Journal of King Saud University – Computer and Information Sciences 34 (2022) 3025–3039

Contents lists available at ScienceDirect

Journal of King Saud University – Computer and Information Sciences

journal homepage: www.sciencedirect.com

Segmentation-based, omnifont printed Arabic character recognition without font identification

Aziz Qaroush *, Abdalkarim Awad, Mohammad Modallal, Malik Ziq

Department of Electrical and Computer Engineering, Birzeit University, Palestine

ARTICLE INFO

Article history:
Received 22 July 2020
Revised 26 September 2020
Accepted 3 October 2020
Available online 10 October 2020

Keywords:
Optical character recognition
Arabic OCR
Mono-font
Mixed-font
Character segmentation
Feature extraction
Convolutional neural network

ABSTRACT

Optical Character Recognition OCR is an essential part of many real-world applications such as digital archiving, automatic number plate recognition, handle cheques, etc. However, developing an OCR for printed Arabic text is still a challenging and open research field due to the special characteristics of Arabic cursive script. In this paper, we propose a segmentation-based, omnifont, open-vocabulary OCR for printed Arabic text. The proposed approach doesn't require an explicit font type recognition stage. It uses an explicit, indirect character segmentation method. The presented segmentation method is baseline dependent and employs a hybrid, three-steps character segmentation algorithm to handle the problem of character overlapping. Besides, it uses a set of topological features that are designed and generalized to make the segmentation approach font independent. The segmented characters are fed as an input to a convolutional neural network for feature extraction and recognition. The APTID-MF data set has been used for testing and evaluation. The average accuracy of the proposed segmentation stage is 95%, while the average accuracy of the recognition stage is 99.97%. The whole approach achieves an average accuracy of 95% without using font-type recognition or any post-processing techniques.
© 2020 The Authors. Published by Elsevier B.V. on behalf of King Saud University. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

1. Introduction

The existence of a large number of paper documents in the form of books, historical documents, magazines, newspapers, and notes increases the need for an efficient Optical Character Recognition system (OCR) to convert text images into an editable form or into digital archives (Zoizou et al., 2018). Such conversion allows for automatic processing, searching, mining, easy backup facilities, and eliminate the need for physical storage of printed documents. In addition, the need for reliable OCR systems has increased due to the existence of many data entry applications and information retrieval systems such as search engines. Moreover, OCR systems can improve and automate many human–computer interaction applications which will increase human productivity such as mail sorting, questionnaires processing, exam papers correction, forms processing, bank verification processing, security identification (i.e. license plate recognition system), and many other applications (Perwej et al., 2014; Lawgali, 2015; Printed arabic script recognition, xxxx; Islam et al., 2017). This makes OCR as one of the most researched pattern recognition problems.

Development OCR systems started in 1970 with the beginning of the invention of the retina scanner. It is the process of transferring automatically a type or handwritten text-image captured by a camera or scanner to a computer editable text to avoid retyping, which leads to increase data usage and saves individuals and businesses time and money (Qaroush et al., 2019). Generally, developing OCR systems consists of six sequencing main stages as shown in Fig. 1 (Lawgali, 2015; Printed arabic script recognition, xxxx; Islam et al., 2017; Lorigo and Govindaraju, 2006): image acquisition, preprocessing, layout analysis and segmentation, feature extraction, recognition, and post-processing. According to the text-image acquisition way, OCR systems are classified into online and offline systems. In online systems, the input text-image (e.g. word or character) is taken from pen-based devices such as signpad or cell phones immediately while users are writing. On the other hand, in offline systems, the input is usually a stored text-image taken by a camera, scanner, or other optical devices. Recognition in online systems are performed in real-time and is less complex because there is no need for some preprocessing

* Corresponding author.
E-mail addresses: aqaroush@birzeit.edu (A. Qaroush), akarim@birzeit.edu (A. Awad).
Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.jksuci.2020.10.001
1319-1578/© 2020 The Authors. Published by Elsevier B.V. on behalf of King Saud University.
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

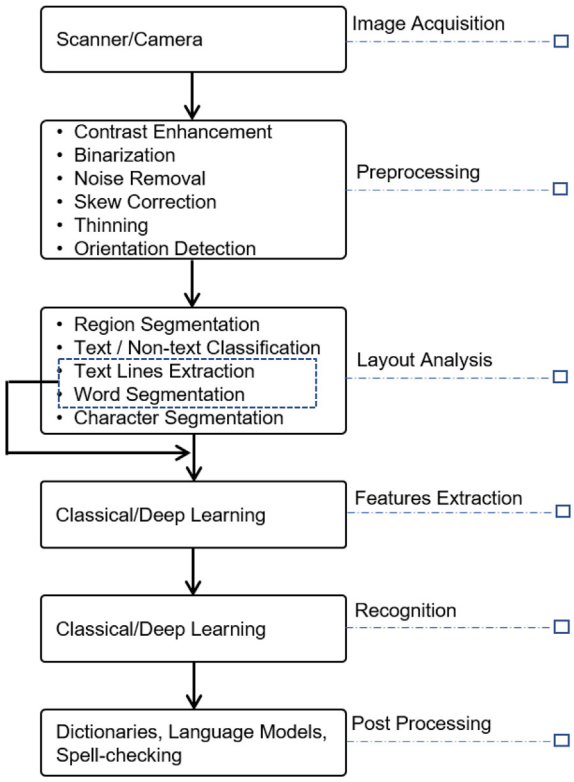

Fig. 1. A block diagram of a typical OCR system main stages. The diagram shows intermediate steps during OCR main stages.

techniques such as noise removal and thinning. In addition. it can capture more temporal or time-based information such as speed, the number of strokes made, and direction of the writing of strokes (Islam et al., 2017). In the pre-processing stage, a set of usually font-independent methods were applied such as contrast enhancement, noise removal, binarization, skew correction, slant correction, and thinning to produce a cleanup and unified version of the input image. Layout analysis aims at segmenting text-image into regions and then classify these regions into text regions and non-text regions (e.g. an image). After that, text regions are further segmented into lines, words, characters, and strokes depending on the script under study and the segmentation methodology used. The segments resulted from the layout analysis stage such as words and characters are processed and analyzed in the features extraction stage to extract a set of statistical, structural, or global transformation features. This information is then passed to the classifier which represents the recognition stage to identify these segments. Finally, a series of post-processing methods are applied to enhance recognition accuracy by incorporating language dictionaries, language models, and spell-checking method.

Building OCR systems face several generals (language independent) challenges such as the quality of the input image (e.g. existence of noise, blurring, skewness, slant, and degradation), font variations (e.g. type, size, and style), the existence of non-character objects and layout complexity of the scanned image (Hamad and Kaya, 2016). In addition, the Arabic language has a set of unique features that made the character segmentation and recognition task more challenging. The features of Arabic script includes the following characteristics and uniqueness (Ahmed and Al-Ohali, 2000; Zeki and Zakaria, 2004; Mahmood, 2013): (i) Arabic text is written cursively from right to left which complicate the process of character segmentation which treated as the most crucial step of any cursive OCR systems, (ii) the availability of complex font types (e.g. Thuluth, Diwani, Andalusi and Naskh) makes the shape and the contour of the characters irregular and diverse, (iii) the existence of diacritics called "Harakat" increase the overlapping between adjacent characters and thus increase under segmentation problems, (iv) most of the characters in Arabic have four forms according to their position in the word (at the beginning, middle, end, or separated) which increase the number of classes used in the recognition stage, (v) Arabic characters do not have a fixed size in terms of height and width, (vi) in some font types such as "Thuluth" and "Naskh" some consecutive connected characters cannot be segmented and thus combined as a new shape called Ligature which is not similar to the result of concatenating the two basic shapes horizontally, (vii) the length of connection strokes vary in different fonts, (viii) several Arabic characters can overlap with the next character in the same or adjacent word and (ix) in some fonts or writing styles, the strokes of some characters like "SEEN character" are omitted to give them a non-standard shape. These unique features make Arabic OCR more challenging, and thus the research in the field of Arabic OCR moves slowly.

Several research methods have been presented in the literature to address the Arabic character segmentation and recognition problem. Besides, some of the Arabic OCR systems are available commercially (e.g. Sakhr, READIRIS PRO, ABBYY FineReader, etc...) or as an open-source (e.g. TESSERACT, KRAKEN, etc...) or free online OCR (e.g. GoogleDocs, i2OCR, etc...) (Zoizou et al., 2018). However, the existing methods and systems have many weaknesses summarized in (i) they are below the desired level of performance specifically for omnifont approach compared to other languages, (ii) some of them are font (type, size, and style) dependent or mono font, and cannot handle complex font types such as Thuluth, Diwani, Andalusi and Naskh, which widely used in old books, (iii) some of the research methods were tested on simple and small unpublished data-sets, and (v) most of them are based on using classical methods in the features extraction and recognition stages. In this paper, we present a segmentation-based, omnifont, open-vocabulary OCR for printed Arabic text. The presented approach uses an explicit, hybrid character segmentation-based method along with using deep learning for feature extraction and recognition stages. Our character segmentation and text recognition methods are font independent without requiring the font recognition stage. Besides, the proposed approach performed on continuous text lines images and the recognition model is an open vocabulary text recognition approach. We also evaluated our approach using APTID-MF (Jaiem et al., 2013) dataset to demonstrate the robustness of the presented methods. The proposed approach has experimented on 50 text blocks with sufficient font variations. The proposed segmentation method achieved an average accuracy of 95%, while the achieved average accuracy of the recognition stage is 99.97%. The whole approach achieved an average accuracy of 95% without using font-type recognition or any post-processing techniques. Hence, the main contributions of this paper can be summarized as follows: First, provide an up-to-date survey and analysis regarding printed Arabic character segmentation, features extraction, and recognition. Second, we present an efficient explicit-based character segmentation method that has the following characteristics: (i) the segmentation method is baseline dependent and uses hybrid segmentation approach of projection profile and connected component methods to handle the problem of overlapping between adjacent characters to reduce the number of ligatures as much as possible, (ii) the segmentation approach uses a set of topological rules that make it font type, size, and style independent, and (iii) optimizes the location of the segmentation points to maintains character shape. Third, we have collected and labeled a dataset of segmented Arabic characters to be used in the experimental evaluation, we will also make this dataset

available for other researchers to use. Fourth, using deep learning methods for feature extraction and recognition stage. Fifth, in the evaluation stage, experimental results on the APTID-MF dataset prove that our method achieves better performance than the state-of-the-art methods that work on complete text lines and proposed for mixed-font types without font type recognition.

This paper is organized as follows: Section 2 presents related works on printed Arabic character segmentation, feature extraction, and text recognition. In Section 3, we present our approaches and techniques for printed Arabic character segmentation, feature extraction, and character recognition. Sections 4 describes the dataset used in the experiments and also introduces and discusses the results along with a comprehensive comparison with similar related works on printed Arabic character segmentation and character recognition. Finally, Section 5 presents our conclusion and future work.

