol{HP})\,; \\
&\boldsymbol{end}
\end{aligned}
\tag{3-5}
$$

Figure 3-16 shows an example of pen size calculations for the two cases. In the example, there are two sub-words. For the first one (left), the pen size is chosen as the most frequent value from the horizontal projection, while in the second sub-word, the pen size is chosen as the most frequent value from the vertical projection.

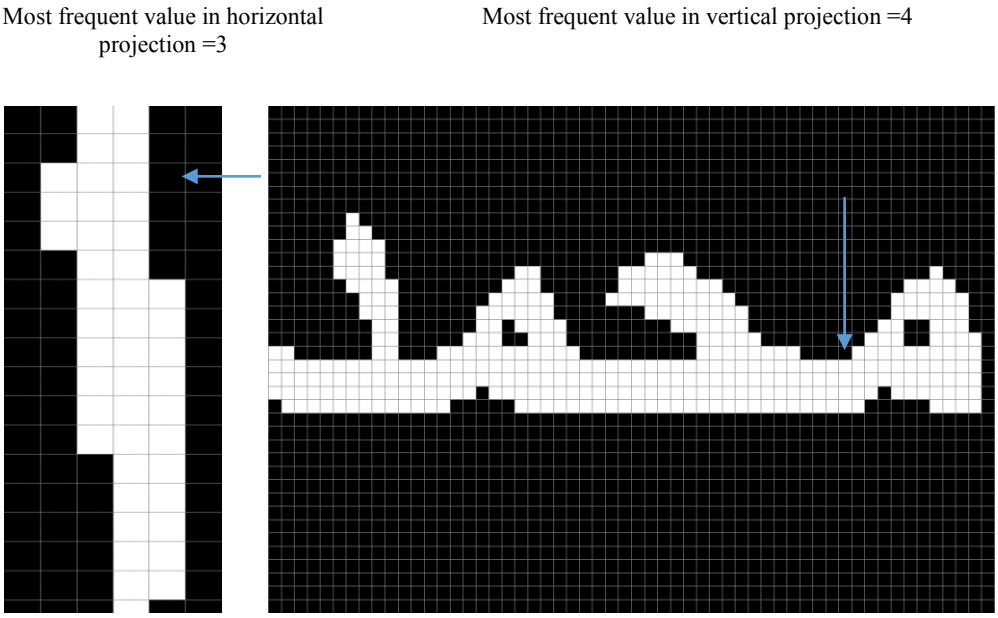


Figure 3-16 Pen size calculation

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

$$\boldsymbol{SS}: \boldsymbol{Separation\ space}$$
$$\boldsymbol{PS}: \boldsymbol{Pen\_Size}$$
$$\boldsymbol{CP}: \boldsymbol{Current\_Part}$$
$$\boldsymbol{NP}: \boldsymbol{Next\_Part}$$

$$\boldsymbol{If}\ \ (\boldsymbol{Length}\ (\boldsymbol{SS}) > \boldsymbol{mean}\ (\boldsymbol{PS}(\boldsymbol{CP}), \boldsymbol{PS}(\boldsymbol{NP}))$$
$$\boldsymbol{Word\_index}++$$
$$\boldsymbol{End} \tag{3-6}$$

Figure 3-17 shows how to determine if the two separated parts are related to the same word or different words. The figure shows that there are three separated parts (“ا”, “**حمد**”,”**بن** ”), and the pen size of these parts are 3, 4 and 3 respectively. The

separation space between the first 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. Algorithm 6 shows the steps in details.

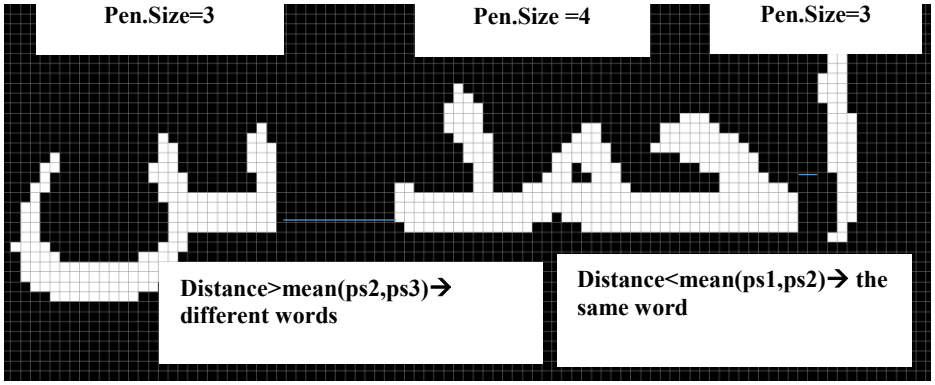


Figure 3-17 Distance between two sub-words in the same word and different words

3.2.3 *Sub-words Overlapping Case and Diacritics Extraction:*

After the words and sub-words extraction step, every word/sub-word is passed to another level of processing to check for the overlapping case between sub-words since the vertical projection method can't handle this case (no clear gap between them, they are overlapped) and also to link each diacritic to its related sub-word. In addition, the detection of a single dot and two touching dots is done in this step, they are needed in the character segmentation post processing step to determine some special cases like "SEEN" and "SHEEN" characters.

Figure 3-18-1 shows two overlapped sub-words, the overlapped sub-words in this case are extracted by applying the connected components algorithm on the main

body image version of the sub-word [32] since it is difficult to find the overlapped sub-words using vertical projection technique which used if a clear gap is found, by using the connected component every group of touching pixel is considered as one component. Figure 3-18-2 shows the main bodies for the two overlapped sub-words. The number of the sub-words equals the number of the connected components in the main body image version.

Figure 3-18-1 Overlapped sub-words

Figure 3-18-2 Two overlapped main bodies of sub-words

In this step, another issue appears related to the extracted connected components ordering in the overlapped sub-words. If there are many overlapped sub-words and the connected components are extracted, they should be sorted from right to left, to solve this issue they are re-ordered by max column index (depending on Arabic script catachrestic) for each connected component to give the correct result.

To assign each diacritic to its related sub-word, firstly, all diacritics of the diacritic's sub-word image that related to the processed main body image are extracted using

the connected component algorithm [32]. Figure 3-18-3 shows the diacritics that are related to the overlapped sub-words.

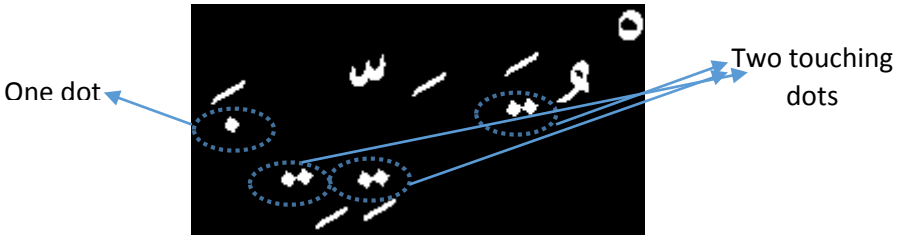


Figure 3-18-3 Diacritics image of overlapped sub-words

Then, the overlapping percentage is calculated for each diacritic with the overlapped sub-words extracted from the previous step. The diacritic that are assigned to the sub-word have the max overlapping percentage with it. If the sub-word consists of one connected component then no overlapping is detected, so all diacritics in the diacritics image version are related to it. The overlapping percentage is calculated as following:

- Find the overlapping area by finding the intersection area between the diacritic and sub-word at the column level.
- Find the size of the overlapping area
- Find the size of the diacritic
- Find the overlapping percentage between the size of the overlapping area and the diacritic size

$$\boldsymbol{Overlapping\_percentage} = \frac{\boldsymbol{overlapping\_area\_size}}{\boldsymbol{diacritic\_size}} \quad (3\text{-}7)$$

Figure 3-18-4 shows the final segmentation result in case of overlapped sub-words and diacritics and assigning each diacritic to its related sub-word. It is worthy to mention that while assigning the diacritics to its related sub-word, the generated images' sizes should be equal to the original image (sub-word) size in case of overlapped sub-words, so the indices that have been gotten from the extraction of connected components can be used again since the images perform a reference to the original one.

Figure 3-18-4 Sub-words and diacritics segmentation result in case of overlapping

Algorithm 6: Words/ Sub-words and Diacritics Segmentation

Input: A line image in binary format (im_line)
Output: List of sub-words
begin
  //vertical projection is applied
  $V\_proj \leftarrow im\_line \triangleright sum(p(i, j))$

  //indices where vertical projection equals to zero
  $Separation\_indices \leftarrow V\_proj == 0$

  //separation indices are grouped to be considered as separation regions.
  $separation\_regions \leftarrow group\_continuous\_indices(Separation\_indices)$

  for $i \leftarrow 2$ ***to*** $length(separation\_regions)$ **do**
    $sub-word\_min\_column\_index \leftarrow max(separation\_regions(i))$

    $sub-word\_max\_column\_index(i) \leftarrow min(separation\_regions(i+1))$

    //the separation area length between two sub-words
    $Separation\_length(size) = max(current\_group\_indices)$- $\min(current\_group\_indices)$

    //determine if the separation region is between two sub-words in the same word or different words .
    if $Separation\_length(size) > mean(sub-words(i)\_pen\_size, sub-words(i-1)\_pen\_size)$ **then**
      $Word\_index++$
    end
    $sub-words(i).word\_index \leftarrow$ Word_index

    //sub-word image
    $sub-words(i).im\_sub-word \leftarrow im\_line(:, sub-word\_min\_column\_index(i):$
    $sub-word\_max\_column\_index(i))$

    //sub-word diacritics image
    $sub-words(i).im\_diacritics \leftarrow$
    $im\_diacritics(:, sub-word\_min\_column\_index(i):sub-word\_max\_column\_index(i))$

    //sub-word main body image
    $sub-words(i).im\_main\_body \leftarrow$
    $im\_main\_body(:, sub-word\_min\_column\_index(i):sub-word\_max\_column\_index(i))$

    //check for overlapping, one dot, two dots and to link each diacritic with its correct sub-word
    $Get\_Overlapped\_Sub\_words\_AND\_Diacritics(sub\_words(i))$;
    $ListOf(sub-words).Add(sub-word(i))$;
  end
  return $ListOf(sub-words)$
end

How to determine the dot and two touching dots in the diacritics image version?

