Pattern Recognition 43 (2010) 2773–2798

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Development of an efficient neural-based segmentation technique for Arabic handwriting recognition

Husam A. Al Hamad [a,*], Raed Abu Zitar [b]

[a] College of Computer, Qassim University, Qassim, Saudi Arabia
[b] School of Engineering and Computing Sciences, New York Institute of Technology, Amman, Jordan

ARTICLE INFO

Article history:
Received 15 June 2009
Received in revised form
27 February 2010
Accepted 7 March 2010

Keywords:
Segmentation
Handwritten
Arabic

ABSTRACT

Off-line Arabic handwriting recognition and segmentation has been a popular field of research for many years. It still remains an open problem. The challenging nature of handwriting recognition and segmentation has attracted the attention of researchers from industry and academic circles. Recognition and segmentation of Arabic handwritten script is a difficult task because the Arabic handwritten characters are naturally both cursive and unconstrained. The analysis of Arabic script is more complicated in comparison with English script. It is believed, good segmentation is one reason for high accuracy character recognition. This paper proposes and investigates four main segmentation techniques. First, a new feature-based Arabic heuristic segmentation AHS technique is proposed for the purpose of partitioning Arabic handwritten words into primitives (over-segmentations) that may then be processed further to provide the best segmentation. Second, a new feature extraction technique (modified direction features—MDF) with modifications in accordant with the characteristics of Arabic scripts is also investigated for the purpose of segmented character classification. Third, a novel neural-based technique for validating prospective segmentation points of Arabic handwriting is proposed and investigated based on direction features. In particular, the vital process of handwriting segmentation is examined in great detail. The classifier chosen for segmentation point validation is a feed-forward neural network trained with the back-propagation algorithm. Many experiments were performed, and their elapsed CPU times and accuracies were reported. Fourth, new fusion equations are proposed and investigation to examine and evaluate a prospective segmentation points by obtaining a fused value from three neural confidence values obtained from right and center character recognition outputs in addition to the segmentation point validation (SPV) output. Confidence values are assigned to each segmentation point located through feature detection. All techniques components are tested on a local benchmark database. High segmentation accuracy is reported in this research along with comparable results for character recognition and segmentation.

1. Introduction

The term ‘‘handwriting’’ is defined as meaning of artificial graphic marks containing some message through the mark's relation to language [1]. Only people can understand and recognize the handwritings of the others. The concept of handwriting has existed for old time, for the purpose of expanding people's memory and facilitating communication together. Much of culture may be attributed to the advent of handwriting. Many researchers began to focus their attention on attempting to simulate intelligent behavior. One such example was the attempt to imitate the human ability to read and recognize printed and handwritten characters [2]. Arabic civilization has a wide culture as many civilizations have. Arabic is spoken by millions of peoples [3] and it is an important part in the culture of many other civilizations. A lot of researches have been published in the area of handwritten Arabic script segmentation and recognition [4], until now outstanding results were not reached, one of the reasons is that it is considerably harder than that of Latin script [5].

1.1. Historical background

Earlier surveys discussed recognition and segmentation of both machine-print and handwriting, with much more discussion on machine-print. In 1980, Nouh et al. suggested a standard Arabic character set to facilitate computer processing [6]. Parhami and Taraghi [7] presented a new approach to Arabic recognition system in 1981. In that approach, sub-words were segmented and

* Corresponding author.
E-mail addresses: hhamad@qu.edu.sa, hushamad@yahoo.com (H.A. Al Hamad), rzitar@nyit.edu (R. Abu Zitar).

recognized according to features such as concavities, loops and connectivity. Increasing the tolerance to font variations, in 1986, Amin and Masini proposed a system for segmentation and recognition that used horizontal and vertical projections and shape-based primitives [8]. On 100 multi-font words, it achieved a character recognition rate of 85% and a word recognition rate of 95%. In a 1988, the recognition system introduced by El-Sheikh and Guindi used segmentation points that were based on minimal heights of word contours. The character classification used Fourier descriptors [9]. In 1990, Sami El-Dabi et al. used segmented characters based on invariant moments only after they were recognized. Recognition was attempted on regions of increasing width until a match was found [10]. In 1996, Yamin and Aoki presented a two-step segmentation system which used vertical projection onto a horizontal line followed by feature extraction and measurements of character width [11]. In 1998, Al-Badr and Haralick presented a holistic recognition system based on shape primitives that were detected with mathematical morphology operations [12]. In 1997, Alherbish et al. presented a parallel recognition algorithm which achieved a speed-up of 5.34 times [13]. In 1999, Khorsheed and Clocksin used features from a word's skeleton for recognition without prior segmentation [14]. In 2002, Hamami and Berkani developed a structural approach to handle many fonts, and it included rules to prevent over-segmentation [15]. Al-Qahtani and Khorsheed presented a system based on the portable hidden Markov model toolkit [16]. In 2007, Srihari and Ball, applied heuristic techniques for image processing representation of the binary image counter and removal of noise and dots [17].

1.2. Types of handwriting input: off-line versus on-line

A distinction must be cleared between the two very different forms of handwriting recognition: off-line and on-line. Whenever the term handwriting recognition is mentioned without prefixing it with the above terms, people usually think of handwriting entry using any recognition devices or techniques such as personal digital assistants (PDAs). It is therefore always necessary to qualify the form of handwriting recognition that is being performed. A very important reason for qualification has to do with the performance of each method. Off-line handwriting recognition refers to the process of recognizing words that have been scanned from a paper or book and are stored digitally in grey or binary scale format. After being stored in the above-mentioned format, it is conventional to perform further processing to allow superior recognition.

The main approaches that exist for off-line Arabic handwriting recognition may be divided into segmentation-based and holistic ones. In general, the former approach uses a strategy based on the recognition of individual characters or patterns, whereas non-segmentation-based deals with the recognition of the word image as a whole [18]. The objective is to over-segment the word sufficient number of times to ensure that all appropriate word boundaries have been dissected [19–21]. To determine the best segmentations, many research works studied merging segments of the word image and invoking a classifier to score the combinations. Most techniques employ an optimization algorithm making use of some sort of lexicon-driven and dynamic programming techniques [22].

1.3. Arabic scripts segmentation problems

Many research efforts have been published in the area of segmentation and recognition of handwritten Arabic script [4,5], until now these efforts have not reached good results because there is no enough researches in this topic and it is harder than that of Latin script due to the following reasons: (i) Arabic words are overlapped and written always cursively, i.e., more than one character can be written connected to each other. (ii) Arabic uses many types of external objects, such as 'dots', 'Hamza', 'Madda', and diacritic objects. These reasons make the task of line separation and segmentation scripts more difficult. (iii) Arabic characters can have more than one shape according to their position: initial, middle, final, or standalone.