2. Related works

2.1. Character segmentation

Character segmentation is an operation that seeks to segment an input word/sub-word image into sub-images of individual segments such as ligatures, characters, or stroke (Casey and Lecolinet, 1996; Alginahi, 2013). Character segmentation is an important stage in developing OCR systems since it affects directly the success of the feature extraction and recognition stages. The performance of the segmentation algorithm is highly dependent on the nature of language. Indeed, character segmentation is the most crucial, hardest, and most time-consuming stage of any OCR system especially for cursive scripts because of the presence of touching and overlapping characters. Character segmentation can be done in two approaches namely explicit segmentation and implicit segmentation as shown in Fig. 2 (Casey and Lecolinet, 1996; Alginahi, 2013; Printed arabic script recognition, xxxx; Naz et al., 2016b; Zeki et al., 2011). The explicit segmentation approach segments a word image into a set of small components while the implicit segmentation approach combines the segmentation stage and recognition stage through segments the words into characters and recognizes them simultaneously.

2.1.1. Explicit segmentation

In explicit-based or dissection strategy segmentation, the words are segmented into smaller independent units, such as ligatures, characters, or strokes (part of the character), based on a set of a predefined hypothesis, characteristics, or rules which are used to determine the validity of the segmentation points (Lawgali, 2015; Alginahi, 2013; Choudhary, 2014). It is further classified into two sub approaches direct segmentation and indirect segmentation. In the former, the word is directly segmented into ligatures and characters, while in the latter, the word is divided into smaller segments which could be characters or parts of characters such as strokes (i.e dots, diacritics). Then, the generated strokes are merged by exploiting a set of features such as starting points, ending points, points of a sudden change in the contour, cusps, open curves, closed curves, and others. The explicit-based approach has the advantage of minimizing the under-segmentation problem (e.g reduced number of training classes). In addition, it yields slightly better results than less complex implicit-based segmentation methods and is also more suitable for mixed-font recognition systems (Lawgali, 2015; Alginahi, 2013; Naz et al., 2016b; Rehman et al., 2009). However, explicit-based segmentation methods are computationally complex and language-dependent because different alphabets have different characteristics (Inkeaw et al., 2018).

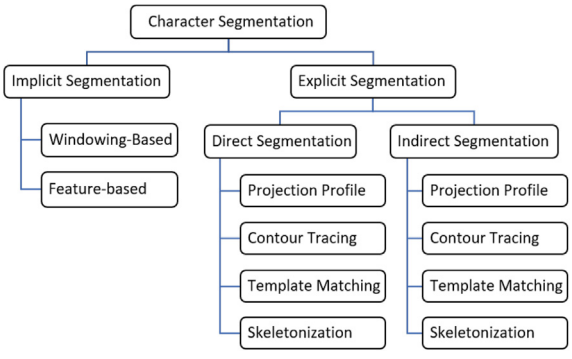

Fig. 2. Arabic Character segmentation methods.

Explicit-based segmentation methods can be classified into a projection profile, contour-based, skeletonization-based, and template matching method. Projection profile methods (Nawaz et al., 2003; Zheng et al., 2004; Zidouri et al., 2005; Shaikh et al., 2009; Anwar et al., 2015; Mahmoud et al., 2017; Marwa Amara and Zidi, 2016) are based on applying a vertical projection profile on the word segment to search for the location of the potential segmentation points based on the fact that the connection stroke between adjacent characters has less thickness than the character itself (Lawgali, 2015; Alginahi, 2013; Naz et al., 2016b). Projection profiles methods are computationally simple and reduce the two-dimensional characters' image into one-dimension information. However, it produces more ligatures for cursive text with overlapping characters and condensed spacing between characters such as in the "Thulth" Arabic font type. In addition, employing vertical projection alone is directly prone to over-segmentation when a character is composed of several parts.

Contour-based methods (Omidyeganeh et al., 2005; Bushofa and Spann, 1997; Mehran et al., 2005; Sari et al., 2002; Romeo-Pakker et al., 1995) extract information about the general shape of the word by finding the word contour which represents the pixels that form the outer shape of the word (Lawgali, 2015; Alginahi, 2013; Naz et al., 2016b). It exploits the representation of the word shape (contour) to locate the potential segmentation point based on the fact that each character consists of a high contour followed by a flat or low contour where the segmentation points located before the contour start rising. This method works fine for fonts that have ligatures and overlapping characters. However, contour tracing is sensitive to noise and character brakes. In addition, it suffers from over-segmentation especially when the characters are composed of several parts.

Skeletonization-based methods (Altuwaijri and Bayoumi, 1998; Motawa et al., 1997; Timsari, 1996; Cowell, 2001; Qomariyah and Mahmudy, 2017; Ahmad, 2007; Mostafa, 2004) employ a set of morphological operations such as opening, closing, and thinning to extract word/character components or features that are used to represent and describe region shape information such as boundaries, curvatures, angles and skeletons (Lawgali, 2015; Alginahi, 2013; Naz et al., 2016b). Morphological segmentation allows breaking cursive text segments into smaller units. In addition, using the thinning operation to generate a skeleton of the words reducing the amount of data to handle. However, since there is an information loss after applying thinning operation, the skeleton of the characters may differ from the original shape, making the segmentation process more difficult.

Template matching methods (Saabni, 2014; Zhang et al., 2013) find the character's potential cutting points based on the sliding window and predefined character templates. It starts by finding the baseline. Then, it searches for matching between the templates and the text-image by sliding the templates over the text (Lawgali, 2015; Alginahi, 2013; Naz et al., 2016b). Template matching

methods work fine for printed text with simple fonts. However, it is sensitive to the dissimilarity in the characters' size and position such that the performance decreases as the number of predefined segments increases when using more font types and styles. In addition, their results are highly dependent on the performance of the pre-processing methods especially the binarization and the normalization operations. Moreover, it is computationally expensive to check all predefined templates.

2.1.2. Implicit segmentation

In implicit or recognition-based segmentation methods (Radwan et al., 2016; Rosenberg, 2012; Bushofa and Spann, 1997; Naz et al., 2016a; Gouda and Rashwan, 2004; Al-Muhtaseb et al., 2008) the word segments are searched for components that match predefined classes in its alphabet (Printed arabic script recognition, xxxx). Therefore, the segmentation stage and the recognition stage are performed simultaneously. In other words, the word segments are segmented into their characters while being recognized without dividing it into a smaller unit (e.g. characters) (Inkeaw et al., 2018). It utilizes feedback from the character classifier model in order to detect segmentation points. Such methods fall into two sub approaches windowing-based and feature-based. Windowing methods segment the word image blindly based on using a mobile sliding window of variable width to provide the temporal segmentation without regarding the image features. Thus, it tries to choose the optimal segmentation by evaluating the classification of the generated sub-images. On the other hand, feature-based methods are based on detecting the physical location of the image features, and then seek to segment this representation into well-classified subsets. Therefore, windowing methods employ recognition to search for "hard" segmentation boundaries, while feature-based are "soft" segmentation boundaries (Casey and Lecolinet, 1996).

Implicit segmentation methods are a simple process and usually language independent. Their accuracy is related to or depends on classification performance. They are used in order to overcome the complexity and the problems of cursive script segmentation (Inkeaw et al., 2018; Rehman et al., 2009). In addition, they are providing all temporary segments and let the recognizer/classifier to choose the best segmentation. Therefore, implicit segmentation-based recognition techniques require a large amount of training data. However, there is a trade-off in selecting a total number of segments for a word segment. Indeed, using a smaller number of segments reduces computational time but increase the problem of under-segmentation. In addition, things get worse when the overlapping between adjacent characters are existing because we need to recognize all possible combinations of valid characters instead of recognizing only valid characters.

On the other hand, using a large number of segments reduces under segmentation problem and thus reduce the number of ligatures. However, it produces more slices which increase computational time and over-segmentation problems. In addition, parts of characters are recognized as a valid character, which is commonly known as a class overlapping problem. Almost all of the implicit-based segmentation methods adopt the second approach since it is easier to merge small segments to form valid character (Inkeaw et al., 2018).

2.2. Feature extraction

Features capture and represent the information extracted from word or character such as pixels, shape information, or statistical properties (Lorigo and Govindaraju, 2006; Kesiman et al., 2016; Soora and Deshpande, 2018; Trier et al., 1996). Feature extraction transforms the input raw data into a reduced representation set of features called a feature vector. Therefore, the main goal of feature extraction is to capture the most relevant information from the original raw data and represent that information in a lower dimensionality space by ignoring redundant and irrelevant information making the task of classifying the pattern (e.g. characters) easy and in a formal way (Kumar and Bhatia, 2014).

Features extraction methods for character recognition can be classified into two approaches, Traditional or classical and deep learning. Traditional methods are mainly based on hand-crafted features. Hand-crafted features are sometimes straightforward and less ambiguous. In addition, they do not require a large training set. However, hand-crafted features require knowledge about the application and thus cannot be transferred easily to other applications. Generally, traditional features can be classified into local or global (Gonzalez and Woods, 2017). For character recognition, it can be classified as shape-based (e.g. geometric, moments, local and spatial) and non-shape-based (e.g. statistical) (Soora and Deshpande, 2018). Another major classification of features extraction in character recognition fall into four sub-categories: structural features (e.g. width, height, holes, etc.), statistical features (e.g. mean, standard deviation, probability distribution, projections, zoning, etc.), Local features (e.g. SIFT, SURF, etc.) and global transformation features (e.g. Fourier transform, moments, etc.) (Lawgali, 2015; Printed arabic script recognition, xxxx; Noushin Najafiragheb and Harifi, 2016).

On the other hand, machine learning especially the deep learning approach learn a feature extractor directly from the input pixels. Therefore, by training a feature extractor model on a data set, the trained model can be easily adapted to many input types and variations. In addition, deep learning (e.g. Convolutional Neural Network (CNN) and Long Short Term Memory (LSTM)) provide a very flexible and universal learnable framework for detecting and representing features. Moreover, this approach is problem independent and didn't need expert knowledge of the data being analyzed. However, using deep learning requires a large data set to train the extractor model.

It is generally agreed that one of the main factors influencing the performance of the classifier is the selection of an appropriate set of features for representing the input images. In addition, the large variability of font variations makes the selection of the appropriate feature sets even more complex. Therefore, several studies use features selection methods to select the best feature subset that achieved the maximum accuracy (Cilia et al., 2019). In fact, feature selection helps in reducing information redundancy, reducing the effect of the curse of dimensionality, and may be improving the classifier performance, Besides, feature selection reduces computational complexity, and produces a simple classification model (Abualigah, 2018; Abualigah et al., 2018, 2017).