The connected components in the diacritics image that related to the sub-word are passed for checking, each connected component performs a single diacritic, then the region properties (solidity, major axis length, minor axis length and centroid) for each diacritic are calculated, these properties describe the dot and can distinguish it from other diacritics. Figure 3-18-3 shows diacritics' image which contains one dot and two touching dots.

To consider the diacritic as one or single dot the following conditions should be satisfied:

- The solidity for single dot is more than .8, the solidity gives an indication to how much the shape is filled, it is the ratio between the shape area and its convex hull area. By conducting the experiments for different font sizes and types the results shows that one dot has a solidity greater than .8.

$$\boldsymbol{diacritic.region\_props(solidity) > .8} \quad (3\text{-}8)$$

- The nearest two pixels from the centroid should have a value equals to 1; that means the shape is filled in region near to center.
- The approximation for the ratio between the major axis length and the minor axis length should be equal to one.

$$\boldsymbol{Round\left(\frac{major\_axis\_length}{minor\_axis\_length}\right) = 1} \quad (3\text{-}9)$$

Figure 3-19 shows the major Axis length, the minor axis length and the ratio between them that equals to one.

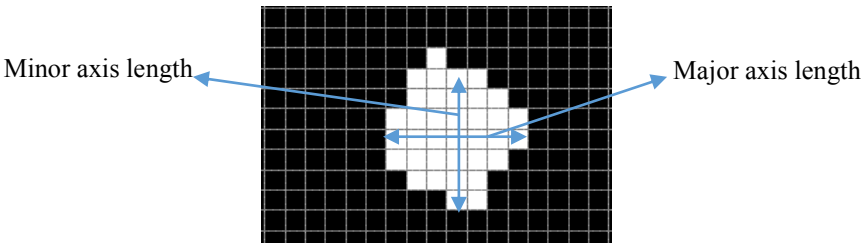


Figure 3-19 Ratio between major axis length and minor axis length is approximately equal 1

If the previous conditions are not satisfied, then the algorithm starts checking for the two touching dots. The diacritic is split into two parts from the center point then the checking for one dot is repeated for each part: the solidity should be greater than .88, this value obtained from conducting experiments for different font sizes and types, and the approximated ratio between major axis length and minor axis length should be equal to one. Figure 3-20 shows two touching dots.

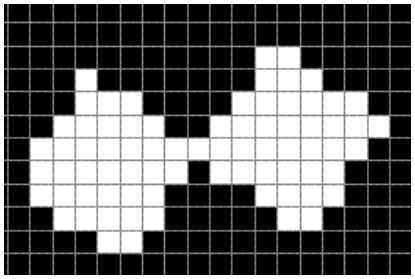


Figure 3-20 Two touching dots

Figure 3-21 shows an example of word/sub-word and diacritics segmentation.

Figure 3-21-1 Input image (line)

Figure 3-21-2 Main body line version

Figure 3-21-3 Diacritics and dotting line version

Figure 3-21-4 Splitting result after applying vertical projection to the main body line

Figure 3-21-5 Diacritics and doting version after splitting at the same locations where the splitting regions locates in the main body line version

Figure 3-21-6 Main bodies' separation after applying the connected component concept

Figure 3-21-7 Diacritics and dots separation after applying the connected component concept and computing the overlapping percentage

Figure 3-21-8 Sub words' separation result after merging the previous two output images

Figure 3-21-9 Words separation result after computing the pen size between every two consequent sub-words

Figure 3-21 An example of word/sub-word and diacritics segmentation

3.3 *Character Segmentation*

The proposed algorithm for character segmentation based on the contour extraction technique and passes through the following steps as shown in figure 3-22:

- Word/sub-word binary image is used as input.
- Contour extraction stage: in this stage the outer shape or boundary of the sub-word's main body is extracted [3].
- Up-contour extraction: the upper part of the contour is extracted to determine the cutting regions; tracing algorithm is used to extract the upper part.
- Post processing step is needed because the initial splitting causes an over segmentation problem for some characters like SEEN (س), SHEEN (ش), SAD (ص) and DAD (ض).

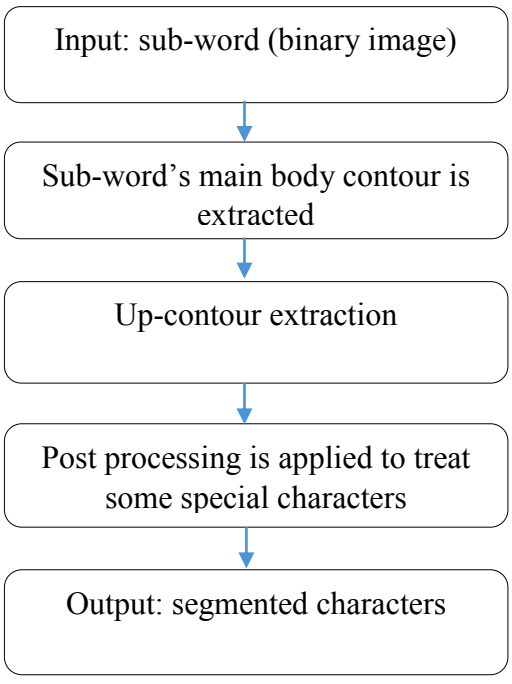


Figure 3-22 Character segmentation algorithm

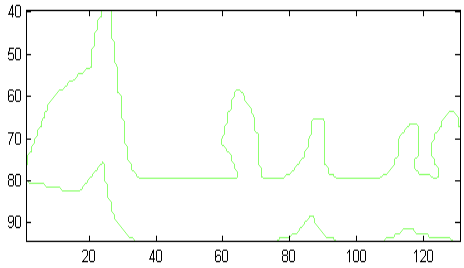
3.3.1 *Contour Extraction*

The proposed algorithm depends on extracting the contour to split the sub-word into sequence of characters; contour extraction technique gives a clear description for the sub-word or characters shape, so determining the splitting regions would be easier. Then, the outer contour is extracted using sequence of morphological operations.

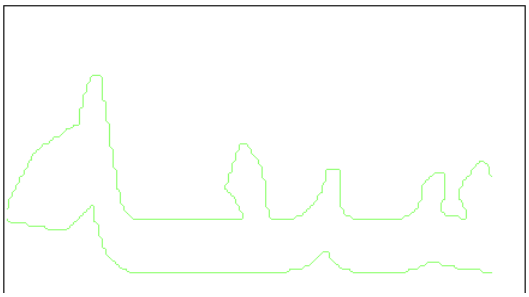
Many ways were tested to extract the contour of the sub-word, the best results obtained by using imcontour() MATLAB function with some enhancements added to it. The using of imcontour() function gives better results specially for small font sizes, while extraction the contour by sequence of morphological operations shows some problems like the connection between contour parts, and sometimes part of the word is erased which cause major problems in next processing steps (extraction of the up contour and the splitting areas or regions) [33]. The contour of the sub-word is extracted only while the diacritics are removed from the input sub-word since only the main body is needed. Enhancements are applied to the sub-word's main-body by filling the holes [29] and dilation by two pixels [31]. As a result, some smoothing is done without affecting the details of small font sizes, so the structure of some characters like seen (س) is not erased. The imcontour() function is applied to the main body by one level to get the outer contour, the axis of resulted figure are then adjusted by using 'auto' option and by removing the ticks, the figure is then converted to binary image and some gaps are removed. Figure 3-23 shows the steps of the contour extraction.

Original sub-word

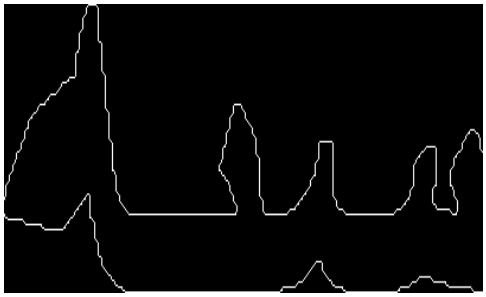
Main-body as input to the contour extraction function



Imcontour function output (outer contour up to one level)



Contour figure enhancements (auto axis and removing the ticks)



Converting the contour figure to binary image.

Figure 3-23 Contour extraction steps

3.3.2 *Up-Contour Extraction*

In this step, two points on the resulting contour are elected as a start and end points for the path that will form the up contour [18], the start point locates in the first part of the sub-word's contour, and the end point locates in the last part of the sub-word's contour.