- Arabic uses many ligatures, especially in handwritten text. Ligatures, shown in Fig. 1, are characters that occupy a shared horizontal space creating vertically overlapping connected or disconnected.
- Different writers and the same writer under different conditions will write some Arabic characters in completely different ways [23], as shown in Fig. 2.
- Arabic characters can have more than one shape according to their position inside a word: beginning, middle, end, or standalone, as shown in Fig. 3.
- The characters like 'س' or 'ش' also complicates segmentation and recognition when it occurs in the middle of a word as shown in Fig. 4.

Other characters have very similar contours and are difficult to segment and recognize especially when non-characters and external objects are present in the scanned image. Fig. 5 shows a list of characters. This makes the problems of segmentation of Arabic scripts into characters, and their classification even more difficult [23].

1.4. Motivation and contribution

The characters of Arabic characters are used by a much higher percentage of the world's population to write languages such as Arabic, Farsi (Persian), and Urdu. Thus, the ability to automate the interpretation of written Arabic would have widespread benefits [4]. Arabic handwriting recognition can also enable the automatic reading of ancient Arabic manuscripts. Since written Arabic has changed little over time. Automatic processing can greatly increase the availability of their content. Although it was briefly mentioned that Arabic segmentation and recognition systems for reading handwriting have existed, it is noticed that the development of accurate word recognition systems is still quite an open problem. Segmentation and recognition rates may still be

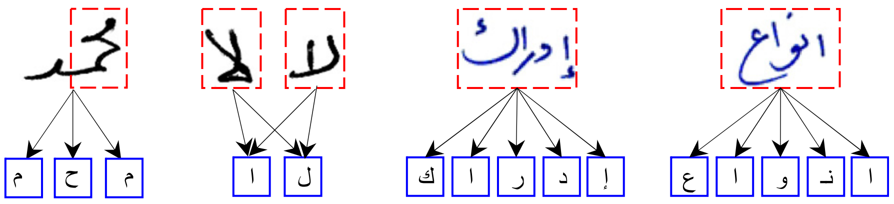


Fig. 1. Arabic ligatures and their constituent characters.

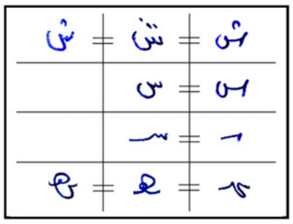


Fig. 2. Four characters written in completely different ways.

Isolated	Final	Medial	Initial
ع	ـع	ـعـ	عـ

Fig. 3. Shapes of the character ‘ع’ takes according to its position inside a word.

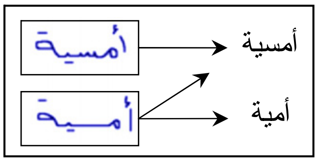


Fig. 4. The letter ‘س’ may be missed when appearing in the middle of a word.

- ‘بـ’ or ‘ـبـ’ like a part from ‘س’ or ‘سـ’
- ‘غـ’ ,‘عـ’ ,‘قـ’ ,‘فـ’, and ‘مـ’ are similar
- ‘حـ’ ,‘جـ’, and ‘خـ’ are similar
- ‘خ’ ,‘ح’ ,‘غ’ ,‘ع’, and ‘ج’ are similar

Fig. 5. Characters with similar contours.

increased and systems still require enhancement. Accurate segmentation techniques are still in demand to be used as important components in the overall Arabic handwriting recognition process.

The original contributions of this work are: (1) A new feature-based heuristic technique is developed completely, that is able to enhance the over-segment of Arabic handwritten words. (2) A new feature extraction technique is improved to enhance segmentation accuracy of Arabic handwritten characters. (3) A new method for calculating direction features is developed. This method lead to reducing steps of normalization processes. (4) A new neural-based segmentation point validation system is introduced. It uses three confidence values from segmentation areas to validate prospective segmentation points. (5) A novel segmentation-based method for verification of segmentation points is also presented. It verifies the validity of the segmentation points set by the initial "segmentor". Another technique is also introduced by merging more than one successive segmentation area to increase the performance and accuracy. (6) Incorporating a combination of novel, heuristic and intelligent techniques for the segmentation of handwritten Arabic words is used. (7) A new fusion equations are presented to process results from segmentation point validation techniques.

2. The techniques and methodologies

The first section describes the feature-based Arabic heuristic "segmentor" (AHS) module in details. Fig. 6 illustrates the various components that compose our entire handwriting segmentation technique.

2.1. Feature-based Arabic heuristic segmenater (AHS)

An integral part of the handwriting recognition process is segmentation. The technique requires an over-segmentation stage which partition handwritten words into primitives that may then be processed further to provide the best segmentation. The segmentation algorithm described below is one such algorithm. It has been named the: feature-based Arabic heuristic algorithm. It employs various heuristics to decide how to split the word into segments. Prior to the detailed discussion, an overview of the algorithm is given in the Fig. 7.

2.1.1. Handwritten word local database

Because there is no standard benchmark database for Arabic handwriting; the database that was used for a majority of segmentation experiments in this research were obtained from twenty different persons, (age 12–50 years), exactly 500 random words were extracted from two paragraphs containing all shapes of Arabic characters written by those persons. In this research, all the word images were converted to (jpg) format. The data that was selected for this research contain different samples of Arabic handwritten word as shows in Fig. 8.

2.1.2. Image preprocessing

This section first and foremost deals with the preprocessing procedures that were investigated and tested on a locally database of handwritten words. Due to the fact that the research deals with a segmentation-based scheme for word segmentation, it was also necessary to deal with the preprocessing of individual characters as well.

2.1.2.1. Thresholding. First: converting the RGB image to grayscale by eliminating the hue and saturation information while retaining the illumination, then converting the grayscale image to binary format (matrix). The output binary image has values of 0 as a foreground pixel (black) for all pixels in the input image and 1 as a background pixel (white) for all other pixels. Since, different thresholds are used for detection; some negligible information (feature) of characters will be lost. However, this will have little effect on the final results. "Word smoothing" algorithm is used to reduce most lost features, and the resulting effect will be very little. All images were converted and only binary images remained and could be used for further processing, an example may be seen in Fig. 9.