2.3. Character recognition

According to the character segmentation method used, character recognition techniques for a cursive script such as Arabic text can be classified into holistic (segmentation-free) and analytical (segmentation-based) approaches (Alginahi, 2013; Naz et al., 2016b; Zeki et al., 2011; Mohammad et al., 2019; Qaroush et al., 2019). In a holistic approach, since characters in the cursive script could be overlapping, slant, and may have different shapes, the whole words are handled as a unified unit without further segmenting the words into its low-level units such as ligatures, characters, strokes, dots, and diacritics. The holistic approach deals with words instead of characters and requires calculating a global feature vector of the input word which is then utilized to classify the word against a stored lexicon of words (Nashwan et al., 2017). However, the main challenge in this approach is a large number of classes present in the recognition stage when dealing with large lexicon size of words, which results in performance degradation as the number of words increases. Therefore, this

approach is not useful for general text recognition and is usually used when targeting to recognize particular words or when the lexicon is statically defined such as numbers, the names of cities, and bank cheque recognition (Printed arabic script recognition, xxxx). On the other hand, the Analytical (segmentation-based) approach relies on segmenting word unit into a sequence of smaller units (usually characters) either explicitly or implicitly (Din et al., 2017). Segmentation-based methods have the advantage of reducing the number of training classes making this approach more general and practical than the holistic approach for real-world problems. However, segmenting cursive scripts into characters especially when the overlapping between character exists is a challenging task and requires more processing.

3. Printed character segmentation and recognition

The proposed approach is an analytical (explicit-based segmentation), open-vocabulary recognition model. It works on continuous text-lines images and consists of three sequencing stages, namely character segmentation, feature extraction, and character recognition. In this section, we present our approach for printed Arabic character segmentation and recognition. Section 3.1 describes our segmentation method and Section 3.1.1 describes the proposed feature extraction and recognition method.

3.1. Character segmentation

The proposed segmentation method is an explicit indirect segmentation-based method. The proposed method takes a printed text-line image as an input and returns a list of their characters or ligatures as an output. It consists of three sequencing stages that are designed carefully to achieve the following objectives: (i) simplicity, (ii) generalized to be font type, size, and style independent, (iii) handling overlapping between characters, and (iv) perform quite well against over-segmentation and under-segmentation problems. It starts with the preprocessing stage that aims to prepare the input for the segmentation stage and includes image binarization, line space trimming, and line partitioning. The second stage is baseline detection and localization where the baseline width is determined. Finally, in the third stage, three segmentation methods were employed to segment the line into characters and ligatures.

3.1.1. Text-line image preprocessing

The line preprocessing stage aims to prepare a cleanup version from the input image and segment the text-line input image into N segments based on the line length. It includes several techniques that are independent of font type and size. It takes a printed Arabic text-line image as input and returns the cleanup binary text-image line segments. The first preprocessing step is image binarization. Image binarization converts a grayscale image to binary form aiming at cleaning the image and reducing image dimensionality. There are several document binarization methods introduced in the literature (Lokhande and Dawande, 2015). In our algorithm, Otsu's algorithm is used to determine the threshold value. Otsu's is an optimal, parameter-free, and global threshold technique (Vala and Baxi, 2013). It works by finding the global threshold value that minimizes the intra-class variance of the resulting black and white pixels. Fig. 3b shows the binarized form of the input image of Fig. 3a.

In the second step, spaces around the text-line image are trimmed. This step aims to find the exact text-line image height and width to find statistics attributes such as character width and height that will be used in the next stages. Space trimming works by removing the spaces around outer white pixels borders.

Fig. 3. Text-line image preprocessing steps.

The outer white pixels borders determined by first applying horizontal and vertical projection on the binarized text-line input image and then finding the first column, last column, first row, and last row that contains only white pixels. Fig. 3c shows the detected borders and Fig. 3d shows the binarized text-line image after trimming spaces.

Finally, the text-line image is partitioned into N segments based on line width as shown in Fig. 3e. Therefore, the text-line image could be segmented into one segment (when the line width is less than or equal one-third of the page width), or two segments (when the line width is between one-third of the page width and two-thirds of the page width), or three segments (when the line width is greater than or equal two-thirds of the page width). The resulting segments may not equal in width and the cutting points are determined using Vertical Projection (VP) specifically when $VP = 0$. Indeed, text in long line images may have different skews that cause an error in the baseline width detection process as shown in Fig. 3f. Therefore, to solve this problem the text-line image is segmented into N segments based on the line image width and then the baseline width detection stage is applied to each segment separately. Fig. 3g shows that the red line which represents the baseline covers the exact baseline width for each segment. Algorithm 1 summarizes text-line preprocessing stages.

Algorithm 1 Text-line image preprocessing

INPUT GLI: as Text-line gray-level image
SET: **BLI** as Text-line binary image
SET: **PW** as page width
$BLI \leftarrow Otsu's(GLI)$;
$HP \leftarrow horizontalProjection(BLI)$;
$VP \leftarrow verticalProjection(BLI)$;
$Borders \leftarrow findBorders(HP, VP)$;
$BLI \leftarrow spaceTrimming(Borders)$;
$LW \leftarrow getLineWidth(BLI)$;
if $(LW <= 0.33 \times PW)$ **then**
    $Segments \leftarrow 1$
else if $(LW <= 0.66 \times PW)$ **then**
    $Segments \leftarrow 2$
else
    $Segments \leftarrow 3$
end if
$LineSegments \leftarrow lineSegmentation(Segments, VP)$;
Output LineSegments

3.1.2. Baseline detection and localization

The baseline is considered one of the distinctive features of any cursive script such as Arabic script. It is a virtual line on which the characters are connected/joined to each other. Baseline detection is an important step in cursive text recognition. It can be used in finding potential segmentation points, skew correction, and feature extraction. The proposed baseline detection stage takes a binary text-line segment image as input and returns the baseline width as an output. It includes two main steps: localizing the initial baseline and finding the baseline width. In the first step, localizing the initial baseline starts by applying horizontal projection on the input text-line segment followed by a smoothing operation to reduce the number of peaks. Finally, the maximum peak value after applying these sequential operations is found which represents the index of the baseline. Fig. 4b shows the horizontal projection without and with smoothing along with the peak value, and Fig. 4c shows the index of the baseline. In the second step, the baseline width is determined by finding the baseline upper and lower indexes (rows indexes), where the upper index is located above the baseline and the lower index is located below the baseline index as shown in Fig. 4c. To find these indexes, we apply the mode operation after computing vertical projection in the image above the baseline index and also on the image below the baseline index. The return mode values represent the distance between the baseline index and the upper end and the lower end of the baseline. Algorithm 2 shows the procedure of detecting and localizing baseline for the Arabic text-line segment, where the algorithm takes a binary text-line image as an input and returns the upper and lower indexes of the baseline as an output.

Algorithm 2 Baseline detection and localization

1: **Input**: **TLIS** as Text-line image segment
2: $HP \leftarrow horizontalProjection(TLIS)$;
3: $HP \leftarrow smoothing(HP)$;
4: $BaselineIndex \leftarrow getMaximumPeakValue(HP)$;
5: $VPU \leftarrow verticalProjection(LTSI[:BaselineIndex])$;
6: $VPL \leftarrow verticalProjection(LTSI[BaselineIndex:])$;
7: $UperIndex \leftarrow mode(VPU)$;
8: $LowerIndex \leftarrow mode(VPL)$;
9: **OUTPUT**: **UperIndex, LowerIndex**

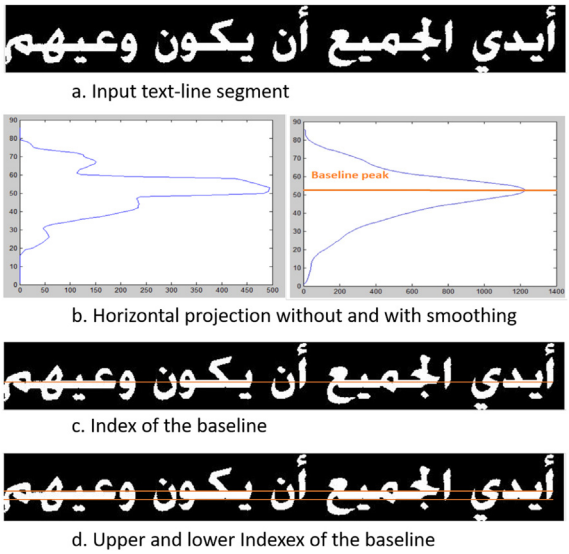

Fig. 4. Baseline detection and localization.

3.1.3. Characters and ligatures segmentation

In this stage, each text-line segment image is segmented into characters or ligatures. To handle the overlapping between characters and reducing the number of ligatures and hence reducing the number of classes, the segmentation is done in three steps using three segmentation methods namely segmentation by vertical projection, by connected components, and by baseline removal. Before starting the segmentation, we removed the dots and diacritics by applying the connected component algorithm and then delete all components that do not intersect with the baseline index. The existence of dots and diacritics hide some segmentation points and also increase the number of connected components. Fig. 5b shows the output after removing dots and diacritics from Fig. 5a.

In the first segmentation method, the text-line segment image is segmented using vertical projection, in which the cutting points (segmentation points) are identified when the $VP = 0$. The output of this step could be words, sub-words, characters, or ligatures as shown in Fig. 5c, where the red lines represent the segmentation points. The output words/sub-words of this stage could have overlapped and not connected characters as in the (الحمرا) sub-word. To segment these cases, we apply segmentation by connected components as a second segmentation method. Segmentation by connected components algorithm extracts the components (e.g. sub-words or characters) that are not connected from the input image, which aims at segmenting the characters that are not connected and they are vertically overlapped. The output of this step could be characters, ligatures, or word/sub-words that contain connected characters through baseline. Fig. 5d shows the output after applying a connected component to the (الحمرا) sub-word. Finally, the words/sub-words are segmented by first removing the baseline, and then the segmentation points are identified using vertical projection. The location of the segmentation points should be

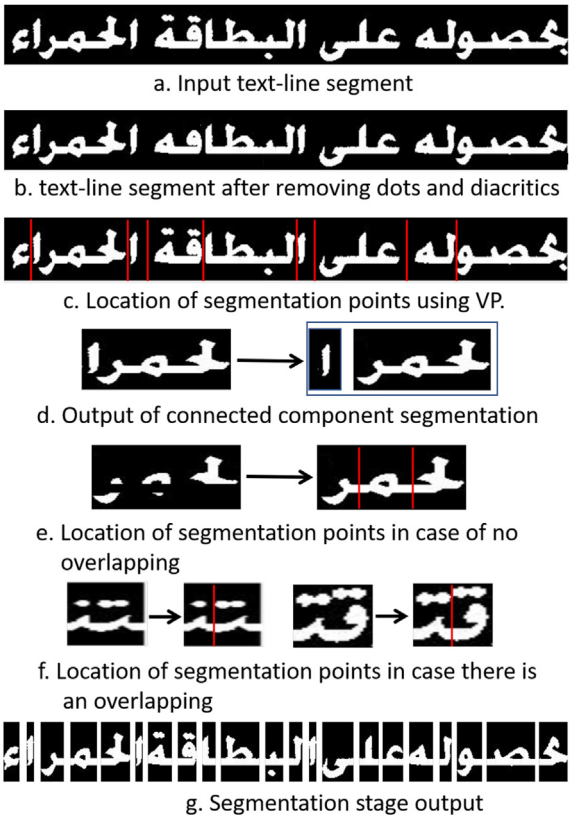

Fig. 5. Output of segmentation stage.

carefully determined to save the shape of the characters. Fig. 5e and f show the location of the segmentation points using baseline removal. The segmentation points are located at the middle of the separation region (area of consecutive zero projection values) if there is no overlapping as shown in Fig. 5e, or around the middle of the separation region (where VP equal zero or equal the minimum projection value) if there is an overlapping between two consecutive connected characters as shown in Fig. 5f. Fig. 5g shows the segmented characters as an output of the segmentation stage based on the identified segmentation points from the three segmentation steps. Algorithm 3 summarizes characters and ligatures segmentation stages.