To determine the first and the last parts of the sub-word's main body, firstly, a threshold value is determined according to the average length for of the character for the current used pen size, so the first part locates between the maximum column index of the sub-word contour and the second column, which is determined by subtracting the previous threshold value from the max column index [33].

$$\boldsymbol{CCI}: \boldsymbol{Contour\ Columns\ Indeces}$$
$$\boldsymbol{TV}: \boldsymbol{Threshold\ Value}$$
$$\boldsymbol{CFP}: \boldsymbol{Contour\ First\ Part}$$

$$\boldsymbol{Column\_1} = \boldsymbol{max}(\boldsymbol{CCI}) \quad (3\text{-}10)$$
$$\boldsymbol{Column\_2} = \boldsymbol{Column\_1} - \boldsymbol{TV}$$
$$\boldsymbol{CFP} = \boldsymbol{contour}\ (\boldsymbol{Column\_2} : \boldsymbol{Column\_1})$$

In the same manner, the last part is determined, but at this time the region locates between the first column index of the contour and the second column which is assigned by adding the previous threshold to the first column index value. Figure 3-24 shows the first and the last parts of the sub-word contour.

$$\boldsymbol{TV}: \boldsymbol{Threshold\ Value}$$
$$\boldsymbol{CLP}: \boldsymbol{Contour\ Last\ Part}$$

$$\boldsymbol{Column\_2} = \boldsymbol{1} + \boldsymbol{TV} \quad (3\text{-}11)$$
$$\boldsymbol{CLP} = \boldsymbol{contour}\ (\boldsymbol{1} : \boldsymbol{Column\_2})$$

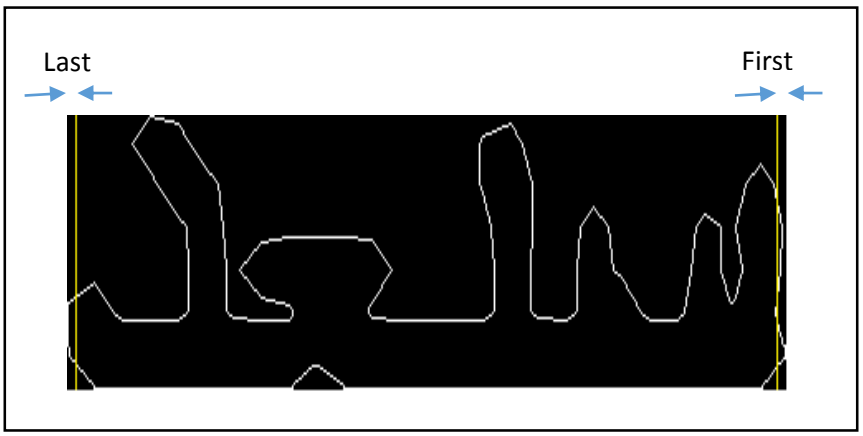


Figure 3-24 Last and first parts of the up-contour

To elect the start point; the two pixels which have the minimum row index values are located, then the pixel with maximum column index is elected, this way gives better accuracy in extracting the up-contour. Algorithm 7 describes the steps for electing the start point. The same for the end point election; the two pixels which

have the minimum row index values are located, then the pixel with minimum column index is elected. Algorithm 8 describes the steps for electing the end point. Now, the start and the end points are ready to start the tracing and extracting the up-contour. Figure 3-25 shows the elected start and end points.

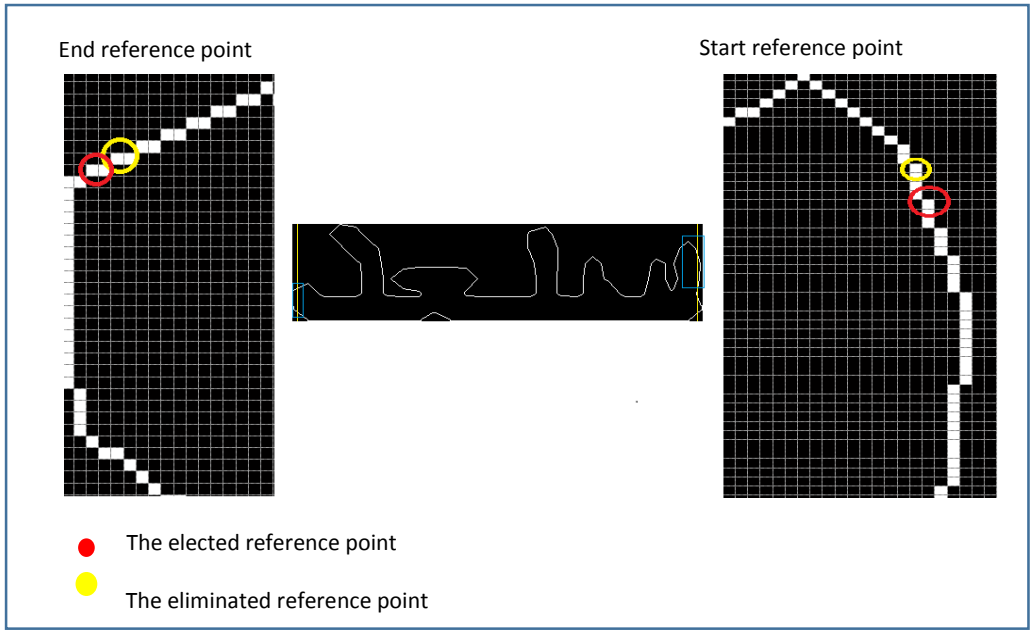


Figure 3-25 Start and end points election for tracing and extracting the up-contour

Algorithm 7: Start Point Election

Input: Contour First Part (CFP)
Output: Start_Point(X,Y)
begin
// find CFP rows indices
$rows\_indices \leftarrow find\_rows(CFP)$

//sort in Ascending order
$sorted\_indices \leftarrow sort(rows\_indices)$

//find min two values
$min\_two\_row\_indices \leftarrow$ sorted_indices(1,2)

//find max column index (y) from the points that have row index (x) equals to the first min row value
$col\_index\_option1 \leftarrow max\_c$(CFP(c,$min\_row\_1$))

//find max column index (y) from the points that have row index (x) equals to the second min row value
$col\_index\_option2 \leftarrow max\_c$(CFP(c,$min\_row\_2$))

if $col\_index\_option1 > col\_index\_option2$ **then**
$Start_Point_x =$min_two_row_indices(1)
$Start_Point_y =$col_index_option1
else
$Start_Point_x =$min_two_row_indices(2)
$Start_Point_y =$col_index_option2
end
$Start\_Point =$ (Start_Point_x,$Start\_Point\_y$)
return $Start\_Point$
end

Algorithm 8: End Point Election

```
Input: Contour Last Part (CLP)
Output: End_Point(X,Y)
begin
    // find CLP rows indices
    rows_indices ← find_rows(CLP)

    //sort in Ascending order
    sorted_indices ← sort(rows_indices)

    //find min two values
    min_two_row_indices ←sorted_indices(1,2)

    //find min column index (y) from the points that have row index (x) equals to the first min row
     value
    col_index_option1 ← min_c(CLP(c,min_row_1))

    //find min column index (y) from the points that have row index (x) equals to the second min row
     value
    col_index_option2 ← min_c(CLP(c,min_row_2))

    if col_index_option1 >col_index_option2 then
        End_Point_x =min_two_row_indices(2)
        End_Point_y =col_index_option2
    else
        End_Point_x =min_two_row_indices(1)
        End_Point_y =col_index_option1
    end
    End_Point = (End_Point_x,End_Point_y)
    return End_Point
end
```

The tracing operation is started from the elected start point pixel until reach the elected end point pixel.

A new empty image is created with the same size of the sub-word contour, each time a new pixel is located on the path, it is assigned to this empty image which will contain the up-contour at the end. The path is determined by moving from the start point pixel to the end point pixel in counterclockwise direction using 8-edge neighboring to check different possibilities for next pixel location. Figure 3-26 shows the tracing direction to extract the up-contour [16].

The problems appear during the tracing operation are summarized by stuck in loops, and visiting the already visited points. To overcome these problems, a checking

operation for the next pixel -that is located in the expected next movement- is done, if it is already assigned on the up-contour image then it is ignored, else, the movement is done and the pixel is assigned to the up-contour image.

Each time the movement action is taken a place, the previous point is tracked, so in case of stuck in loops, an automatic change in path direction from the previous point is done. The tracing operation is finished when the elected end point in the contour is reached, at this moment, the up-contour is ready to be used and passed to another stage to determine the splitting areas or regions located on.

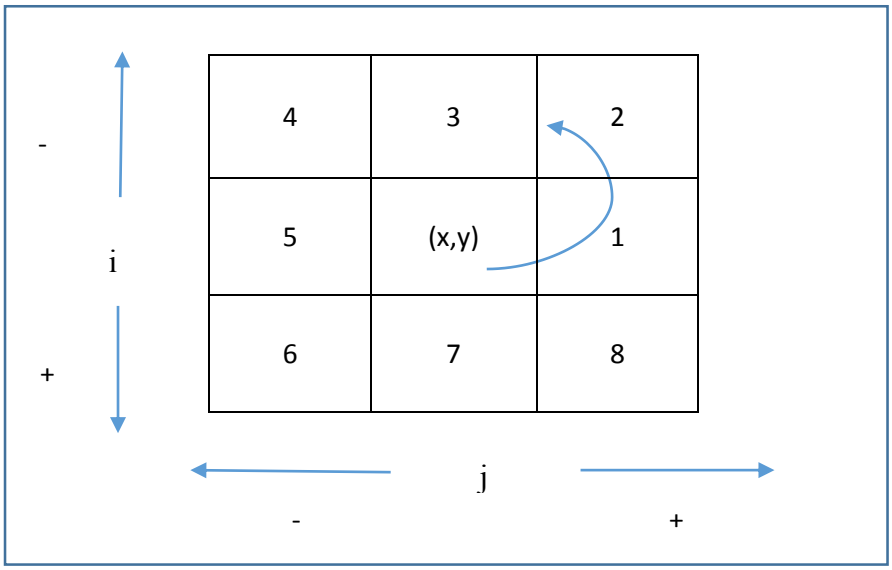


Figure 3-26 Tracing in counterclockwise direction to extract the up-contour

3.3.3 *Splitting Regions Extraction*

In this step, the up-contour is scanned to extract the splitting areas, where the sub-word can be segmented.