2.1.2.2. Noise removal. The goal of this technique is to remove the noise as well as small foreground objects that were not part of the writing. Once the component of word image as matrix were identified, it was possible to perform various useful operations. First and foremost it was possible to transfer word image to binary format, then the small connected components which were

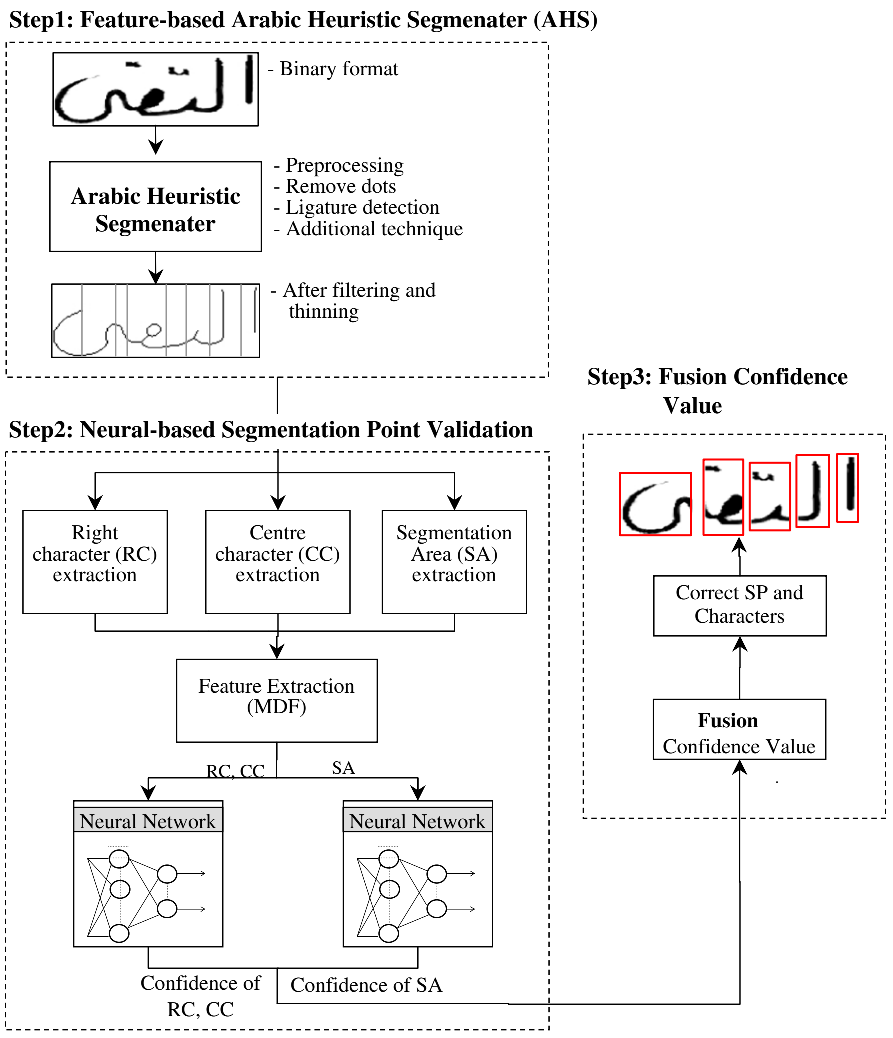

Step1: Feature-based Arabic Heuristic Segmenater (AHS)
- Binary format
Arabic Heuristic Segmenater
- Preprocessing
- Remove dots
- Ligature detection
- Additional technique
- After filtering and thinning
Step2: Neural-based Segmentation Point Validation
Right character (RC) extraction
Centre character (CC) extraction
Segmentation Area (SA) extraction
Feature Extraction (MDF)
RC, CC
SA
Neural Network
Neural Network
Confidence of RC, CC
Confidence of SA
Step3: Fusion Confidence Value
Correct SP and Characters
Fusion Confidence Value

Fig. 6. Components of our complete handwriting segmentation technique.

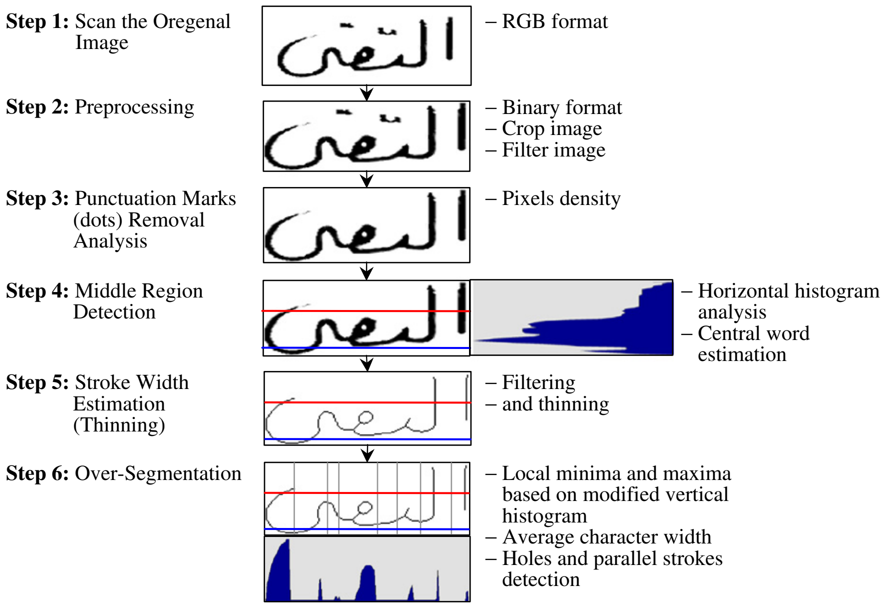

Step 1: Scan the Oregenal Image
– RGB format
Step 2: Preprocessing
– Binary format
– Crop image
– Filter image
Step 3: Punctuation Marks (dots) Removal Analysis
– Pixels density
Step 4: Middle Region Detection
– Horizontal histogram analysis
– Central word estimation
Step 5: Stroke Width Estimation (Thinning)
– Filtering
– and thinning
Step 6: Over-Segmentation
– Local minima and maxima based on modified vertical histogram
– Average character width
– Holes and parallel strokes detection

Fig. 7. Overview of AHS technique.

deemed as being “noise” in the word image and hence determine further processing.

2.1.3. Removal of punctuation marks (dots)

The technique was designed to work on words which have already contained punctuation marks. One interesting issue raised by this image processing procedure is the removal of dots, as shown later, when locating the prospective segmentation point using vertical histogram technique, punctuation marks will cause incorrect segment point because most of the punctuation marks placed on another character or above or under ligature. In general, the density shape of any dot is smaller than any other character in Arabic scripts; therefore we can mark it and then remove. Once the *x* and *y* coordinates of each connected component (dots) in matrix of word image were identified, it was possible to perform various useful operations such as the removal of dots components in the word image by calculating the density of each isolated shapes (foreground pixel), if the density was less than the constant value *c* (calculated based on average character density in Arabic language) the shape (dot) will be marked as dot and then removed by replacing foreground pixel (black color=0) by background pixel (white color=1). Fig. 10 illustrates steps of detect and remove dots.