Algorithm 3 Characters and ligatures segmentation

Input: **TLIS** as Text-line image segment
Input: **BUI** as baseline upper index
Input: **BLI** as baseline lower index
TLIS ← *removeDotsAndDiacritics*
VP ← *verticalProjection*(*TLIS*);
*listOfSegments*1 ← *cut TLIS where VP* == 0
while *each segment in listOfSegments*1
    *listOfSegments*2 ← *CC*(*segment*);
    while *each segment in listOfSegments*2
        segment ← *removeBaseline*(*segment*, *BUI*, *BLI*)
        VP ← *verticalProjection*(*segment*);
        segmentationPoints ← *index where VP* == 0
    end while
end while
OUTPUT: **Segmentation Points**

However, the output of the third segmentation step could cause under or over-segmentation problems for some characters. Therefore, to reduce the under-segmentation problems, which are due to the vertical overlapping between characters or between the characters and dots/diacritics as seen in Fig. 6a, in addition to the false touching between characters caused by scanning and binarization

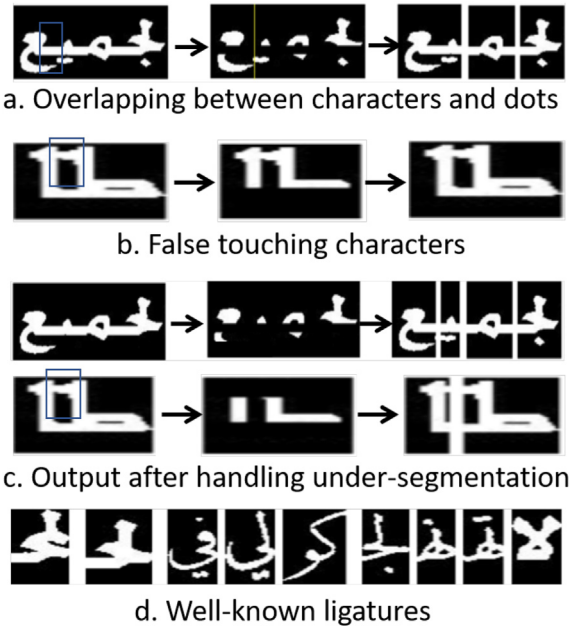

Fig. 6. Example of under-segmentation problems.

specifically for small font size as seen in Fig. 6b, we applied the vertical projection in the image without dots and diacritics. Besides, the vertical projection was computed on 60% of the height of the input word/sub-word (neglect the first 20% and the last 20% rows of the input word/sub-word). Fig. 6a and b show an example of the segmentation output before handling these under-segmentation problems. Fig. 6c shows the same example after handling these under-segmentation problems. The problem of under-segmentation cannot be solved completely because some Arabic fonts such as "Thulth" are written in a way that making some characters overlapped vertically and cannot be segmented. Therefore, other under-segmentation cases were treated as ligatures. Fig. 6d shows some well-known ligatures.

On the other hand, the segmentation process may segment some characters into more than one sub-characters which known as the over-segmentation problem. In the proposed approach, the Over-Segmentation problem mostly occurs in the following three cases: (i) the characters located at the end of the word such (ت، ت، ث، د), (ii) Seen character (سسسسسسسس), and (iii) Sheen Character (شششششش). Fig. 7a shows an example of over-segmentation on these cases.

To handle these cases, we first check if the current segment is a stroke. Indeed, the segment is a stroke if it meets the following characteristics: (i) it is a single connected component, (ii) the sum of horizontal projection above baseline is greater than the sum of horizontal projection below baseline, (iii) the height of the segment (end of the segment) is less than twice the second peak value of the horizontal projection, and (v) the segment has no Holes. Fig. 7b shows an example of stroke shapes. For the first case, if the segment is a stroke shape and it is the last segment, we ignore the segmentation point as shown in Fig. 7c. For the second case, if the current, the next, and/or the after next segment are strokes then merge these segments which represent SEEN character as shown in Fig. 7d. Finally, if the current segment and/or the after next segment are strokes without dots while the second segment is a stroke with dots, then merge these segments which represent SHEEN character as shown in Fig. 7e. Algorithm 4 summarizes over-segmentation rules.

Algorithm 4 Handling over-segmentation

Input: **Segment**
Input: **List of Segmentation Points LSP**
SET *i* ← 0
while *i is less than length of LSP* **do**
    if (*Segment is stroke && the last segment*) **then**
        remove segmentation point
        i ← *i* + 1
    else if
    (*current*, *next and/or after next segments are strokes*)
    then
        merge these segments
        i ← *i* + 3
    else if (*current and/or after next segments are strokes*
    without dots while second segment is stroke with dots)
    then
        merge these segments
        i ← *i* + 3
    else
        i ← *i* + 1
    end if
end while
OUTPUT: **Segmentation Points**

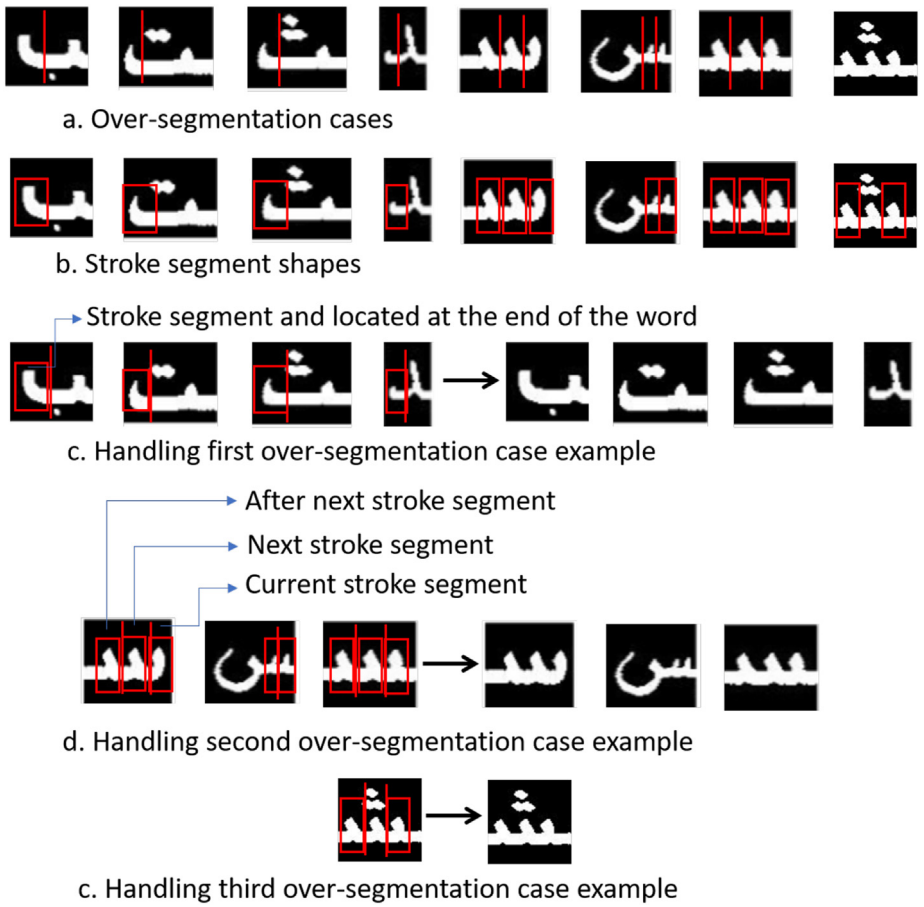

Fig. 7. Example of over-segmentation problems.

3.2. Features extraction and character recognition

Once the text-image (e.g. word/sub-word) is segmented into isolated segments (e.g ligatures, characters, and stroke), the next stage is feature extraction. For OCR systems, the feature extraction stage is correlated with other OCR stages, such as preprocessing and recognition stage, and treated as one of the most significant stages that affect the overall performance of the OCR (Printed arabic script recognition, xxxx). The selection of the feature extraction method in OCR remains the most important key factor for achieving high recognition accuracy. Therefore, several important aspects of feature extraction methods should be considered for higher recognition rate summarized in Hossain et al. (2012): (i) features need to be invariant with respect to character shape variations caused by the variability of the font types, styles, and sizes, (ii) represents the raw image data of the input characters by a reduced set of the most relevant information/features, (iii) simple and lightweight, (iv) need to be invariant to a certain degree of translation, rotation and shape distortions, (v) should be designed carefully with respect to the OCR preprocessing steps such as denoising, binarization and thinning, and (iv) in some cases the set of selected features must match the specification of the selected classifier.

Due to the nature of the Arabic text (e.g. existence of complex font types, different forms of the same character, overlapping between adjacent characters, the existence of ligatures, ...), designing hand-crafted features of mixed-font recognition approach, takes a long time to design and validate (Kumar and Bhatia, 2014). In addition, hand-crafted features require careful handling of the pre-processing methods used. Recent studies on deep learning (e.g. CNN and LSTM) have shown their success in automatic feature extraction and recognition tasks. They provide a very flexible and universal learnable framework for detecting and representing features. In addition, deep learning recognition based architectures (e.g. CNN) have the effective advantage of not require designing a hand-crafted feature vector. They are capable of automatic learning of the salient features directly from the training data (character image samples), which are noise resistance and invariant to a certain degree of shift, translation, and shape distortions of the input characters.

The proposed method for feature extraction and character recognition is based on using deep learning recognition-based architectures specifically using CNN architecture. CNN is a special class of multi-layer feed-forward neural networks that shows excellent recognition rates for digits and character recognition (type and handwritten). It is mainly focused on instance learning and thus can work better for isolated character recognition. CNN has the ability to learn variable, complex, non-linear mappings from a very large number of input images. It is composed of two basic parts of feature extraction layers, which consist of several convolution layers followed by pooling (e.g. max-polling) and an activation function, and a classification layers, which usually consist of fully connected layers.