To extract the splitting areas, the up-contour is scanned row by row, then the continuous regions in every scanned row are determined. To consider this region as

a splitting area, it should perform a local minima region [25]. The first point and the last point in the region are examined before; if they satisfy the conditions as in the pseudo code shown below, then the region is considered as a splitting area so the first and last points in the region are considered as a splitting area reference points [33]. Figure 3-27 shows the splitting areas, and figure 3-28 shows their reference points.

$\boldsymbol{CSR}: \boldsymbol{Current\ Splitting\ Region}$
$\boldsymbol{r}: \boldsymbol{Row\ Index}$
$\boldsymbol{c}: \boldsymbol{Column\ Index}$
$\boldsymbol{UC}: \boldsymbol{Up\ Contour}$

$$\begin{aligned}&\boldsymbol{if\ UC}\left(\boldsymbol{CSR}(\boldsymbol{r})-\boldsymbol{1}, \boldsymbol{min}\left(\boldsymbol{CSR}(\boldsymbol{c})\right)\right) == \boldsymbol{1}\ ||\ \boldsymbol{UC}\ (\boldsymbol{CSR}(\boldsymbol{r})\ - \\ &\boldsymbol{1}, \boldsymbol{min}(\boldsymbol{SRC}(\boldsymbol{c}))\ ==\ \boldsymbol{1} \\ &\quad \boldsymbol{expected\_min\_col}\ ==\ \boldsymbol{min}(\boldsymbol{CSR}(\boldsymbol{c})) \\ &\boldsymbol{end}\end{aligned} \tag{3-12}$$

$$\begin{aligned}&\boldsymbol{if\ UC}(\boldsymbol{CSR}(\boldsymbol{r})-\boldsymbol{1}, \boldsymbol{max}(\boldsymbol{CSR}(\boldsymbol{c})))\ ==\ \boldsymbol{1}\ ||\ \boldsymbol{UC}\ (\boldsymbol{CSR}(\boldsymbol{r})\ - \\ &\boldsymbol{1}, \boldsymbol{max}(\boldsymbol{CSR}(\boldsymbol{c}))+\boldsymbol{1}\ )\ ==\ \boldsymbol{1}) \\ &\quad \boldsymbol{expected\_max\_col}\ =\ \boldsymbol{max}(\boldsymbol{CSR}(\boldsymbol{c})) \\ &\boldsymbol{end}\end{aligned}$$

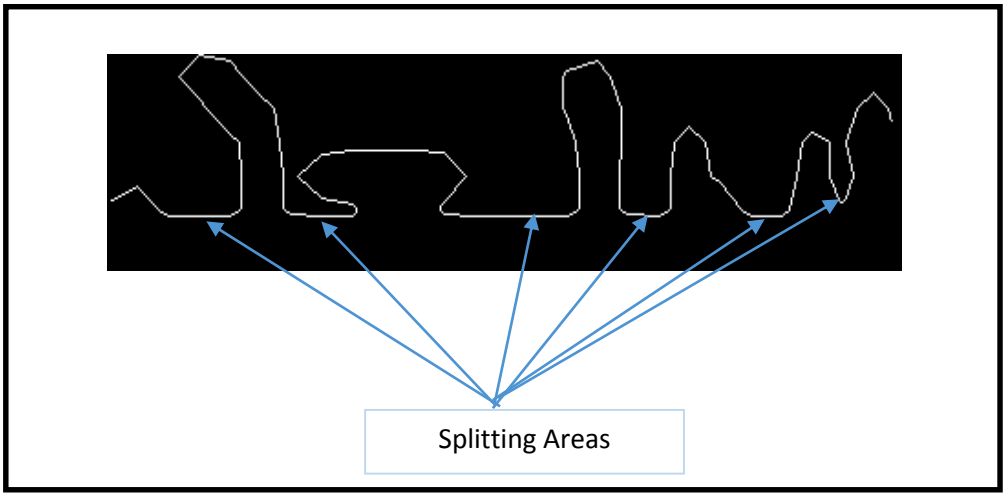


Figure 3-27 Up-contour splitting areas

Now each character locates between two splitting areas, but there are some special cases in which the splitting area locates within a character like SAD (ص), DAD(ض), SEEN(س), SHEEN (ش), TAA(ط) and THAA(ظ), also there are splitting areas may locate in characters when they are locating at the end of the sub-word or exist independently in the text like BAA(ب) and YAA(ي), so post processing step for character segmentation is necessary to ignore these splitting areas.

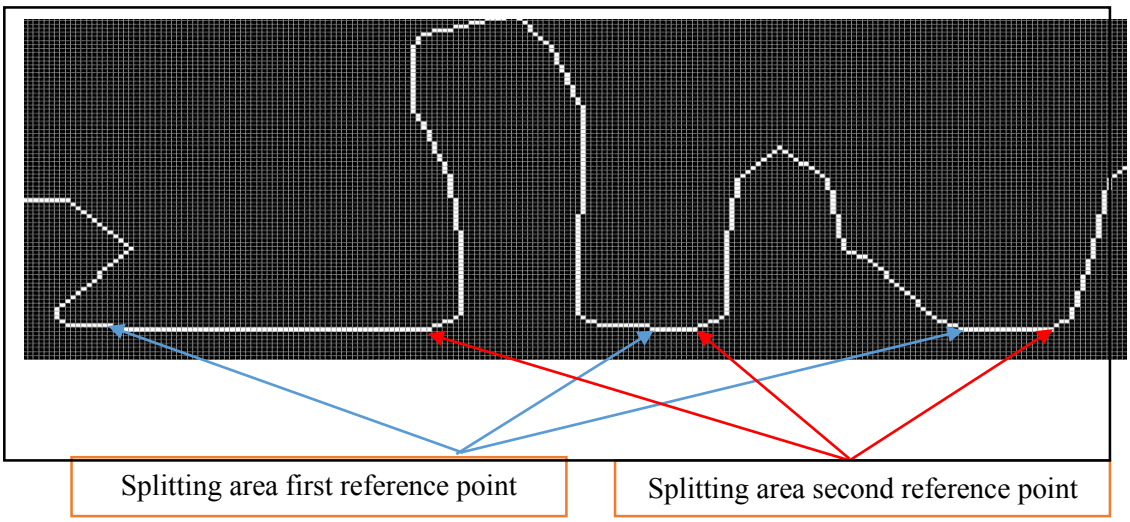


Figure 3-28 Splitting area reference points

3.3.4 *Character Segmentation Post Processing*

After determining the splitting areas, the areas pass through extra checking to distinguish which of them should be ignored and which not, for example, SEEN (سـ) character in the middle of the sub-word has three splitting areas, the first two should be ignored and the third one should be considered as a cutting region.

In the post processing step, the character segmentation algorithm passed through several steps, the first step is to determine the character's part, the character's part is defined as part of the up-contour which locates between two splitting areas as in

equation (3-13). The character's part performs a full character or a part within a character locates, so checking for some specifications is done to determine if this part should be merged or to consider it as a standalone character.

$$\boldsymbol{uc}: \boldsymbol{up\ contour}$$

$$\boldsymbol{csr}: \boldsymbol{current\ splitting\ region}$$

$$\boldsymbol{psr}: \boldsymbol{previous\ splitting\ region}$$

$$\boldsymbol{frp}: \boldsymbol{first\ reference\ point}$$

$$\boldsymbol{srp}: \boldsymbol{second\ reference\ point}$$

$$\boldsymbol{cp}: \boldsymbol{character\ part}$$

$$\boldsymbol{cp} = \boldsymbol{uc}(\boldsymbol{csr}.\boldsymbol{frp}(\mathbf{2}): \boldsymbol{psr}.\boldsymbol{srp}(\mathbf{2}), :) \tag{3-13}$$

Some specifications should be satisfied to merge this part; these specifications are as following:

- Euler number, which is one of the region properties that can be estimated; it should be equal or greater than one, which means that no holes exists in this part.
- Character's part height; the height should be less than the pen size multiplied by two.
- The part has no dots above or below. Figure 3-29 shows an example of a valid character's part that should be merged.

If the character's part satisfies the conditions or specifications, then the checking for SEEN (ش) case is taken a place, if the character's part is part of SHEEN (ش)

character (only the last part of SHEEN (ش) character satisfies the conditions), then the previous two splitting areas are ignored this means that the part is merged with two previous parts to perform SHEEN (ش) character. Figure 3-30 shows the detection for sheen (**ش**) case, if it is not SHEEN (ش) character, then the space counter is incremented by one.