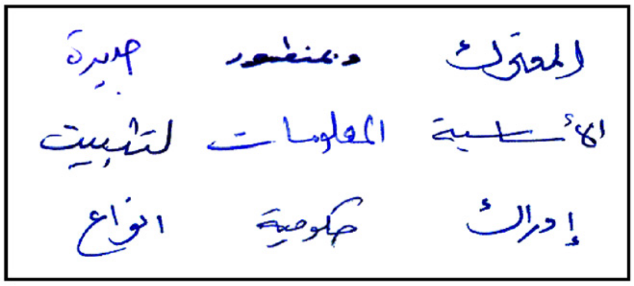


Fig. 8. Samples of handwritten words by different persons.

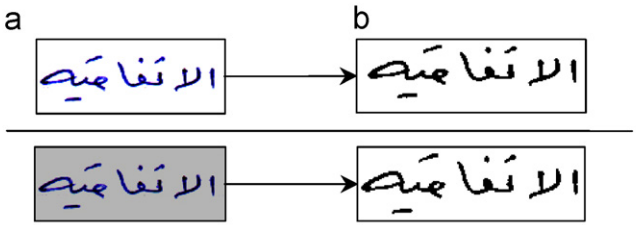


Fig. 9. Example of a “Threshold” image.

Fig. 11a, illustrates example for correct removal dots algorithm. One disadvantage appears for using this dots removal technique is the very small characters that may be recognized as dots, such as the characters ‘ر’ and ‘د’. However, only one writer out of ten wrote some characters similar to dots as shows in Fig. 11b. Improving the technique will be seen in future work by removing the dots that are shown only above upper based line and under the lower baseline (above and under middle region).

2.1.4. Middle region (ligature) detection

A ligature is a small point (stroke) that is used to connect two characters. The major feature of a ligature is usually located within the “middle region” of handwritten words. Hence, a baseline detection technique can be used to identify this middle region. Following this, a vertical histogram is generated based on the middle region of the handwriting to locate possible ligatures [24].

2.1.4.1. Baseline detection. Some Arabic characters are contained on strokes above or under the center of handwritten word body such as: ‘ـة’, ‘ـش’, ‘ـر’, ‘ـكـ’, ‘ـي’ etc. these strokes are called ascenders and descenders, respectively. Ascenders and descenders of the main body of the word may overlap parts of characters in the main body that do not contain such strokes. To increase over-segmentation of the word image, it is necessary to remove ascenders and descenders before the beginning of segmentation process. In this research, this technique calculates the horizontal histogram by counting the total number of foreground pixels (black pixels) of word image. The largest density refers to ligature of the word image. The average values of the maxima and minima on the upper and lower ligature are calculated, respectively. Fig. 12 illustrates an example of horizontal histogram analysis.

2.1.4.2. Stroke width estimation (thinning). Due to the variability of written handwriting, it was first necessary to uniform the characteristics that varied from one word to another. One such important characteristic was reducing the stroke width (thinning). Although thinning word image is detrimental rather than advantageous, sometimes substantial amount of information (features) is lost via the thinning process. It can cause a loss of information in the word image similar to the loss of information at small holes and strokes at some characters. After thinning, some of filled holes are disappeared and therefore removing one of the important features. However, it was necessary to make the stroke width uniform for each word image preparing for the heuristics that will be employed in the segmentation algorithm. A simple algorithm was employed for performing uniform stroke width of each word. The benefits of the thinning

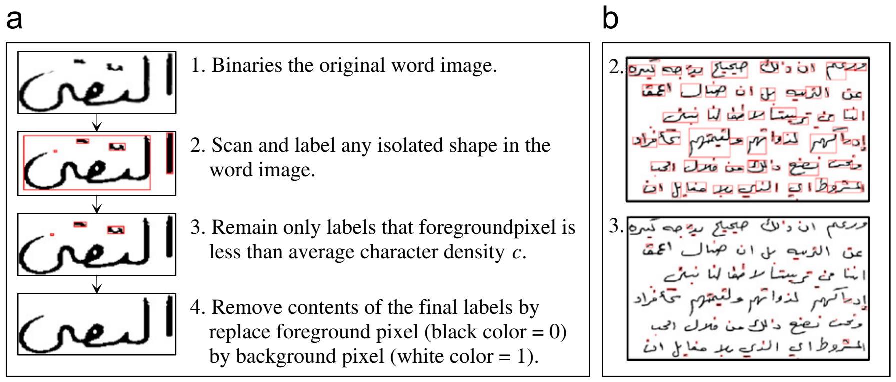

1. Binaries the original word image.
2. Scan and label any isolated shape in the word image.
3. Remain only labels that foregroundpixel is less than average character density *c*.
4. Remove contents of the final labels by replace foreground pixel (black color = 0) by background pixel (white color = 1).

Fig. 10. Steps of detect and remove the dots.

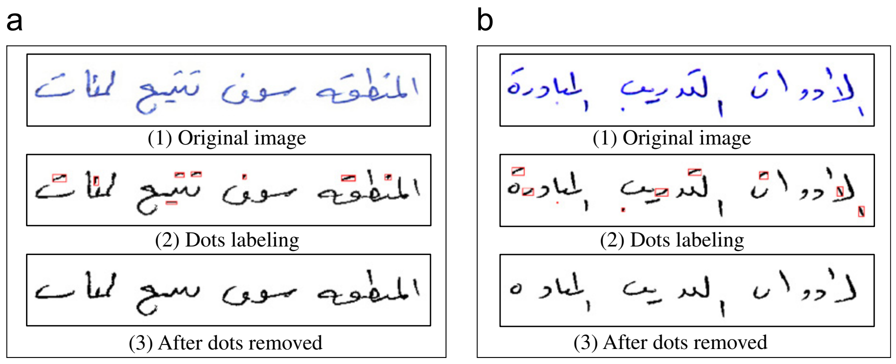


Fig. 11. Arabic words image after removal dots algorithm.

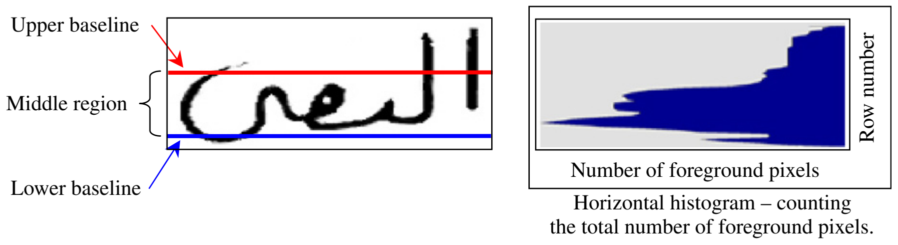


Fig. 12. Baseline based on horizontal histogram.