In our work, we used LeNet-5 CNN architecture, which is a well-known CNN architecture designed for handwritten and type-written character recognition. Lecun et al. (1998) show that LeNet-5 delivers better accuracy for character recognition compared to other techniques and also does that at high speed.

LeNet-5 CNN architecture consists of 8 layers including one input layer, one output layer, two convolutional layers, and two sub-sampling layers for automatic feature extraction, two fully connected layers as multi-layer perceptron hidden layers for non-linear classification. Table 1 summarizes the architecture of the LeNet-5 in terms of layer type, feature maps, size, kernel size, stride, and activation function.

4. Experiments setup and results

In this section, the effectiveness of the proposed approach is evaluated using a set of conducted experiments under the APTID-MF dataset. We describe the experiments we made and discuss the results besides comparing them with results from other related works. APTID-MF is briefly described in Section 4.1. Results of the character segmentation stage and character recognition are reported in Sections 4.2.1 and 4.2.2 respectively. Finally, Section 4.2.3 compares our results with state-of-the-art related methods.

4.1. Dataset

Several datasets were developed for printed Arabic character recognition including APTI (Slimane et al., 2009), APTID-MF (Jaiem et al., 2013), KAFD (Luqman et al., 2014), DARPA (Davidson, 1997), Alph (Moussa et al., 2010) and PATDB (Al-Hashim and Mahmoud, 2010). They mainly differ in the form or text-level they offer (e.g. pages, text blocks, lines, and words), size of data (e.g. number of pages), font variations (e.g. types, sizes, and styles), source of data (e.g. newspapers, magazines, and books), way of preparing images (e.g. computer-generated or scanned), way of labeling (e.g. text files or XML files) and resolution. Table 2 provides a summary of these datasets in terms of the above differences.

Among these datasets, APTI, APTID-MF, and KAFD have large-scale open-vocabulary with sufficient variations in fonts types, sizes, and styles. However, APTID-MF and KAFD are more suitable to evaluate our approach since we assume that the input of the proposed approach is a text-line image. In addition, APTI images are generated at 72 dpi by an automatic program, which presents a disadvantage when evaluating the systems for recognizing scanned Arabic printed documents. To evaluate our approach we have selected the APTID-MF dataset since it has ground truth and is made using XML files which add more flexibility in the evaluation stage.

4.2. Results

4.2.1. Character segmentation results

This section presents the results of the conducted experiments using the APTID-MF dataset. In order to test the proposed segmentation method, the APTID-MF was preprocessed by segmenting each text block into its text lines using a horizontal projection profile approach. Therefore, for each font type, we selected randomly around 50 text blocks that cover the four font sizes and the two font styles. Table 3 shows the performance of the character segmentation stage in terms of character segmentation accuracy, which defined as the ratio of the total number of correctly segmented characters and ligatures to the total number of input characters and ligatures. The table shows that the algorithm achieves an average segmentation accuracy of 95% where the algorithm achieved the best accuracy under Advertising bold font with an accuracy of 98.7%, and achieved the worst accuracy under "Thulth" font type with an accuracy of 91.3% since the structure of this font produces may ligatures that consist of two or three characters due to the intra-overlapping problem.

The segmentation errors distributed between over-segmentation and under-segmentation problems. Therefore, most of the over-segmentation errors which are common among all fonts happened in SEEN and SHEEN (س، ش) characters, especially when using the small font size. Indeed, in small font sizes, the segmentation points may not be visible and thus cannot be detected because the distance between subparts is very small. On the other hand, most of the over-segmentation errors occur in "Thulth" and "Naskh" font types due to the overlapping between characters. Besides, things get worse in terms of under and over-segmentation with the existence of problems such as noise, false connected characters, and characters break due to the scanning and binarization.

4.2.2. Character recognition results

In this section, we describe the experimental results of the character recognition stage. We used Keras's implementation of the LeNet-5 CNN network. We trained the network using Google Colab with Nvidia Tesla K80 GPU. We have collected around 14,000 truly segmented characters/ligatures images distributed over 132 classes of characters and well-known ligatures with an average of 80 images per class to cover different variations of font types, sizes, and styles. We trained the network for 200 epochs using learning rates of 0.01 and batch size of 16. We partitioned the collected images into 60% training, 20% validation, and 20% testing sets.

Fig. 8 shows the training loss as a function of epoch for the training set images in our collected dataset. Besides, we recognized all images of the training set and all images of the test set after each epoch of training. The objective of doing this was to see how quickly the system was learning the characteristics of the characters. Figs. 9 and 10 show the results per epoch in term of classification accuracy. A high level of correct recognition performance was achieved after relatively few epochs for training and testing sets, with approximately 89% correct recognition achieved after about 35 epochs. This is consistent with the training loss in Fig. 11, which dropped quickly, then began a slow descent after about 35 epochs. Another 165 epochs of training were required

Table 1
Summary of LeNet-5 Architecture.

Layer	Layer Type	Feature Maps	Size	Kernel Size	Stride	Activation
Input	Image	1	32*32	–	–	–
1	Convelution	6	28*28	5*5	1	tanh
2	Average Pooling	6	14*14	2*5	2	tanh
3	Convelution	16	10*10	5*5	1	tanh
4	Average Pooling	16	5*5	2*2	2	tanh
5	Convelution	120	1*1	5*5	1	tanh
6	Fully Connected	–	84	–	–	tanh
Output	Fully Connected	–	10	–	–	softmax

Table 2
Summary of printed Arabic character recognition datasets.

Evaluation Criteria	APTI	APTID-MF	KAFD	P-KHAT	DARPA	ALPH	PATDB
Form or text level	Words	Text blocks	Pages/lines	lines	Pages	Paragraphs	Pages
Dataset size	45,313,600	1,845	28,767	9310	345	5000	6,954
Number of fonts types	10	10	40	8	4	14	8
Number of sizes	10	4	10	–	–	1	–
Number of styles	10	2	4	–	–	1	4
Preparing images	Synthesized	Scanned	Scanned	Scanned	Scanned	Scanned	Scanned
Ground truth	XML files	XML files	Text files	Text files	Text files	Text files	Text files
Resolution (dpi)	72	300	100, 200,300, 600	300	600	200	200, 300, 600

Table 3
Character segmentation results.

Font Type	Accuracy
Tahoma	94.4%
Advertising Bold	98.7%
Andalus	95.5%
Thulth	91.3%
Diwani	95%
M Unicode Sara	95.2%
Naskh	93%
Simplified Arabic	96.5%
Traditional Arabic	94.1%
Arabic Transparent	96%
Average Accuracy	**95%**

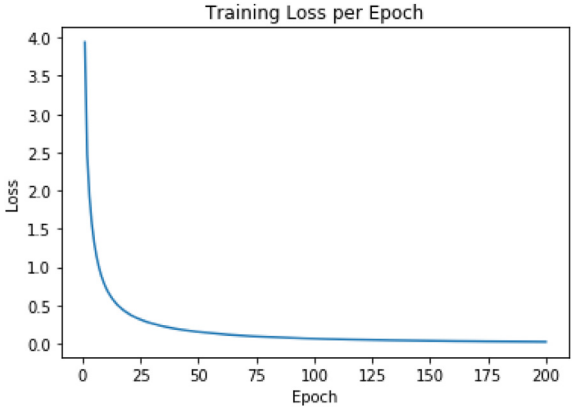

Fig. 8. Training loss as a function of epoch.

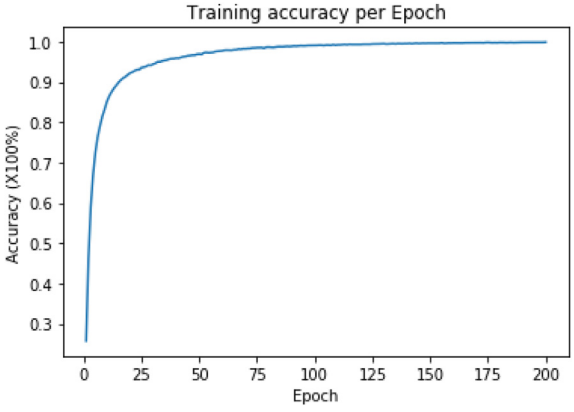

Fig. 9. Training accuracy as a function of epoch.

for the system to achieve recognition of about 92,2%. The most revealing feature of these two graphs is that CNN did equally as well on both sets of data. This is a good indication that the training

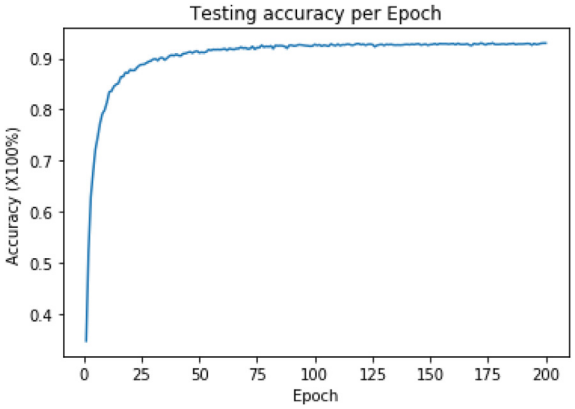

Fig. 10. Testing accuracy as a function of epoch.

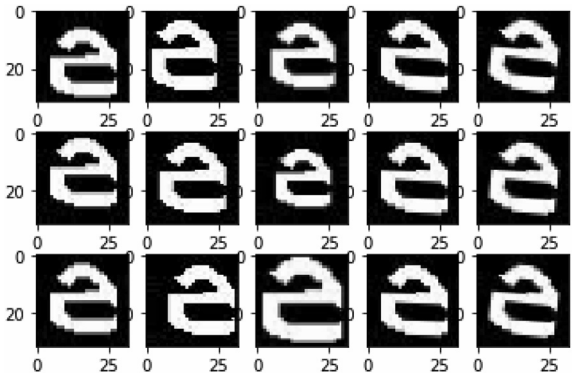

Fig. 11. An example of data augmentation.

was successful, and that it generalized well to digits it had not seen before. This is an example of the neural network not "over-fitting" the data in the training set.

CNN needs plenty of training data per class, so huge training data are required for classification with many classes. Therefore, to solve the problem of sparse training data, data augmentation methods have been developed, which increase the number of character image samples by applying suitable image affine transformations methods. Adding affine transformations to the original samples decreases the error rates and improves the generalization capability of the model by reducing over-fitting (Ciresan et al., 2011, 2012). However, this approach needs careful handling since the generated image samples by general image augmentation methods may lose important features and the samples may be correlated with each other.

Similar to Ciresan et al. (2011, 2012) we adopted 4 affine transformation methods: translation (horizontal and vertical), resizing, shearing and rotation. The horizontal and vertical translation

should add tolerance to segmentation variability. Rotation and shear transformation with small angles (10 for rotation and 5 for shearing) were also used, which are useful for the italic font (Counterclockwise rotation) and also for line skew (both Counterclockwise and Counterclockwise rotation). Besides, resizing (zoom in and out) transformations were applied to account for text size changes. The adopted data augmentation methods are implemented using Keras API, Fig. 11 shows an example of adopted transformation methods. The size of the dataset after applying the data augmentation method increased to 29,5218 images.