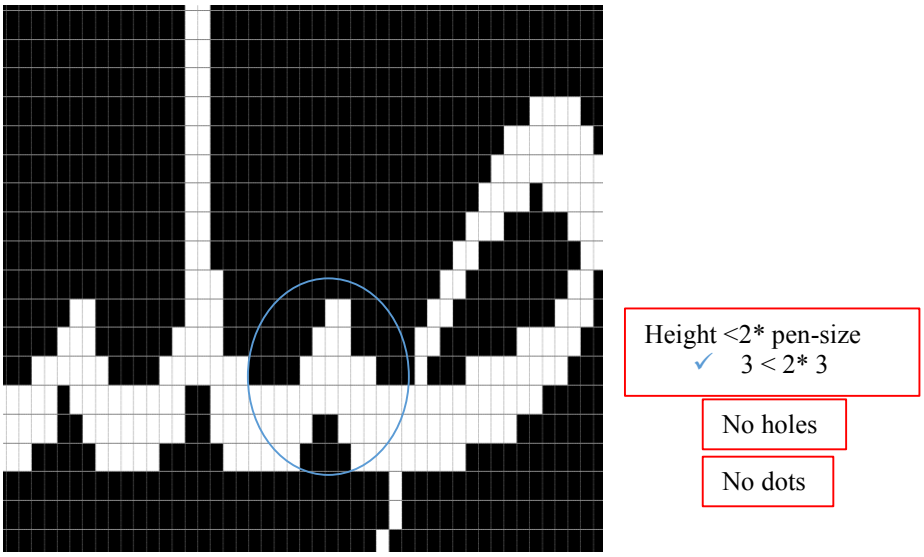


Figure 3-29 Valid character's part to be merged

Else if the character's part doesn't satisfy the conditions, then two cases should be checked:

- The first case, if the character's part is within the last character in the sub-word (the last character is treated independently).
- The second case, is the general case which checks the count of the spaces to discover or detect some special characters; when the space counter equals to three then SEEN (س) character is detected (in this case the character's part is merged with previous two parts to perform SEEN (س) character), and

when the space counter equals to one, then SAD (ص) or DAD (ض) character is detected (in this case the character's part is merged with the previous part to perform SAD (ص) or DAD (ض) character). Figure 3-31 shows seen case detection when the space counter equals to three (three characters' parts satisfy the conditions), and figure 3-32 shows SAD (ص) and DAD (ض) cases detection when the space counter equals to one (one character's part satisfies the conditions).

If the character's part locates within the last character, and last splitting region is below the baseline [13] with a distance equals to the pen size or more, then the splitting region should be ignored. Figure 3-34 shows the last splitting area which should be ignored.

After ignoring the last part, and if it has no dots, then that means that the part is not a full character and should be merged with some parts before to perform one character, like in SEEN (س), SHEEN(ش), SAD (ص) and DAD (ض) cases, so additional checking should be done to detect these cases; if the last part is a part of SHEEN (ش) character, then the last part should be merged with previous two parts, if the space counter equals to two, then SEEN (س) case or character is detected and the part should be merged with previous two parts, lastly, if the space counter equals to one, then SAD (ص) or DAD (ض) case is detected and the part should be merged with previous part. Figure 3-35 shows how the last splitting area is ignored due to its existence below the baseline, and how the last part is merged with the previous parts to perform one character according to the detected character. For TAA (ط) and THAA (ظ) characters, the splitting region locates above the base line with distance

equals at least the sub-word's pen size, if this condition is satisfied, then the splitting region is ignored. Figure 3-33 shows TAA (ط) case, and algorithm 9 shows these steps in details.

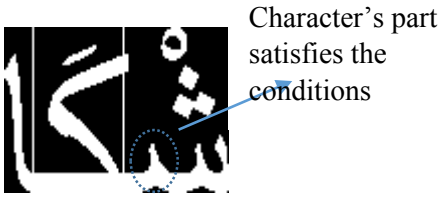


Figure 3-30 SHEEN case is detected, so the character's part is merged with previous two parts to perform one character

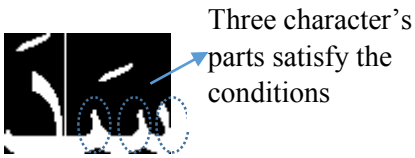


Figure 3-31 Space counter = 3, SEEN case is detected and the three parts are merged to perform one character

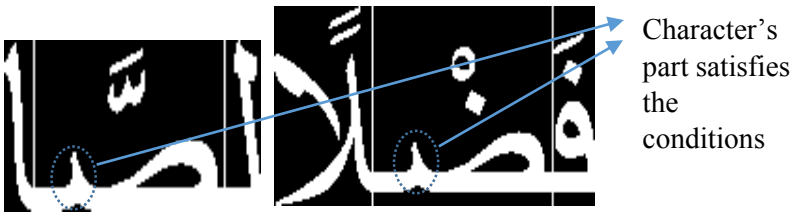


Figure 3-32 Space counter = 1, so SAD or DAD case is detected and the part is merged with previous one to perform one character

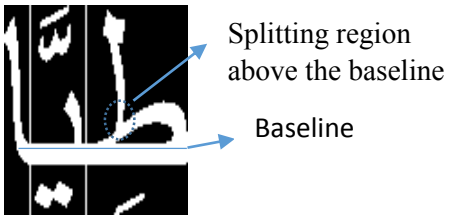


Figure 3-33 Splitting region is above the baseline so it should be ignored (TAA and THAA cases)

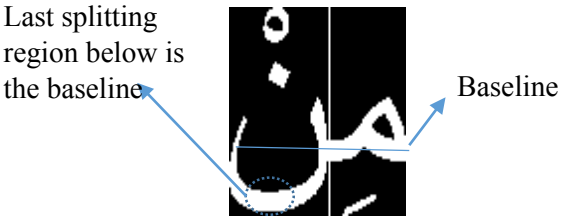


Figure 3-34 Last splitting region is below the baseline so it is ignored

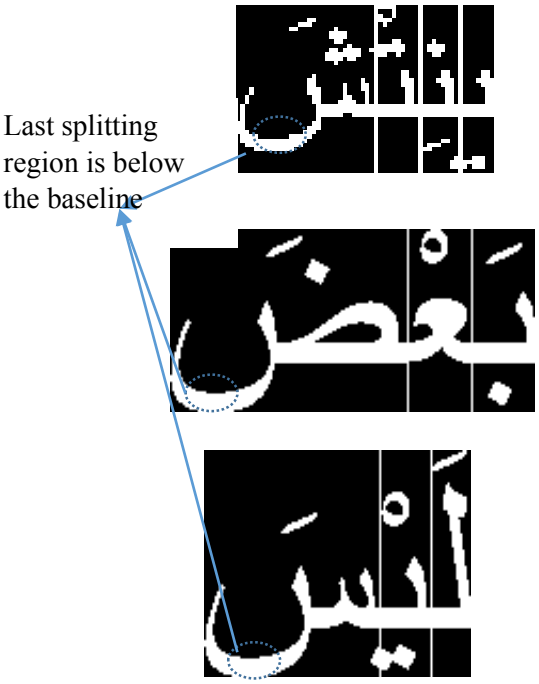


Figure 3-35 Last splitting region is below the baseline so it is ignored, also the last part is merged after it is detected as part of other character like SEEN, SHEEN, SAD or DAD.

SHEEN (ش) case detection:

The algorithm for post processing depends mainly on finding character's part with some specifications, like not having dots and not having holes, these specifications are met on the third part of the SHEEN (ش) character. When the checking sheen case is taking a place, then the previous two parts are determined by the previous two splitting regions, also, the specifications should be met for the previous two characters' parts except not having dots. The expected SHEEN (ش) character should have three dots above (three separated dots or one dot plus two touching dots), and these dots should be overlapped to perform the three dots. Figure 3-36 shows SHEEN (ش) character with three overlapped separated dots, and figure 3-37 shows SHEEN (ش) character with three overlapped dots (one single dot plus two touching dots).

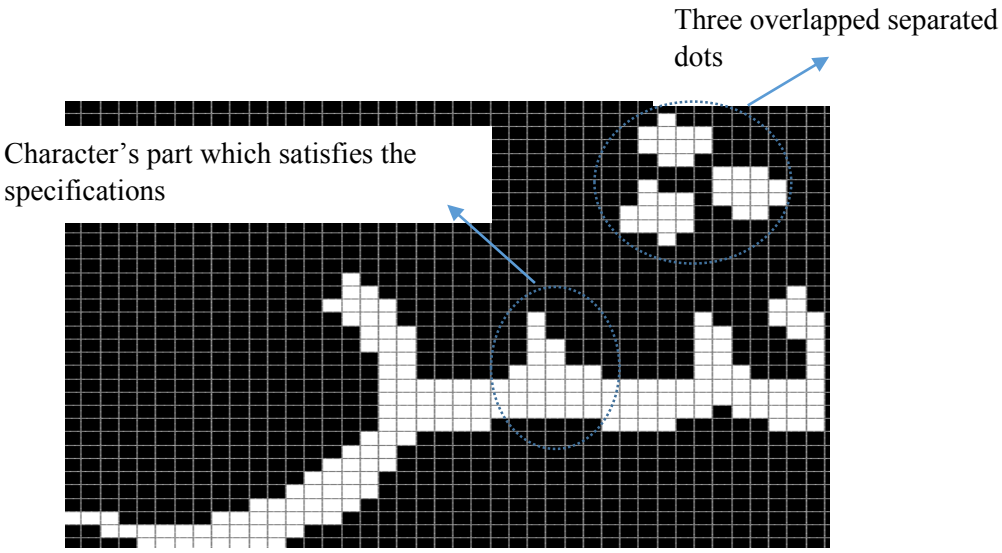


Figure 3-36 SHEEN (ش)case detection by one character's part that satisfies the specifications and three overlapped separated dots.