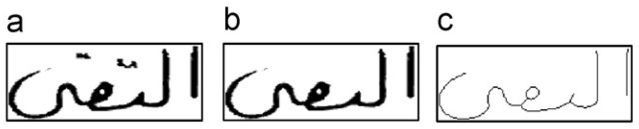


Fig. 13. A word image before and after thinning: (a) original image, (b) word image after removal dots and (c) word image after thinning.

image are: (1) unification a characteristics of varied word image. (2) Locating prospective segmentation point more correctly by using vertical histogram as it will be explain it the next section. (3) Assist to locate the holes and parallel strokes. (4) Speed up the algorithm and make it more efficient. An example after employing the algorithm may be seen in Fig. 13.

2.1.4.3. Modified vertical histogram. After detecting the middle region and thinned stroke width as shown previously, the next step is to locate the prospective segmentation points. One common approach is using the vertical (density) histogram analysis. The analysis is based on the vertical distribution of foreground pixels. The histogram is drawn by counting the total number of foreground pixels (black pixels) in each column of the word image. The histogram is examined for areas that have a pixel density of zero. These areas are marked as definite segmentation points (space between characters). Areas with low pixel density are then identified as prospective segmentation points. Fig. 14 illustrates an example of vertical histogram for the word image before thinning; the vertical histogram is formed based on the middle region of the word “التقى” after removal the dots [24].

One disadvantage using the original image before thinning, excessive number of “low” density regions appears like a noise, whish means excessive number of anomalous points still in the word image. To solve this problem and to improve the accuracy by decreasing the anomalous segmentation points, the histograms are recalculated based on word image after thinning is shown in Fig. 15.

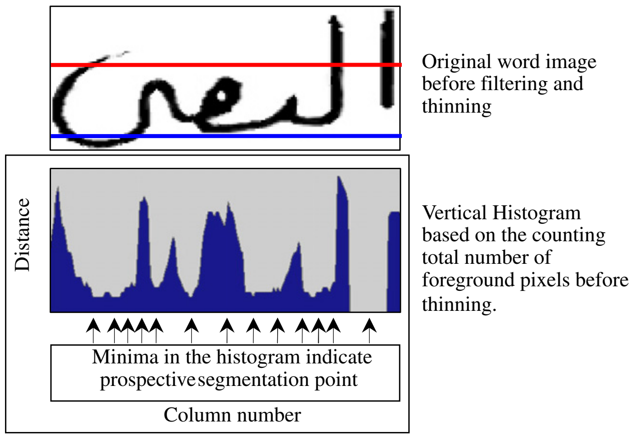


Fig. 14. Vertical histogram analysis before thinning word image.

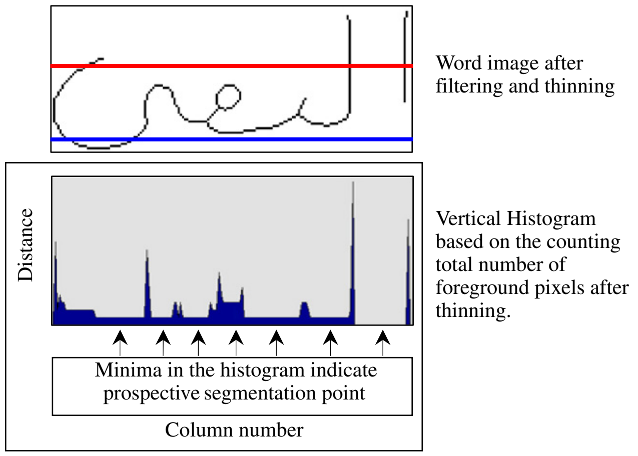
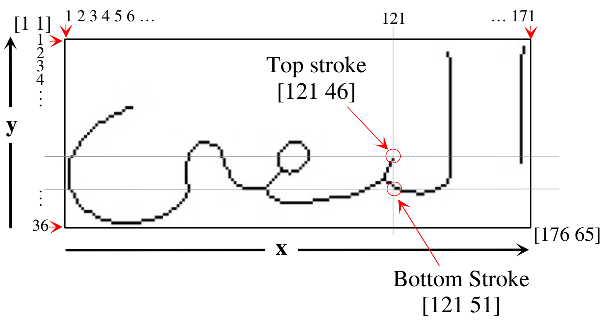


Fig. 15. Vertical histogram analysis after thinning word image.

Although, excessive prospected points are eliminated and the accuracy is improved as shown in the previous figure, many excessive number of “low” density regions appeared this time. This is because the vertical histogram itself is not adequate to distinguish the differences between “holes or parallel stroke” and “ligatures”. In this research, a modified vertical histogram is used to improve the accuracy of prospective segmentation point and solve all previous problems. Modified vertical histogram is calculated based on the distance between top and bottom foreground pixels of word image after thinning. For example, after converting to binary, the word image the work will be on matrix [*x y*], the top left of matrix is [1 1] and the bottom right is [*x y*] depending on size of word image, if we need to calculate the stroke distance *d* from top to bottom of foreground pixels, we must first detected the top and bottom of this stroke on matrix (d=value of bottom stroke−value top stroke) then represent the result of all *x* column as histograms, Fig. 16 shows an example for calculating the distance *d*.

Holes in characters such as ‘و’, ‘ط’, ‘ه’ and parallel strokes in characters such as ‘ـعـ’, ‘ـحـ’ are recognized now as a character not as a ligature. More details about holes and parallel stroke will be seen next section. Fig. 17 shows the modified vertical histogram.



For example:
The distance for point [121 46] and [121 51]
d = value of lower stroke – value upper stroke
d = 51 – 46
d = 5

Fig. 16. Calculating the distance.

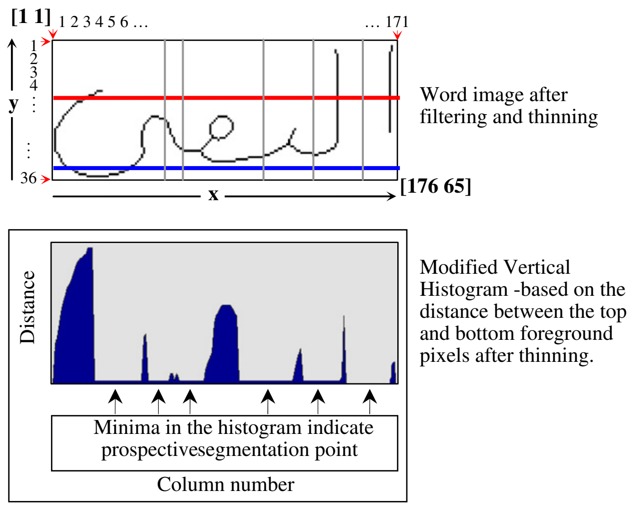


Fig. 17. Modified vertical histogram after thinning word image.