After applying data augmentations, we have trained the network for 200 epochs using learning rates of 0.01 and batch size of 128. We partitioned the collected and generated images into 80% training and 20% testing sets. Fig. 12 shows the training loss as a function of epoch. Figs. 13 and 14 show the training and testing results per epoch in term of classification accuracy. A high level of correct recognition performance was achieved after relatively few epochs for training and testing sets, with approximately 95% correct recognition achieved after about 20 epochs. This is consistent with the training loss in Fig. 11, which dropped quickly, then began a slow descent after about 20 epochs. Another 180 epochs of training were required for the system to achieve recognition of about 99.97%. Using data augmentations, the model is trained faster and significantly improves the accuracy from 92.2% to 99.9%. Finally, the overall performance of the proposed method which computed as the performance of the character segmentation stage times the performance of the classification stage (95% $\times$ 99.97%) is 95%.

According to the results obtained from the experiments made, we can draw the following highlights or conclusions: (i) the difficulties/challenges inherited by the cursive script are unavoidable making the segmentation stage a challenging task, (ii) the existence of small font sizes and complex font types increase the problem of character overlapping, which most of the current character segmentation algorithms cannot successfully handle this issue, (iii) the performance of the segmentation approach is highly dependent on the complexity of the font type, (iv) using deep learning as a feature extractor method have the ability to learn features that easily adapted to font variations, (iv) using data augmentation methods decreases the error rates and improve generalization capability of the model, and (vi) since the proposed approach is a segmentation-based, results show that the performance of the recognition stage is highly dependent on the output segmentation stage.

4.2.3. Comparison with other related works

Table 4 shows a subjective comparison between our proposed character segmentation approach against other related works presented in the literature in terms of the character segmentation type, test dataset, font variations (types, sizes, and styles), and segmentation accuracy. Although it is impossible to make complete objective comparisons, Table 4 shows that our proposed method outperforms other methods tested on scanned datasets with sufficient of font variations.

Table 5 provide a subjective comparison between the proposed character recognition approach against other related works presented in the literature in terms of the character segmentation method, dataset, font variations (types, sizes, and styles), adopted features (handcrafted/deep learning), Classifier, results in terms of character recognition accuracy, and the use of post-processing methods. It is worth mentioning that complete objective comparisons cant be made due to the following issues: (1) some of the authors have tested their work on their own collected data and didn't make it public, (2) the implementation code of the proposed methods in the literature is not available and sometimes it is difficult to write the code since it may depend on assumptions, parameters, hypothesis, tools, and APIs that are not mentioned clearly in the published paper, (3) the datasets are varied or diverse in terms of text-level, font variations, way of preparing images (scanned or synthesized) and resolution, (4) the adopted classifier and their parameters, (5) way of training (e.g. cross-validation) and data partitioning (e.g. percent of a train and test sets), (6) testing with mono-fonts or with mixed fonts (with and without font types recognition) and (7) the using of post-processing methods (e.g. n-grams) to improve accuracy. Nevertheless, the table shows our proposed method outperforms other related works designed for mixed-fonts without font type recognition stage and without using post-processing methods.

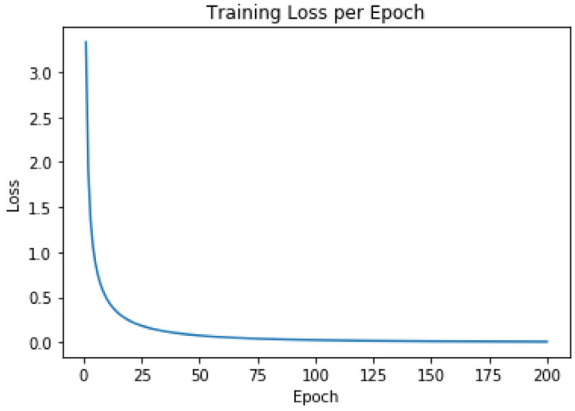

Fig. 12. Training loss as a function of epoch with data augmentations.

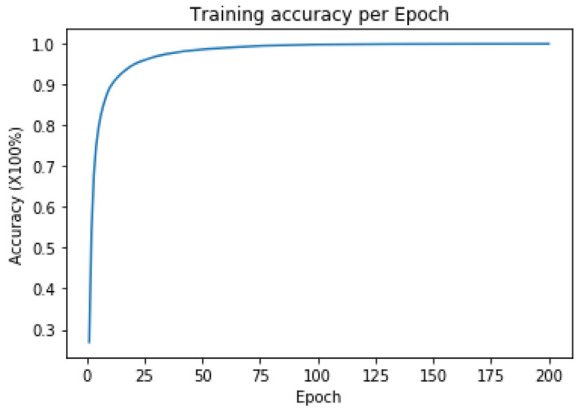

Fig. 13. Training accuracy as a function of epoch with data augmentations.

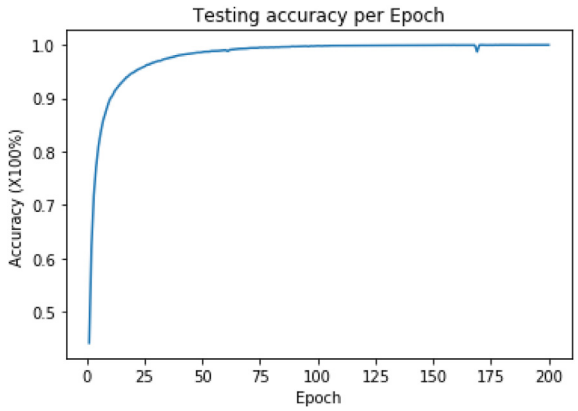

Fig. 14. testing accuracy as a function of epoch with data augmentations.

Table 4
Subjective comparison with other character segmentation methods presented in the literature.

Author	Segmentation Type	Data-set	Font Variations			Accuracy
			Type	Size	Style	
Zheng et al. (2004)	Vertical histogram and some structural characteristics rules	500 samples of Arabic text	Simplified Arabic and Arabic Transparent	12, 14, 16, 18, 20 and 22	Plain	94.8%
Javed et al. (2010)	Pattern matching techniques	1282 unique ligatures extracted from 5000 high frequency words	Noori Nastalique font.	36	Plain	92%
Saabni (2014)	Partial Segmentation and Hausdorff Distance	APTI	different fonts to cover different complexity of shapes	10 different sizes	Plain	96.8%
Anwar et al., 2015	Projection based	127 sentences composed of 1061 letters	Traditional Arabic	70	Plain	97.5%
Marwa Amara and Zidi (2016)	Histogram and contextual properties	APTI	Different font types	Different sizes	Plain, italic, and bold	85.6%
Radwan et al. (2016)	Multichannel Neural Networks	APTI	Arial, Tahoma, Thuluth, and Damas	18	Plain	95.5%
Qomariyah et al. (2017)	Interests Points, Contour based	10 Lines of 30 sub-word	–	–	Plain	86.5%
Firdaus et al. (2017)	Connected component	5 Lines 15 words	–	–	Plain	80.2%
Amara et al. (2017)	Projection profile, SVM	APTI	Advertising Bold	6,8,10, 12	Plain, italic, and bold	98.24%
Marwa Amara and Zidi (2016)	Contour based and Template matching	83 Lines of 984 Words	34 different fonts	Different font sizes	Plain	94.7
Mohammad et al. (2019)	Contour-based method	APTI	Advertising Bold, Simplified Arabic, Arial, Traditional Arabic, and Times New Roman	8, 9, 10,12, 14, 16, 18 and 24	Pain, Italic, and Bold	98.2%
Qaroush et al. (2019)	Projection profile method with statistical and topological features	APTI	Andalus, Transparent, Advertising Bold, Diwani, Thuluth, Simplified, Tahoma, Traditional, Naskh, and M Unicode Sara	10, 12, 14, 16, 18, and 24	Pain, Italic, Bold	97.5%
Proposed method	Hybrid (Projection profile, connected components, and baseline removal)	APTID-MF	Andalus, Transparent, Advertising Bold, Diwani, Thuluth, Simplified, Tahoma, Traditional, Naskh, and M Unicode Sara	12, 14, 16, 18	Plain, bold	95%

Table 5
Subjective comparison with other character recognition methods presented in the literature.

Author	Seg. Type	Features	Dataset	Font Variations			Classifier	Accuracy
				Type	Size	Style		
Supriana and Nasution (2013)	Explicit	Hand-crafted	Custom: 37 scanned pages	Arial, Arial Unicode MS, Microsoft Sans Serif, Segoe UI, Tahoma, Traditional Arabic	–	–	DT (C4.5)	82% for mixed fonts
Radwan et al. (2016) and Radwan et al. (2018)	Implicit	CNN	APTI	Arabic Transparent	18	Plain	Neural Network	94.38%
Ahmad et al. (2016)	Implicit	Hand-crafted	APTI	Andalus, Arabic Transparent, Diwani Letter, Simplified Arabic, Traditional Arabic	24	Plain	HMM	97.93% for mono-font, 92.29% for mixed-fonts without font-type recognition, 97,08% for mixed-fonts with font-type recognition
Ahmad et al. (2016)	Implicit	Hand-crafted	P-KHAT	8 fonts	–	Plain	HMM	97.11% for mono-font, 87.81% for mixed-fonts without font-type recognition, 96,56% for mixed-fonts with font-type recognition
Khoury et al. (2015)	Implicit	Hand-crafted	APTI	Andalus, Arabic Transparent, Diwani Letter, Simplified Arabic, Traditional Arabic	6, 8, 10, 12, 18, 24	Plain	HMM	96.5% for mono-font with 5-gram language models
Krayem et al. (2013)	Holistic	Hand-crafted	Custom: (252 words)	Andalus, Simplified Arabic, Tahoma, Thuluth, Traditional Arabic	14	Plain	HMM	67.10% for multi-font model
Al-Muhtaseb et al. (2008)	Implicit	Hand-crafted	Custom: 2766 lines	Arial, Tahoma, Akhbar, Thuluth, Naskh,Simplified Arabic, Andalus, Traditional Arabic	–	–	HMM	99.21% for mono-fonts

Table 5 (*continued*)