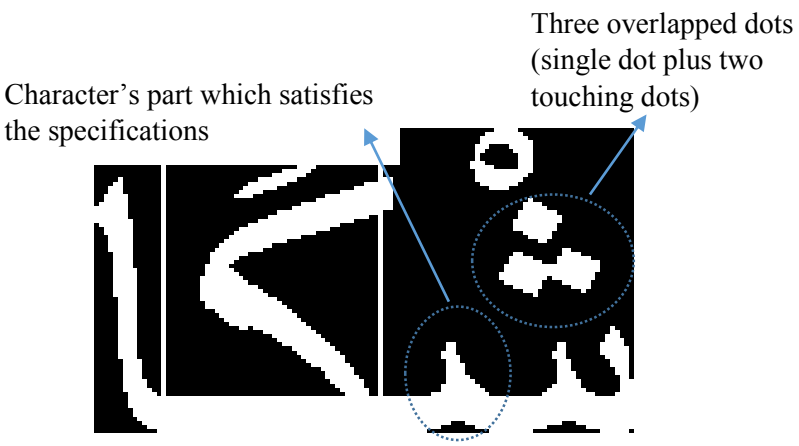


Figure 3-37 SHEEN (ش) case detection by one character's part that satisfies the specifications and three overlapped dots (one single dot and two touching dots).

Algorithm 9 Character Segmentation Post Processing

Input: List of splitting regions (*splitting_regions*)
Result: Each splitting region is assigned by a flag to be ignored or not

```
begin
    for i ← 1 to length(splitting_regions) do
        if Splitting_regions(i).Locates_above(Baseline)) then
            //solves the TAA and THAA cases
            Splitting_regions(i).ignore ← true
            return
        end
        Character_part ← part.Locates_between(splitting_regions(i)ˢsplitting_regions(i +1));
        if Character_part.EulerNumber >= 1 and not (Character_part.has_dot_ind) and
         Character_part.Height < 2 ∗ pen_size then
            // conditions are satisfied
            if sheen_ind then
                // Ignore the previous two splitting regions.
                Splitting_regions(i − 1).ignore ← true  Splitting_regions(i -2).ignore ← true
            else
                Space_counter ++;
            end
        else
            // conditions are not satisfied
            if Is_last_one(splitting_regions(i)) then
                if locates_under_baseline(splitting_regions(i)) then
                    splitting_regions(i).Ignore ← true
                    // check if this part is a part of another character like SEEN, SHEEN, SAD, and DAD
                    if sheen_ind then
                        Splitting_regions(i − 1).ignore ← true  Splitting_regions(i -2).ignore ← true
                    else if Space_counter == 2 then
                        // SEEN case
                        Splitting_regions(i − 1).ignore ← true  Splitting_regions(i -2).ignore ← true
                    else if Space_counter == 1 then
                        // SAD or DAD case
                        Splitting_regions(i − 1).ignore ← true
                end
            else
                // general case
                if Space_counter == 3 then
                    // SEEN case
                    Splitting_regions(i − 2).ignore ← true
                    Splitting_regions(i − 3).ignore ← true
                else if Space_counter == 1 then
                    // SAD or DAD case
                    Splitting_regions(i − 2).ignore ← true
                end
            end
        end
    end
end
```

Chapter 4 Testing and Results

Two datasets and specific tools were used to measure the accuracy of the proposed algorithms. In section 4.1 the tools are described, and the datasets and their structure are described in section 4.2, and finally, the accuracy and comparison results are mentioned in section 4.4.

4.1 *Tools*

4.1.1 *Software*

The code was implemented using MATLAB software program version R2013a, it provides an image processing tool, plus an easy representation for image by using matrix operations.

4.1.2 *Hardware*

The code was run on a Pentium 2.16GHz laptop with 4 G RAM.

4.2 *Datasets Description*

Two datasets were used for testing purpose, the first one was **APTID / MF** (Arabic Printed Text Image Database / Multi-Font), which is used by researchers as a benchmark dataset for Arabic OCR different purposes, and the second dataset was created in this research for testing purpose.

4.2.1 *APTID / MF Dataset*

APTID / MF dataset consists of 1,845 Arabic printed image text-blocks, that contains of 126,792 different Arabic words, written in 10 font types (Andalus, Arabic Transparent, AdvertisingBold, Diwani, DecoType Thuluth, Simplified Arabic, Tahoma, Traditional Arabic, DecoType Naskh and M Unicode Sara), two

different styles (regular and bold), and 4 font-sizes (12, 14, 16 and 18 points), the dataset also includes XML files as a meta data for these images [34].

4.2.2 *Thesis (Local) Dataset*

This dataset consists of different images for printed Arabic text with and without diacritics written in 5 different font types (Simplified Arabic, Times New Roman, Arial, Advertising Bold, and Arabic Transparent), styles (regular, bold and italic), and sizes (8,9,10,12,14,16,18 and 24 points), these images are with 300 dpi resolution. Table 4-1 shows an example of these font types with different styles.

Table 4-1 Font types and styles that have been used in the local dataset

Font name	**Regular**	**Bold**	**Italic**
Simplified Arabic	بسم الله الرحمن الرحيم	**بسم الله الرحمن الرحيم**	*بسم الله الرحمن الرحيم*
Times New Roman	بسم الله الرحمن الرحيم	**بسم الله الرحمن الرحيم**	*بسم الله الرحمن الرحيم*
Arial	بسم الله الرحمن الرحيم	**بسم الله الرحمن الرحيم**	*بسم الله الرحمن الرحيم*
Advertising Bold	**بسم الله الرحمن الرحيم**	**بسم الله الرحمن الرحيم**	***بسم الله الرحمن الرحيم***
Arabic Transparent	بسم الله الرحمن الرحيم	**بسم الله الرحمن الرحيم**	*بسم الله الرحمن الرحيم*

4.3 *Performance Metrics*

The performance for segmentation is measured in terms of line segmentation rate, word segmentation rate and character segmentation rate. Line segmentation rate is the ratio of the number of lines that are correctly segmented to the total number of lines. Word segmentation rate is the ratio of the number of words that are correctly segmented to the total number of words. Character segmentation rate is 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, and also, last character with no splitting area is not considered.

4.4 *Testing Results*

Each segmentation stage is tested using some data of the previously mentioned datasets. The OCR segmentation code runs sequentially, that means to segment into characters then line, word/sub-word and diacritics segmentation codes are also executed. The testing is done manually by eye and defining counters variables in the code to count the totals of segmented lines, words and characters.

4.4.1 *Line Segmentation Results*

Table 4-2 Line segmentation results for regular style with different font sizes (local dataset).

	Font name	**No. of lines correctly segmented**	**Total number of lines**	**Accuracy**
Regular	Simplified Arabic	3421	3446	99.3%
	Times New Roman	1836	1852	99.1%

	Arial	1818	1834	99.1%
	Advertising Bold	1832	1846	99.2%
	Arabic Transparent	1834	1845	99.4%
	TOTALS	**10741**	**10823**	**99.2%**

Table 4-3 Line segmentation results for bold style with different font sizes (local dataset).

	Font name	**No. of lines correctly segmented**	**Total number of lines**	**Accuracy**
Bold	Simplified Arabic	2979	2989	99.7%
	Times New Roman	1828	1840	99.3%
	Arial	1823	1836	99.3%
	Advertising Bold	1828	1844	99.1%
	Arabic Transparent	1819	1835	99.1
	TOTALS	**10277**	**10344**	**99.4%**

Table 4-4 Line segmentation results for italic style with different font sizes (local dataset).

	Font name	**No. of lines correctly segmented**	**Total number of lines**	**Accuracy**
Italic	Simplified Arabic	3020	3034	99.5%
	Times New Roman	1834	1852	99 %
	Arial	1816	1834	99%

	Advertising Bold	1835	1846	99.4%
	Arabic Transparent	1833	1845	99.4%
	TOTALS	**10338**	**10411**	**99.3%**

Table 4-5 Line segmentation results summary (local dataset).

Summary	**No. of lines correctly segmented**	**Total number of lines**	**Accuracy**
	31356	31578	99.3%

Figure 4-1 An example of line segmentation output

4.4.2 *Word/Sub-word Segmentation Results*

Table 4-6 Word segmentation results for regular style with different font sizes (local dataset).

	Font name	**No. of words correctly segmented**	**Total number of words**	**Accuracy**
Regular	Simplified Arabic	1564	1572	99.5%
	Times New Roman	1544	1554	99.4%
	Arial	1541	1550	99.4%
	Advertising Bold	1333	1566	85.1%
	Arabic Transparent	1542	1548	99.6%
	TOTALS	**7524**	**7790**	**96.5%**

Table 4-7 Word segmentation results for bold style with different font sizes (local dataset).

	Font name	**No. of words correctly segmented**	**Total number of words**	**Accuracy**
Bold	Simplified Arabic	1563	1571	99.5%
	Times New Roman	1550	1556	99.6%
	Arial	1543	1549	99.6%
	Advertising Bold	1336	1546	86.4%
	Arabic Transparent	1540	1548	99.5%
	TOTALS	**7532**	**7770**	**96.9%**

Table 4-8 Word segmentation results for italic style with different font sizes (local dataset).

	Font name	No. of words correctly segmented	Total number of words	Accuracy
Italic	Simplified Arabic	1558	1572	99.1%
	Times New Roman	1538	1554	99%
	Arial	1534	1550	99%
	Advertising Bold	1324	1566	84.5%
	Arabic Transparent	1534	1548	99.1%
	TOTALS	**7488**	**7790**	**96.1%**

Table 4-9 Word segmentation results summary (local dataset).