The concept of the modified vertical histogram is formed by calculating the distance between the top and bottom foreground pixels for each column in the word image. As may be seen from the previous figure, the ligature region is clear and hence easy for the segmentor to detect. One weakness of the modified vertical histogram is that it is not suitable for characters with overlapped strokes. But in this research, since the overlapped strokes are removed in most cases (i.e. the modified vertical histogram is formed from the middle region), the advantage of the modified vertical histogram can then be maximized [24]. After filtering and thinning the word image, only one or two vertical pixels are defined as prospective segmentation point. One weakness of the histogram after thinning is that it is not capable to recognize some of characters shape such as ‘ـنـ’, ‘ـبـ’; we call them “missed” characters. They appear as ligatures in the histogram. However, numbers of these cases was very low. Fig. 18, illustrates an example on a word showing a missed segmentation point after filtering and thinning the word image “تقنيات”.

On the other hand, segmentation points are located based on the distance between two local maxima mediating one local minima. Distance (*D*) between two local maxima is defined as prospective segmentation point. The position of this point has been calculated based on average character width and number of vertical pixels. If *D* is less than average character width, prospective segmentation point is located in median of *D*; otherwise if *D* is greater than average character width, prospective segmentation point is located in a position two or three times of distance length. Fig. 19 illustrates a position calculating of prospective segmentation point based on local minima and maxima of modified vertical histogram for the word “التقى”.

2.1.5. Additional techniques

Additional techniques are used to improve results of an over-segmentation, collections of heuristic techniques/features are used in this phase, such as: (a) average character width technique is calculated to add a new correct segmentation point and/or remove bad existed segmentation point, (b) holes detection technique also is used to remove any segmentation point intersect it, for some characters such as ‘ضـ’, ‘ص’, ‘ط’, ‘هـ’, ‘ق’, ‘ه’. Besides, parallel stroke (parallel horizontal lines) detection is used for the same reason above for some characters such as ‘غ’, ‘خ’, ‘ـعـ’, ‘ـحـ’.

2.1.5.1. Average character width. Average character width is an important measurement. If a correct width is calculated, it may assist in confirming whether adding or confirming a new segmentation point exists in a particular area, or removing bad segmentation point existed in a particular area. As seen previously the prospective segmentation point was obtained by calculating the number of local minima based on modified vertical histogram using width of the word image after thinning; the average character width is calculated by dividing the total word image width by the number of local minima, this method in most cases is adequate for obtaining an estimate of the real character width. An example is shown in Fig. 20.

After calculating the average character width, if the distance between two successive prospective segmentation point is greater

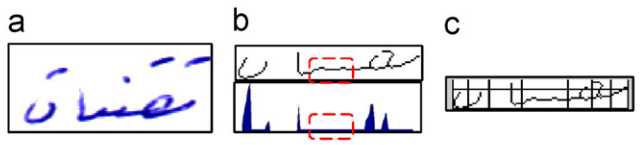


Fig. 18. Example on a word shows a missed segmentation point: (a) original word image ‘تقنيات’, (b) word image after filtering and thinning and its histogram and (c) word image after segmentation.

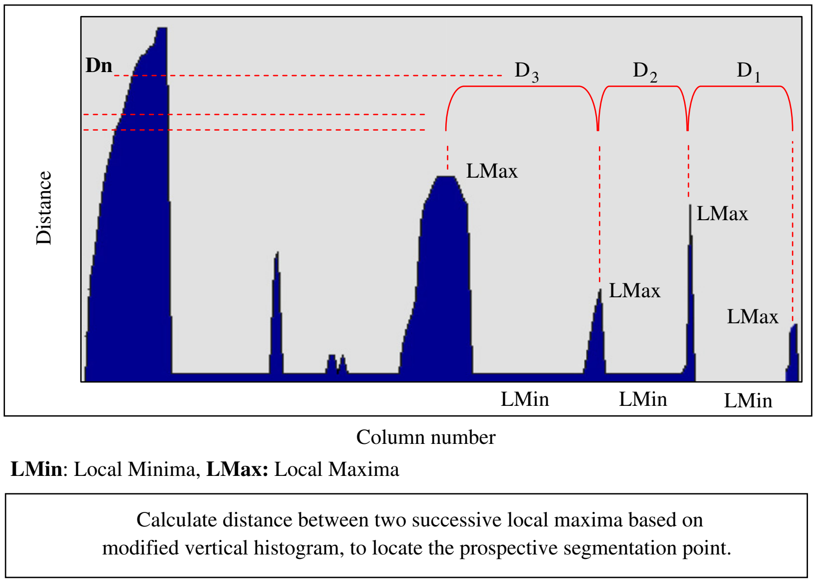
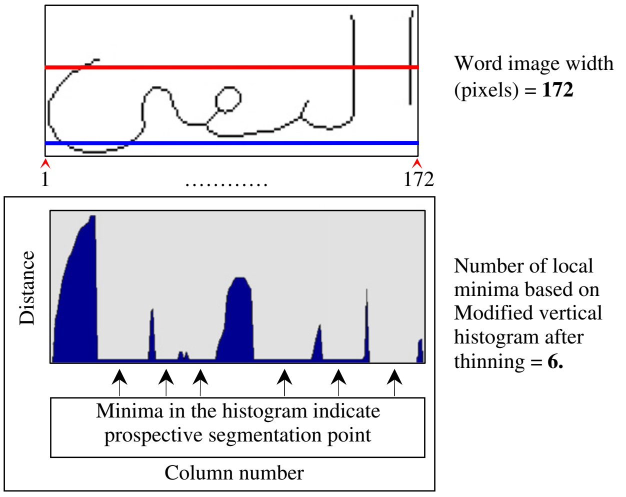


Fig. 19. Calculating position of prospective segmentation point based on local minima and maxima.



For example:
Average character width (pixels) =
(Word image width (pixels)) / (Number of local minima)
= 172 / 6 = 28.66 ≈ **29** pixels

Fig. 20. Calculate average character width (pixels).

than average character width a new segmentation point may added based on local minima and maxima, also if the distance is less than average character width a bad segmentation point may be removed.