Author	Seg. Type	Features	Dataset	Font Variations			Classifier	Accuracy
				Type	Size	Style		
THOCR2 (Slimane et al., 2013)	Implicit	Hand-crafted	APTI	10 fonts in APTI	6, 8, 10, 12, 18, 24	Plain	HMM	99% for mono-font mono-size, 90.65% for multi-fonts with 4-gram language model
Awaida and Khorsheed (2012)	Implicit	Hand-crafted	APTI	10 fonts in APTI	10 font sizes	–	HMM	Top: 96.65% for Mono-font
Khorsheed (2015)	Holistic	Hand-crafted	Custom: 15000 text-line	Tahoma,Simplified Arabic, Traditional Arabic, Andalus, Naskh,Thuluth	–	–	HMM	96.1% for Mono fonts
Khorsheed (2007)	Holistic	Hand-crafted	Custom: 15000 text-line	Tahoma,Simplified Arabic, Traditional Arabic, Andalus, Naskh,Thuluth	–	–	HMM	89.7% for Mono fonts
Bazzi et al. (1999)	Implicit	Hand-crafted	DARPA	Geeza, Baghdad, Kufi, Nadim	–	–	HMM	95.50% for mixed fonts with trigram language model
SID, UPV-BHMM, THOCR1, THOCR2 (Slimane et al., 2013)	Implicit	Hand-crafted	APTI	Arabic Transparent	6, 8, 10, 12, 18, 24	Plain	HMM	SID: 99.75%,UPV-BHMM: 99.88%,THOCR1: 98.09%,THOCR2: 98.58%
SID, UPV-BHMM, THOCR1, THOCR2 (Slimane et al., 2013)	Implicit	Hand-crafted	APTI	DecoType Naskh	6, 8, 10, 12, 18, 24	Plain	HMM	SID: 97.14%, UPV-BHMM: 99.88%,THOCR1: 98.09%,THOCR2: 98.58%
Proposed method	Explicit	CNN	APTID-MF	Andalus, Transparent, Advertising Bold, Diwani, Thuluth, Simplified, Tahoma, Traditional, Naskh, and M Unicode Sara	12, 14, 16, 18	Plain, bold	LeNet-5 CNN	99.97% for the classification stage tested on correctly segmented characters and 95% for overall approach for mixed-fonts without font recognition and without any post processing methods.

5. Conclusion

Character segmentation and recognition is still an active research area especially for the cursive script and for omnifont recognition scenario. In addition, due to font variations (e.g. type, style, and size), the existence of complex font types, and the existence of the overlapping between characters, character segmentation is regarded as one of the most critical stages in developing OCR systems. This paper presents a segmentation-based, omnifont, open-vocabulary Arabic printed text recognition approach. The presented approach uses an explicit, indirect, and hybrid character segmentation method that segments a text-line image into isolated characters or ligatures. The segmentation stage performed in three steps that are designed carefully to be font independent and to handle the existence of overlapping between characters. In addition, the presented segmentation method is parameter-free and performs quite well when dealing with over-segmentation and under-segmentation problems. The proposed approach uses LeNet-5 convolutional network for feature extraction and character recognition. We evaluated the presented approach using the publicly available APTID-MF dataset of printed Arabic scanned texts in multiple fonts. The whole approach achieves an average accuracy of 95% without using font-type recognition or any post-processing techniques. The achieved results are comparatively much better than those that have been reported in terms of the character error rate. Besides, we presented a comprehensive comparison of the proposed work with similar state-of-the-art studies of printed Arabic text segmentation and recognition.

As future work, we plan to move in three directions: First, use the font type recognition stage to enhance and generalized the segmentation stage. Second, replace the third segmentation step by using learning-based segmentation such as using attention-based methods. Third, improving error rates by using post-processing techniques, such as spell checkers, word lexicons, and n-gram language models.

Declaration of Competing Interest

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

Acknowledgement

We immensely grateful to Dr. Irfan Ahmad “assistant professor, information and computer science department, KFUPM” for his comments and suggestions that greatly improved the manuscript.

References

Abualigah, L., 2018. Feature Selection and Enhanced Krill Herd Algorithm for Text Document Clustering. https://doi.org/10.1007/978-3-030-10674-4.

Abualigah, L., Khader, A.T., Hanandeh, E., 2017. A new feature selection method to improve the document clustering using particle swarm optimization algorithm. Journal of Computational Science. https://doi.org/10.1016/j.jocs.2017.07.018.

Abualigah, L., Khader, A.T., Hanandeh, E., 2018. Hybrid clustering analysis using improved krill herd algorithm. Applied Intelligence. https://doi.org/10.1007/s10489-018-1190-6.

Ahmad, J., 2007. Optical character recognition system for arabic text using cursive multi-directional approach. Journal of Computer Science 3, 549–555.

Ahmad, I., Mahmoud, S.A., Fink, G.A., 2016. Open-vocabulary recognition of machine-printed arabic text using hidden markov models. Pattern Recognition 51, 97–111. https://doi.org/10.1016/j.patcog.2015.09.011. URL: http://www.sciencedirect.com/science/article/pii/S0031320315003428.

Ahmed, P., Al-Ohali, Y., 2000. Arabic character recognition: Progress and challenges. Journal of King Saud University-Computer and Information Sciences 12, 85–116.

Alginahi, Y.M., 2013. A survey on arabic character segmentation. International Journal on Document Analysis and Recognition (IJDAR) 16 (2), 105–126.
Al-Hashim, A.G., Mahmoud, S.A., 2010. Benchmark database and gui environment for printed arabic text recognition research. WSEAS Transactions on. Information Science and Applications 7 (4), 587–597.
Al-Muhtaseb, H., Mahmoud, S., Qahwaji, R., 2008. Recognition of off-line printed arabic text using hidden markov models 88, 2902–2912.
Altuwaijri, M.M., Bayoumi, M.A., 1998. A thinning algorithm for arabic characters using art2 neural network. IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing 45 (2), 260–264.
Amara, M., Zidi, K., Ghedira, K., 2017. An efficient and flexible knowledge-based arabic text segmentation approach. IJCSIS 15 (7).
Anwar, K., Adiwijaya, Nugroho, H., 2015. A segmentation scheme of arabic words with harakat. In: 2015 IEEE International Conference on Communication, Networks and Satellite (COMNESTAT), pp. 111–114.
Awaida, S.M., Khorsheed, M.S., 2012. Developing discrete density hidden markov models for arabic printed text recognition. In: 2012 IEEE International Conference on Computational Intelligence and Cybernetics (CyberneticsCom), pp. 35–39. https://doi.org/10.1109/CyberneticsCom.2012.6381612.
Bazzi, I., Schwartz, R., Makhoul, J., 1999. An omnifont open-vocabulary ocr system for english and arabic. IEEE Transactions on Pattern Analysis and Machine Intelligence 21 (6), 495–504. https://doi.org/10.1109/34.771314.
Bushofa, B., Spann, M., 1997. Segmentation and recognition of arabic characters by structural classification. Image and Vision Computing 15 (3), 167–179.
Bushofa, B.M.F., Spann, M., 1997. Segmentation of arabic characters using their contour information. In: Proceedings of 13th International Conference on Digital Signal Processing, vol. 2, pp. 683–686.
Casey, R.G., Lecolinet, E., 1996. A survey of methods and strategies in character segmentation. EEE Transactions on Pattern Analysis and Machine Intelligence 18 (7), 690–706. https://doi.org/10.1109/34.506792. URL:https://doi.org/10.1109/34.506792.
G., Casey, R., Lecolinet, E., 1996. A survey of methods and strategies in character segmentation. Pattern Analysis and Machine Intelligence, IEEE Transactions on 18, 690–706. doi: 10.1109/34.506792.
Choudhary, A., 2014. A review of various character segmentation techniques for cursive handwritten words recognition.
Cilia, N.D., De Stefano, C., Fontanella, F., Scotto di Freca, A., 2019. A ranking-based feature selection approach for handwritten character recognition. Pattern Recognition Letters 121, 77–86. doi: 10.1016/j.patrec.2018.04.007. URL:http://www.sciencedirect.com/science/article/pii/S0167865518301272.
Graphonomics for e-citizens: e-health, e-society, e-education.
Ciresan, D.C., Meier, U., Masci, J., Gambardella, L.M., Schmidhuber, J., 2011. High-performance neural networks for visual object classification. CoRR abs/1102.0183. URL:http://arxiv.org/abs/1102.0183.
Ciresan, D.C., Meier, U., Schmidhuber, J., 2012. Multi-column deep neural networks for image classification. CoRR abs/1202.2745. URL:http://arxiv.org/abs/1202.2745.
Cowell J., H.F., 2001. Thinning arabic characters for feature extraction. In: Proceeding of SPIE. Document Recognition III, pp. 181–185.
Davidson R., H.R., 1997. Arabic and persian ocr training and test data sets. In: Proceedings of Symposium. On Document Image Understanding Technology.
Firdaus, F.I., Khumaini, A., Utaminingrum, F., 2017. Arabic letter segmentation using modified connected component labeling. In: 2017 International Conference on Sustainable Information Engineering and Technology (SIET), pp. 392–397.
Gonzalez, R.C., Woods, R.E., 2017. Digital Image Processing. Prentice Hall.
Gouda, A.M., Rashwan, M.A., 2004. Segmentation of connected arabic characters using hidden markov models. 2004 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications, 2004. CIMSA. pp. 115–119.
Hamad, K., Kaya, M., 2016. A detailed analysis of optical character recognition technology. International Journal of Applied Mathematics, Electronics and Computers 4, 244. https://doi.org/10.18100/ijamec.270374. 244.
Hossain, M.Z., Amin, M.A., Yan, H., 2012. Rapid feature extraction for optical character recognition. CoRR abs/1206.0238. URL:http://arxiv.org/abs/1206.0238.
Inkeaw, P., Bootkrajang, J., Charoenkwan, P., Marukatat, S., Ho, S.Y., Chaijaruwanich, J., 2018. Recognition-based character segmentation for multi-level writing style. International Journal on Document Analysis and Recognition (IJDAR) 21 (1), 21–39.
Islam, N., Islam, Z., Noor, N., 2017. A survey on optical character recognition system. CoRR abs/1710.05703.
Jaiem, F.K., Kanoun, S., Khemakhem, M., El Abed, H., Kardoun, J., 2013. Database for arabic printed text recognition research. In: Petrosino, A. (Ed.), Image Analysis and Processing – ICIAP 2013. Springer, Berlin Heidelberg, Berlin, Heidelberg, pp. 251–259.
Javed, S.T., Hussain, S., Maqbool, A., Asloob, S., Jamil, S., Moin, H., 2010. Segmentation free nastalique urdu ocr. World Academy of Science, Engineering and Technology 46, 456–461.
Kesiman, M.W.A., Prum, S., Burie, J., Ogier, J., 2016. Study on feature extraction methods for character recognition of balinese script on palm leaf manuscript images. In: 2016 23rd International Conference on Pattern Recognition (ICPR), pp. 4017–4022.
Khorsheed, M., 2007. Offline recognition of omnifont arabic text using the hmm toolkit (htk). Pattern Recognition Letters 28 (12), 1563–1571. https://doi.org/10.1016/j.patrec.2007.03.014. URL:http://www.sciencedirect.com/science/article/pii/S0167865507001110.
Khorsheed, M., 2015. Recognizing cursive typewritten text using segmentation-free system. The Scientific World Journal 2015. https://doi.org/10.1155/2015/818432.
Khoury, I., Giménez, A., Juan, A., Andrés-Ferrer, J., 2015. Window repositioning for printed arabic recognition. Pattern Recognition Letters 51, 86–93. https://doi.org/10.1016/j.patrec.2014.08.009. URL:http://www.sciencedirect.com/science/article/pii/S0167865514002682.
Krayem, A., Sherkat, N., Evett, L., Osman, T., 2013. Holistic arabic whole word recognition using hmm and block-based dct. In: 2013 12th International Conference on Document Analysis and Recognition, pp. 1120–1124. https://doi.org/10.1109/ICDAR.2013.227.
Kumar, G., Bhatia, P.K., 2014. A detailed review of feature extraction in image processing systems. In: 2014 Fourth International Conference on Advanced Computing Communication Technologies, pp. 5–12. https://doi.org/10.1109/ACCT.2014.74.
Lawgali, A., 2015. A survey on arabic character recognition. International Journal of Signal Processing, Image Processing and Pattern Recognition 8 (2), 401–426.
Lecun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE 86 (11), 2278–2324.
Lokhande, S., Dawande, N., 2015. A survey on document image binarization techniques, pp. 742–746. doi: 10.1109/ICCUBEA.2015.148.
Lorigo, L.M., Govindaraju, V., 2006. Offline arabic handwriting recognition: a survey. IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (5), 712–724.
Luqman, H., Mahmoud, S., Awaida, S., 2014. Kafd arabic font database. Pattern Recognition 47, 2231–2240. https://doi.org/10.1016/j.patcog.2013.12.012.
Mahmood, A., 2013. Arabic & urdu text segmentation challenges & techniques. InternatIonal Journal of Computer Science and Technology 4, 32–34.
Mahmoud A.A. Mousa Mohammed S. Sayed, M.I.A., 2017. Arabic character segmentation using projection based approach with profile's amplitude filter. arXiv:1707.00800.
Marwa Amara, K., Zidi, K.G.S.Z., 2016. New rules to enhance the performances of histogram projection for segmenting small-sized arabic words. In: International Conference on Hybrid Intelligent Systems.
Mehran, R., Pirsiavash, H., Razzazi, F., 2005. A front-end ocr for omni-font persian/arabic cursive printed documents. Digital Image Computing: Techniques and Applications DICTA'05, 56, pp. 56–56.
Mohammad, K., Qaroush, A., Ayesh, M., Washha, M., Alsadeh, A., Agaian, S., 2019. Contour-based character segmentation for printed Arabic text with diacritics. Journal of Electronic Imaging 28,. https://doi.org/10.1117/1.JEI.28.4.043030 043030.
Mostafa, M., 2004. An adaptive algorithm for the automatic segmentation of printed arabic text. In: 17th National Computer Conference, Saudi Arabia. International Society for Optics and Photonics, pp. 437–444.
Motawa, D., Amin, A., Sabourin, R., 1997. Segmentation of arabic cursive script. icdar 97, 625–628.
Moussa, S.B., Zahour, A., Benabdelhafid, A., Alimi, A.M., 2010. New features using fractal multi-dimensions for generalized arabic font recognition. Pattern Recognition Letters 31 (5), 361–371. https://doi.org/10.1016/j.patrec.2009.10.015. URL:http://www.sciencedirect.com/science/article/pii/S0167865509002906.
Nashwan, F., Rashwan, M., Al-Barhamtoshy, H., Abdou, S., Moussa, A., 2017. A holistic technique for an arabic ocr system. Journal of Imaging 4 (1), 6.
Nawaz, S.N., Sarfraz, M., Zidouri, A., Al-Khatib, W.G., 2003. An approach to offline arabic character recognition using neural networks. In: Electronics, Circuits and Systems, 2003. ICECS 2003. Proceedings of the 2003 10th IEEE International Conference on, vol. 3, IEEE, pp. 1328–1331.
Naz, S., Umar, A., Ahmad, R., Razzak, M., Rashid, S.F., Shafait, F., 2016a. Urdu nasta'liq text recognition using implicit segmentation based on multi-dimensional long short term memory neural networks 5.
Naz, S., Umar, A.I., Shirazi, S.H., Ahmed, S.B., Razzak, M.I., Siddiqi, I., 2016b. Segmentation techniques for recognition of arabic-like scripts: A comprehensive survey. Education and Information Technologies 21 (5), 1225–1241.
Noushin Najafiragheb, A.H., Harifi, A., 2016. A survey of feature extraction techniques in ocr. In: International Conference on New Research Achievements in Electrical and Computer Engineering.
Omidyeganeh, M., Nayebi, K., Azmi, R., Javadtalab, A., 2005. A new segmentation technique for multi font farsi/arabic texts. In: Proceedings. (ICASSP'05). IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 2. IEEE.
Perwej, D.Y., Abdul Hannan, S., Asif, A., Mane, A., 2014. An overview and applications of optical character recognition. International Journal of Advance Research In Science And Engineering (IJARSE) 3, 261–274.
Printed arabic script recognition: A survey. International Journal of Advanced Computer Science and Applications.
Qaroush, A., Jaber, B., Mohammad, K., Washaha, M., Maali, E., Nayef, N., 2019. An efficient, font independent word and character segmentation algorithm for printed arabic text. Journal of King Saud University – Computer and Information Sciences. https://doi.org/10.1016/j.jksuci.2019.08.013. URL:http://www.sciencedirect.com/science/article/pii/S1319157819308158.
Qomariyah, F.U., Mahmudy, W.F., 2017. The segmentation of printed arabic characters based on interest point. Journal of Telecommunication, Electronic and Computer Engineering 9, 19–24.
Qomariyah, F., Utaminingrum, F., Mahmudy, W.F., 2017. The segmentation of printed arabic characters based on interest point. Journal of