Summary	No. of words correctly segmented	Total number of words	Accuracy
	22544	23350	96.5%

Table 4-10 Word segmentation results (APTID / MF dataset).

	Font name	No. of words correctly segmented	Total number of words	Accuracy
Regular	Simplified Arabic	2859	2871	99.6%
	Advertising Bold	1176	1274	92.3%
	Arabic Transparent	1693	1698	99.7%
	TOTALS	**5728**	**5843**	**98%**

Figure 4-2-1 Simplified Arabic regular style as input

Figure 4-2-2 Simplified Arabic regular style word segmentation result.

Figure 4-2-3 Simplified Arabic bold style as input

Figure 4-2-4 Simplified Arabic bold style word segmentation result.

Figure 4-2-5 Simplified Arabic italic style as input

Figure 4-2-6 Simplified Arabic italic style word segmentation result.

Figure 4-2 Simplified Arabic font type word segmentation result with different styles (Regular, Bold, and Italic)

4.4.3 *Character Segmentation Results*

Table 4-11 Character segmentation results for regular style with different font sizes (local dataset)

	Font name	**No. of characters correctly segmented**	**Total number of characters**	**Accuracy**
Regular	Simplified Arabic	3812	3874	98.4%
	Times New Roman	3788	3862	98.1%
	Arial	3792	3864	98.1%
	Advertising Bold	3786	3854	98.2%
	Arabic Transparent	3798	3866	98.2%
	TOTALS	**18976**	**19320**	**98.2%**

Table 4-12 Character segmentation results for bold style with different font sizes (local dataset)

	Font name	**No. of characters correctly segmented**	**Total number of characters**	**Accuracy**
Bold	Simplified Arabic	3804	3874	98.2%
	Times New Roman	3780	3862	97.9%

	Arial	3782	3864	97.9%
	Advertising Bold	3780	3854	98.1%
	Arabic Transparent	3796	3866	98.2%
	TOTALS	**18942**	**19320**	**98%**

Table 4-13 Character segmentation results for italic style with different font sizes (local dataset)

	Font name	**No. of characters correctly segmented**	**Total number of characters**	**Accuracy**
Italic	Simplified Arabic	3800	3874	98.1%
	Times New Roman	3776	3862	97.8%
	Arial	3780	3864	97.8%
	Advertising Bold	3774	3854	97.9%
	Arabic Transparent	3794	3866	98.1
	TOTALS	**18924**	**19320**	**98%**

Table 4-14 Character segmentation results summary (local dataset).

Summary	**No. of characters correctly segmented**	**Total number of characters**	**Accuracy**
	56842	57960	98.1%

Table 4-15 Character segmentation results (APTID / MF dataset)

	Font Name	No. of characters correctly segmented	Total number of characters	Accuracy
Regular	Simplified Arabic	10847	11064	98 %
	Arabic transparent	6222	6326	98.4%
	Advertising Bold	4961	5062	98 %
	TOTALS	**27476**	**27976**	**98.2%**

Table 4-16 Execution Time for different segmentation stages with and without diacritics.

	Segmentation stage	Total units count	Execution time (second)	Average execution time / unit (second/unit)
With Diacritics (Tashkel)	**Line Seg.**	**23**	**1.017699**	**.044**
	Sub-words + Diacritics Seg.	**604**	**33.4098**	**.055**
	Character Seg.	**990**	**155.2239**	**.157**
Without Diacritics (Tashkel)	**Line Seg.**	**23**	**0.939642**	**0.041**
	Sub-words +Diacritics Seg.	**607**	**18.9524**	**0.031**

	Character Seg.	**986**	**158.8897**	**0.161**

Table 4-17 Character segmentation sample images.

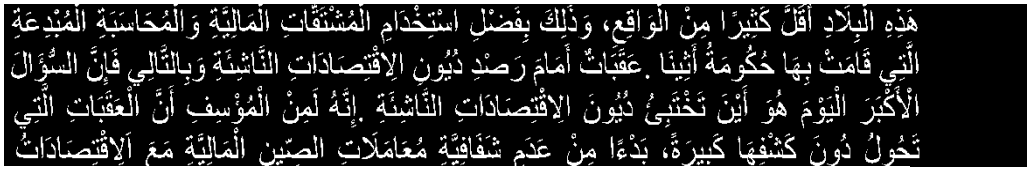


Figure 4-3-1 Input image.

Figure 4-3-2 Line segmentation output.

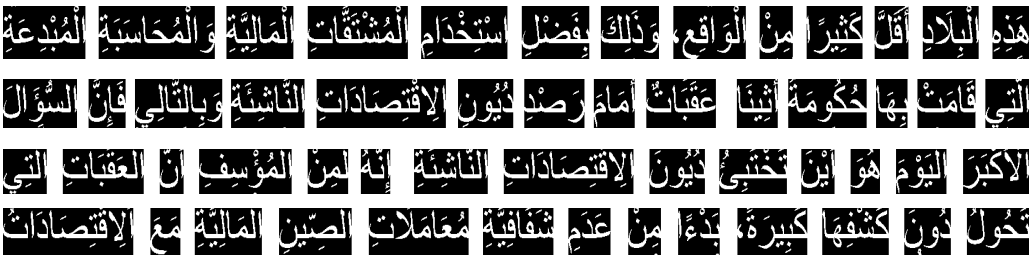


Figure 4-3-3 Word segmentation output.

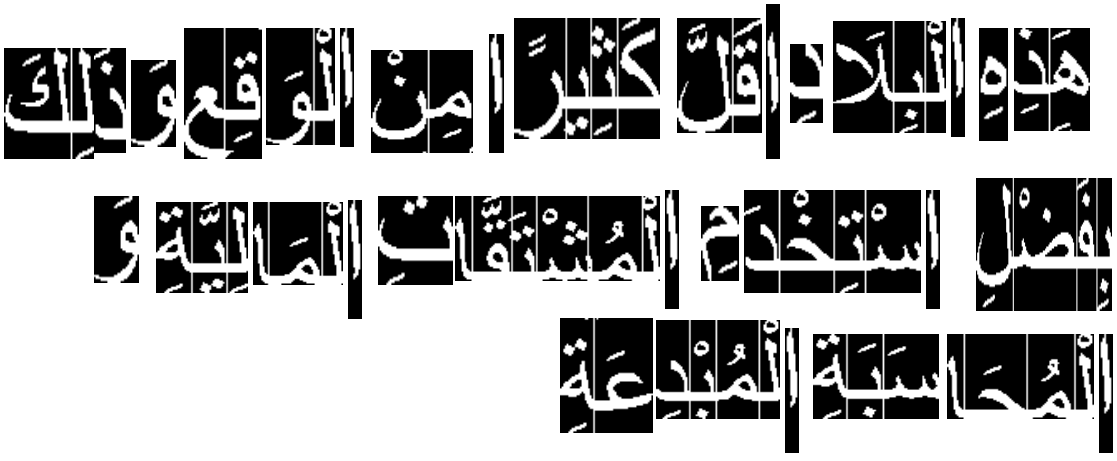


Figure 4-3-4 Character segmentation output for line 1.

Figure 4-3 Example of AOCR output in different stages

4.5 *Source of Failure*

The following cases leads to segmentation errors

- A line with small width followed by line written in large font (large width) and the spacing between them equals to zero, this leads to line segmentation error. In this case, the segment contains lines with different sizes after initial splitting with high threshold value, as shown in figure 4-4.

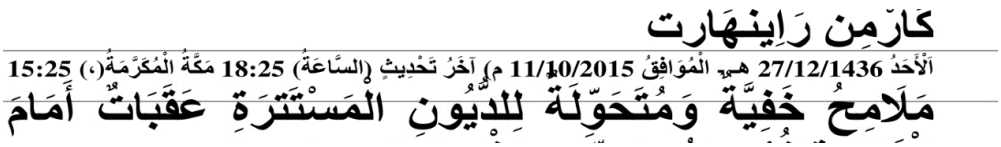


Figure 4-4 Line segmentation failure due to the variance in font sizes with no spacing

- The global maximum peak locates at line touches with another one, leads to wrong width estimation then line segmentation error as shown in figure 4-5.

Figure 4-5 Line segmentation failure due to the touching with other line for the line locates at global maximum peak

- Error in calculating the pen size leads to wrong word / sub-word segmentation as shown in figure 4-6.

Figure 4-6 Word segmentation error due to the pen-size calculation error

- For small font sizes an error happen in detection the dot or two touching dots leads to character segmentation error as shown figure 4-7.

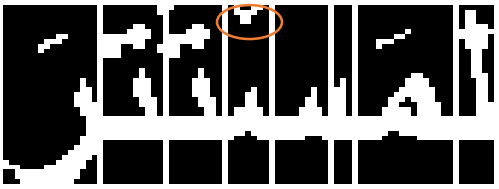


Figure 4-7 Character segmentation error due to an error in dots detection

- Small parts from another line in line segmentation stage sometimes are recognized as dots, and this leads to segmentation error as shown in figure 4-8.