2.1.5.2. Close holes and horizontal parallel strokes (open holes) detection. Another feature that is examined to fixing the existence segmentation points is location of close holes and horizontal parallel strokes (open holes). All words are scanned for these holes i.e. areas in a word that may be occupied by an character such as ‘ة’, ‘ـهـ’, ‘ـه’, ‘ـفـ’, ‘ط’, ‘ص’ etc, and for open holes such as areas in a word that may be occupied by any character such as ‘ـعـ’, ‘ـحـ’ etc. To determine whether a close or open hole exists, the word image is scanned (only for segmentation points that are detected previously) from top to bottom (vertically) if determines any white spaces between two horizontal strokes, the holes will detect. Thinning the image assists to locate these white spaces, stroke width after thinning is equal only one pixel, this means if segmentation point passes more than one pixels at the same vertical line intervening white space within the middle region, then the close and open holes will be determined as non-segmentation points. Fig. 21 illustrates determining the location of white space at holes and at horizontal parallel strokes.

2.2. Novel feature extraction

In this research and for the accordance of the characteristics of Arabic scripts, a novel feature extraction technique is developed for extracting structural features from handwritten characters. Modified direction feature (MDF) extraction technique combines local feature vector and global structural information and provides integrated features to a neural network for training and testing. The proposed approach first employs an existing character outline tracing technique, which traces the contour of a given character image. Then, the directions of line segments comprising the characters are detected and the foreground pixels are replaced with appropriate direction values. Finally, features of the characters, based on the location of background to foreground pixel transitions, are extracted and neural training and classification is performed [22,25].

2.2.1. Direction feature extraction

This section explains the details of each component of the MDF extraction technique. First sub-section describes the direction values. The second sub-section provides determining directions. Third sub-section introduces the boundary detection and finally, the normalization of direction values is explained in sub section four.

2.2.1.1. Direction values. After preparing a word images (binaries, preprocessing) and boundary detection is calculated for a word image as shown in previous sections. The first step of determining the characters direction features is definition of the direction values array as following: (i) 2 for vertical direction, (ii) 3 for right

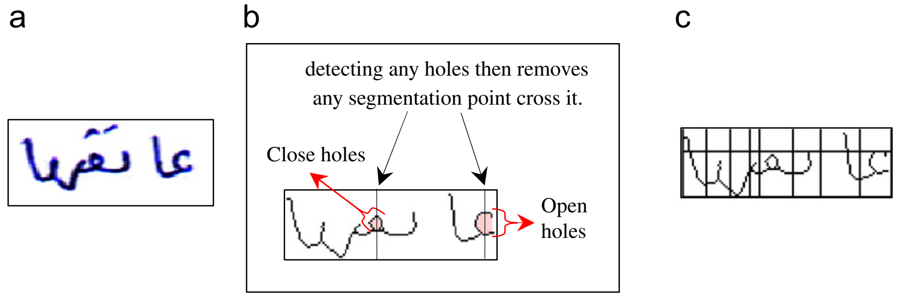
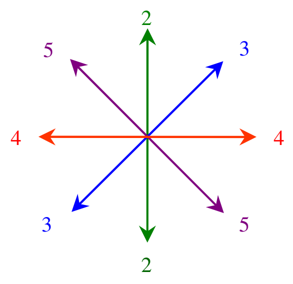

a
b
c
detecting any holes then removes any segmentation point cross it.
Close holes
Open holes

Fig. 21. Holes and parallel stroke detecting: (a) original word image, (b) word image after thinning and detecting close and open holes and (c) word image after remove any segmentation point cross holes.



↕	2	Vertical line segment
⤢	3	Right diagonal line
↔	4	Horizontal line segment
⤡	5	Left diagonal line

Fig. 22. Direction values used for modified direction feature (MDF).

diagonal direction, (iii) 4 for horizontal direction and (iv) 5 for left diagonal direction, all this direction is shown in Fig. 22.

2.2.1.2. Determining directions. Blumenstein et al. [22,25] facilitated the extraction of direction features, the following steps are used to prepare the character pattern:

(1) starting point and intersection point location;
(2) distinguish individual line segments;
(3) labeling line segment information;
(4) line type normalization.

Following to the steps described above, individual strokes in the character images are characterized by one of four numerical direction values (2, 3, 4 or 5). This process is illustrates in Fig. 23.

In this research a new algorithm is proposed to locate and normalize directions of each character, overall processing time is reduced for every word image, the direction features extraction and the normalization processes in whole word image executed is done as a one block. The following steps are proposed to prepare the character pattern and assign the direction values:

(1) scan the right diagonal line of pattern, then replace the original value ‘0’ by the number that dedicated for this direction;
(2) scan the left diagonal line of pattern, and then replace the original value ‘0’ by the number that dedicated for this direction;
(3) scan the horizontal line of pattern, then replace the original value ‘0’ by the number that dedicated for this direction;
(4) scan the vertical line of pattern, then replace the original value ‘0’ by the number that dedicated for this direction;
(5) line type normalization as will be seen in a next section.

Fig. 24 illustrates the steps of new technique to locate and normalize directions of each character image.

2.2.1.3. Boundary detection. One of the main preprocessing ways to calculate direction features is the examination of character components through the connected component analysis. The final result is a set of object boundaries stored as Cartesian coordinates representing the exact locations of connected components in the image. Fig. 25 shows an original word image and the same image after boundary detection.

2.2.1.4. Direction value normalization. There are two steps to normalize a line segment. The first step is finding spurious direction pixels that represent less than two successive pixels and representing it with number 9. The second step is detecting a neighbor direction pixel value, then replacing the spurious pixel that takes number 9 by it. In other words, the value of each spurious direction point will be replaced from number 9 to the normalized value of the line segment. Fig. 26 illustrates the complete process of the direction detection and line segment normalization. As it can be shown, an image of ‘ع’ in its boundary style as binary format is shown in Figs. 26b and c. Then, the image is fed into the new direction by the detection algorithm. Fig. 26d shows the intermediate stage of image processing. One can see from Fig. 26d that the location of direction values for all other foreground pixels has been identified, and in Fig. 26e, the detection of all spurious direction pixels in the image has been labeled. Finally, in Fig. 26f, all line segments in the image have been normalized.

2.2.2. Obtaining the modified direction feature

The MDF feature extraction technique for Arabic characters is described based on location of transition features from background to foreground pixels in the vertical and horizontal directions. In MDF technique, the method calculates location transitions (LTs), and then calculates direction transition (DT) at a particular location. Thereafter, for each transition, a pair of values such as [LT, DT] is stored. This work used the MDF technique for describing Arabic patterns based on their contour or boundary. This prompted an investigation to determine the feasibility of employing MDF for segmentation point validation (SPV), right character validation (RCV), and center character validation (CCV) to enhance the overall segmentation process. More details of MDF have been described in [24].