Telecommunication, Electronic and Computer Engineering (JTEC) 9 (2–8), 19–24.
Radwan, M.A., Khalil, M.I., Abbas, H.M., 2016. Predictive segmentation using multichannel neural networks in arabic ocr system. In: IAPR Workshop on Artificial Neural Networks in Pattern Recognition. Springer, pp. 233–245.
Radwan, M.A., Khalil, M.I., Abbas, H.M., 2018. Neural networks pipeline for offline machine printed arabic ocr. Neural Processing Letters 48 (2), 769–787. https://doi.org/10.1007/s11063-017-9727-y. URL:https://doi.org/10.1007/s11063-017-9727-y.
Rehman, A., Mohamad, D., Sulong, G., 2009. Implicit vs explicit based script segmentation and recognition: A performance comparison on benchmark database. International Journal of Open Problems in Computer Science and Mathematics 2.
Romeo-Pakker, K., Miled, H., Lecourtier, Y., 1995. A new approach for latin/arabic character segmentation. In: Document Analysis and Recognition, 1995, Proceedings of the Third International Conference on, vol. 2, IEEE, pp. 874–877.
Rosenberg, A., 2012. Using sift descriptors for ocr of printed arabic.
Saabni, R., 2014. Efficient recognition of machine printed arabic text using partial segmentation and hausdorff distance. In: 2014 6th International Conference of Soft Computing and Pattern Recognition (SoCPaR), pp. 284–289.
Sari, T., Souici, L., Sellami, M., 2002. Off-line handwritten arabic character segmentation algorithm: Acsa. In: Frontiers in Handwriting Recognition, 2002. Proceedings. Eighth International Workshop on, IEEE, pp. 452–457.
Shaikh, N.A., Mallah, G.A., Shaikh, Z.A., 2009. Character segmentation of sindhi, an arabic style scripting language, using height profile vector. Australian Journal of Basic and Applied Sciences 3 (4), 4160–4169.
Slimane, F., Ingold, R., Kanoun, S., Alimi, A.M., Hennebert, J., 2009. A new arabic printed text image database and evaluation protocols. In: 2009 10th International Conference on Document Analysis and Recognition. IEEE, pp. 946–950.
Slimane, F., Kanoun, S., El Abed, H., Alimi, A., Ingold, R., Hennebert, J., 2013. Icdar 2013 competition on multi-font and multi-size digitally represented arabic text, pp. 1433–1437. doi: 10.1109/ICDAR.2013.289.
Soora, N.R., Deshpande, P.S., 2018. Review of feature extraction techniques for character recognition. IETE Journal of Research 64 (2), 280–295. https://doi.org/10.1080/03772063.2017.1351323.
Supriana, I., Nasution, A., 2013. Arabic character recognition system development. Procedia Technology 11, 334–341. doi: 10.1016/j.protcy.2013.12.199. URL: http://www.sciencedirect.com/science/article/pii/S2212017313003538. 4th International Conference on Electrical Engineering and Informatics, ICEEI 2013.
Timsari B., F.H., 1996. Morphological approach to character recognition in machine-printed persian words. In: Proceeding of SPIE. Document Recognition III.
Trier, Øivind Due, Jain, A.K., Taxt, T., 1996. Feature extraction methods for character recognition-a survey. Pattern Recognition 29 (4), 641–662. https://doi.org/10.1016/0031-3203(95)00118-2. URL:http://www.sciencedirect.com/science/article/pii/0031320395001182.
Ud Din, I., Siddiqi, I., Khalid, S., Azam, T., 2017. Segmentation-free optical character recognition for printed urdu text. EURASIP Journal on Image and Video Processing 2017 (1), 62. https://doi.org/10.1186/s13640-017-0208-z. URL: https://doi.org/10.1186/s13640-017-0208-z.
Vala, H.J., Baxi, A., 2013. A review on otsu image segmentation algorithm. International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) 2 (2), 387–389.
Zeki, A.M., Zakaria, M.S., 2004. Challenges in recognizing arabic characters. International Islamic University Malaysia (IIUM), Kuala Lumpur, Malaysia, National University of Malaysia (UKM), Bangi, Selangor, Malaysia.
Zeki, A.M., Zakaria, M.S., Liong, C.Y., 2011. Segmentation of arabic characters: A comprehensive survey. International Journal of Technology Diffusion 2 (4), 48–82.
Zhang, Y., Zha, Z.Q., Bai, L.F., 2013. A license plate character segmentation method based on character contour and template matching. In: Applied Mechanics and Materials, vol. 333, pp. 974–979. Trans Tech Publ.
Zheng, L., Hassin, A.H., Tang, X., 2004. A new algorithm for machine printed arabic character segmentation. Pattern Recognition Letters 25 (15), 1723–1729.
Zidouri, A., Sarfraz, M., Shahab, S., Jafri, S., 2005. Adaptive dissection based subword segmentation of printed arabic text. Ninth International Conference on Information Visualisation, IV'05. IEEE, pp. 239–243.
Zoizou, A., Zarghili, A., Chaker, I., 2018. A new hybrid method for arabic multi-font text segmentation, and a reference corpus construction. Journal of King Saud University – Computer and Information Sciences. https://doi.org/10.1016/j.jksuci.2018.07.003. URL:http://www.sciencedirect.com/science/article/pii/S1319157818301769.