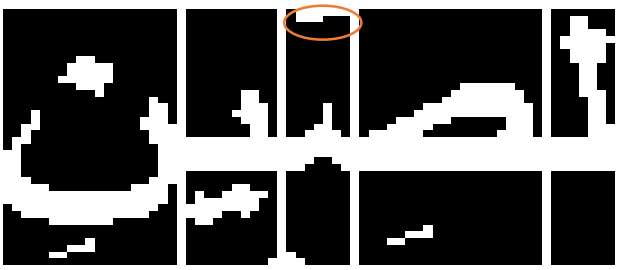


Figure 4-8 Character segmentation error due to small parts from other line

- Absence of splitting region as shown in figure 4-9.

Figure 4-9 Character segmentation error due to the absence of the splitting region

Chapter 5 Conclusion and Future Work

5.1 *Conclusion*

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 thesis, 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 (line segmentation, word/sub-word and diacritics segmentation, and character segmentation). Promising results were achieved by using these proposed methods. the proposed scheme addresses the main problems in Arabic OCR different segmentation stages, for line segmentation stage, two main problems are addressed and solved; the overlapping and the over segmentation problems, the proposed algorithm shows excellent results for documents with diacritics and without diacritics, it shows up to 99%.

Also, an enhanced method for word, sub-word and diacritics segmentation is proposed, the sub-words are extracted in two ways according the sub-words situation. Vertical projection is used in case of full separation between sub-words by finding the gaps between them, the connected component concept is used to find the sub-words in case of overlapping, also the connected components concept is used to extract the diacritics. The proposed method also determines if the sub-words are related to the same word or to different words regardless to the font type or size, by estimating the pen size for each sub-word. The algorithm shows promising results up to 98%.

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 for character shape and details even for small fonts, also the errors in extracting the baseline are eliminated since no need to adjust the baseline many times [14]. A post processing step is needed to solve over segmentation problem; ignore cases checking algorithm is developed in easy and reliable way that can fit many font types and styles. Character segmentation algorithm shows good results up to 98%.

5.2 *Future Work*

AOCR is still an open area for research, further work to segmentation step may include:

- Working on the points that cause errors and failure in the current approach -they are mentioned in the previous section-.
- Working on character recognition stage to get rid of errors that appear in the segmentation stage.
- Enhancing the character segmentation post processing by using some feedback to enhance the segmentation.
- Working on the pre-processing step, like the skew detection and correction, noise removal and document analysis.
- Enhancing the given ideas in the thesis to get better results.

References

1. Zeki, A., Zakaria, M., and Liong, C. "Segmentation of Arabic character," *International Journal of Technology Diffusion* 2, no. 4 (2011): 48–82.
2. Priyanka, N., Pal, S., and Mandal, R. "Line and Word Segmentation Approach for Printed Documents," *IJCA* 2, no. 110119 (2010): 30–36.
3. Cheung, A., Bennamoun, M., and Bergmann, N. "An Arabic optical character recognition system using recognition-based segmentation," *Pattern Recognition* 34, no. 2 (2001): 215–233.
4. Aljarrah, I., Al-Kahaleel, O., Mhaidat, K., Alrefai, M., Alzu'bi, A., and Rabab'ah, M. "Automated system for Arabic optical character recognition with lookup dictionary," *Journal of Emerging Technologies in Web Intelligence* 4, no. 4 (2012): 362–370.
5. Elaiwat, S., Abu-zanona, M., and Al-Zawaideh, F., "A Three Stages Segmentation Model for a Higher Accurate off-line Arabic Handwriting Recognition", *World of Computer Science and Information Technology Journal* 2, no. 3 (2012): 98–104.
6. Abuhaiba, I., "Segmentation of Discrete Arabic Script Document Images", *Journal of Al Azhar University–Gaza (Natural Sciences)* 8 (2006): 85–108.
7. Soujanya, P., Koppula, V., Gaddam, K., and Sruthi, P. "Comparative Study of Text Line Segmentation Algorithms on Low Quality Documents", *International Journal of Computer Science & Informatics* 2, no. 2 (2016): 110–116.

8. Shafii, M. “Optical Character Recognition of Printed Persian/Arabic Documents”, Ph.D. dissertation, department of Electrical and Computer Engineering, University of Windsor, 2014.
9. Al-Badr, B., and Haralick, R. "A segmentation-free approach to text recognition with application to Arabic text", *International Journal on Document Analysis and Recognition* 1, no. 3 (1998): 147–166.
10. Naz, S., Umar, A., Shirazi, S., Ahmed, S., Razzak, M., and Siddiqi, I., "Segmentation techniques for recognition of Arabic-like scripts: A comprehensive survey", *Education and Information Technologies*, (2015).
11. Omidyegane, M., Nayebi, K., Azmi, R., and Javadtalab, A., "A NEW SEGMENTATION TECHNIQUE FOR MULTI FONT FARSI/ARABIC TEXTS", *ICASSP* 2, (2005): 757–760.
12. Mostafa, M. “An Adaptive Algorithm for the Automatic Segmentation of Printed Arabic Text.” *17th National Computer Conference*. Saudi Arabia, 2004.
13. Shaikh, N., Mallah, G., and Shaikh, Z., "Character Segmentation of Sindhi, an Arabic Style Scripting Language, using Height Profile Vector", *Australian Journal of Basic and Applied Sciences* 3, no. 4 (2009): 4160–4169.
14. Alipour, M., "A new Approach to Segmentation of Persian Cursive Script based on Adjustment the Fragments", *International Journal of Computer Applications* 64, no. 11 (2013): 21–26.
15. Kchaou, M., Kanoun, S., OGIER, J. "Segmentation and Word Spotting Methods for Printed and Handwritten Arabic Texts: A Comparative Study" *The*

International Conference on Frontiers in Handwriting Recognition. Bari: IEEE, 2012.

16. Dinges, L., Al-Hamadi, A., Elzobi, M., Al Aghbari, Z., and Mustafa, H., "Offline Automatic Segmentation based Recognition of Handwritten Arabic Words", *International Journal of Signal Processing, Image Processing and Pattern Recognition* 4, no. 4 (2011): 131–144.
17. Hussain, S., Ali, S., and Ul Akram, Q., "Nastalique segmentation-based approach for Urdu OCR", *International Journal on Document Analysis and Recognition* 18, no. 4 (2015): 357–374.
18. Hakro, D., Talib, A., Bhatti, Z., and Moja, G., "A Study of Sindhi Related and Arabic Script Adapted Languages Recognition", *Sindh University Research Journal (Science Series)* 46, no. 3 (2014): 323–334.
19. Al-A'ali, M., and Ahmad, J., "Optical character recognition system for Arabic text using Cursive multi-directional approach", *Journal of Computer Science*, 3, no. 7 (2007): 549–555.
20. Naz, S., Hayat, K., Razzak, M., Anwar, M., and Akbar, H. "Arabic Script Based Character Segmentation: A review", in *Computer and Information Technology (WCCIT), 2013 World Congress on*, Sousse: IEEE, (2013): 1–6.
21. "Optical Character Recognition," Wikipedia, 2016, accessed December 25, 2016, http://en.wikipedia.org/wiki/Optical_character_recognition.
22. Hussain, A., Jian-hua, H., and Xiang-long, T., "Offline Arabic Recognition System," *Journal of Harbin Institute of Technology (New Series)* 10, no. 1, (2003): 80–88.

23. Xiu, P., Peng, L., Ding, X., and Wang, H. “Offline Handwritten Arabic Character Segmentation with Probabilistic Model.” *Document Analysis Systems VII*. New Zealand: Nelson, 2006.
24. Baugher, C. “OCR Software from ABBYY. Best Text Recognition for Windows and Mac.” 2014. Accessed November 20, 2016. https://www.abbyy.com/en-apac/finereader/.
25. Dhannoon, B., Mohammed, I., and Dhamad, M. "A New Segmentation Technique of Handwritten Offline Arabic Text", *International Journal of Scientific & Engineering Research* 5, no. 7 (2014): 228–233.
26. AlKateeb, J., Jiang, J., Ren, J., and Ipson, S. "Component-based Segmentation of Words from Handwritten Arabic Text," *International Journal of Computer Systems Science and Engineering* 5, no. 1 (2009): 54–58.
27. Javed, S., Hussain, S., Maqbool, A., Asloob, S., Jamil, S. and Moin, H. "Segmentation Free Nastalique Urdu OCR", *World Academy of Science, Engineering and Technology*, 2010.
28. Garg, R., and Garg, N. "An algorithm for Text Line Segmentation in Handwritten Skewed and Overlapped Devanagari Script", *International Journal of Emerging Technology and Advanced Engineering* 4, no. 5 (2014): 114–118.
29. "Lexicon-Driven Word Recognition Based on Levenshtein Distance”, *International Journal of Software Engineering and Its Applications* 8, no. 2 (2014): 11–20.

30. Lawgali, A. "A Survey on Arabic Character Recognition", *International Journal of Signal Processing, Image Processing and Pattern Recognition* 8, no. 2 (2015): 401–426.
31. Sahloul, A. and Suen, C. "Off-line system for the recognition of handwritten Arabic character", *Computer Science & Information Technology (CS & IT)*, (2014): 227–244.
32. Humied, I. "Segmentation accuracy for Offline Arabic handwritten recognition based on bounding box algorithm", *International Journal of Computer Science and Network Security* 16, no. 9 (2016): 98–109.
33. Mohammad, K., Ayyash, M., Qaroush, A. and Tumar, I. "Printed Arabic Optical Character Segmentation", *Image Processing: Algorithms and Systems XIII*, (2015).
34. Jaiem, F., Kanoun, S., Khemakhem, M., El Abed, H. and Kardoun, J. "Database for Arabic Printed Text Recognition Research", *ICIAP,* (2015): 251- 259.