To calculate transition information LT and DT, it is necessary to scan each row in the image from left-to-right and right-to-left. Likewise, each column in the image must be scanned from top-to-bottom and bottom-to-top. The LT values in each direction are computed as a fraction of the distance traversed across the image. Therefore, as an example, if the transitions were being computed from right-to-left, a transition found close to the right side would be assigned a high value compared to a transition computed further to the left side, as shown in Fig. 27a. A maximum value (MAX) was defined to be the largest number of transitions (MAX

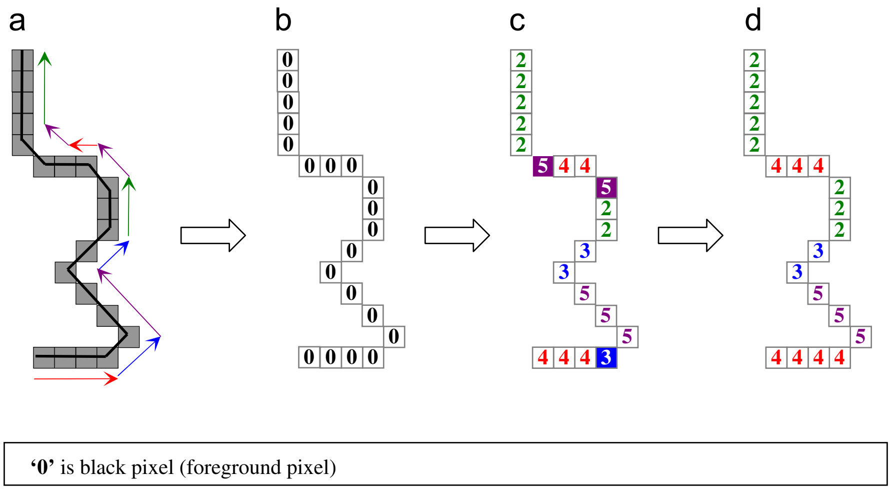


Fig. 23. (a) Original line, (b) line in binary format, (c) after distinguishing directions and (d) after direction normalization.

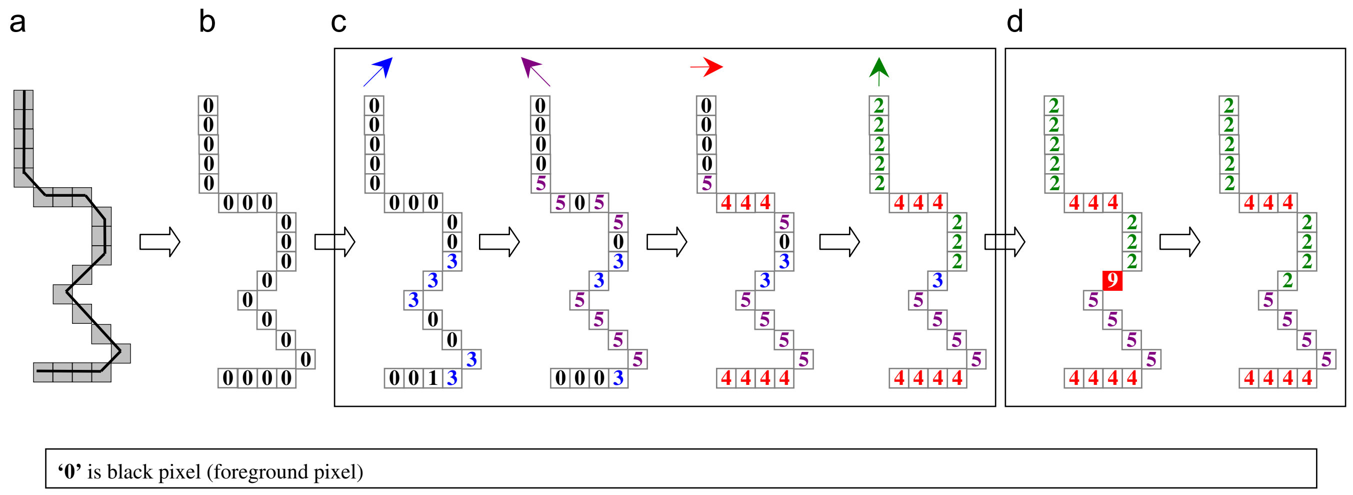


Fig. 24. Steps of new technique to locate and normalize directions: (a) original line, (b) line in binary format, (c) distinguishing directions and (d) normalize the direction (discarding spurious direction values).

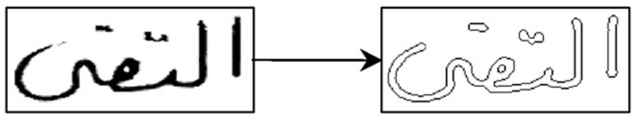


Fig. 25. A word image before and after boundary detection.

equal 3 in this research) that may be recorded in each direction. Conversely, if there were less than MAX transitions recorded (n for example) the remaining MAX–n transitions would be assigned values of 0 (to aid the formation of uniform vectors).

Once a transition in a particular direction is found, along with storing an LT value, the DT value at that position is also stored. The LT value is calculated by dividing the position (P_i) of the transition value by the number of columns/width or number of rows/height according to direction. The DT value is calculated by dividing the direction value (Dv) (at that location) by a predetermined number, in this case: 10. The value 10 was selected to facilitate the calculation of floating-point values between 0 and 1 as shown in Fig. 27a. Therefore, following the completion of the above work, four vectors would be used to represent each set of feature values (eight vectors in total). For both LT and DT values, two vectors would have dimensions MAX $\times$ NC (where NC represents the Number of columns/width of the character) and the remaining two would be MAX $\times$ NR (where NR represents the Number of rows/height of the character). For example, the first direction value from right-to-left in row 9 is equal 2; the transition position P_i for that direction is equal 5 as shown in Fig. 27a, the LT and DT for this transition (only one transition) were calculated as following: LT=1-(P_i/NC)= 1–(5/26)=0.81, DT=Dv/10=2/10=0.2, and so on for all transition in right-to-left, then left-to-right, top-to-bottom, and bottom-to-top.

A further re-sampling of the above vectors was necessary to ensure that the NC/NR dimensions were normalized in size. This was achieved through local averaging. The target size upon re-scaling was set to a value of 5. Therefore, for a particular LT or DT value vector, windows of appropriate dimensions were calculated by determining an appropriate divisor of NC/NR. The average of the LT/DT values contained in each window were stored in a re-sampled 5 $\times$ 3 matrix, as shown in Figs. 27(b, c), for vectors obtained from a right-to-left traversal direction. This was repeated for each of the remaining transition value vectors so

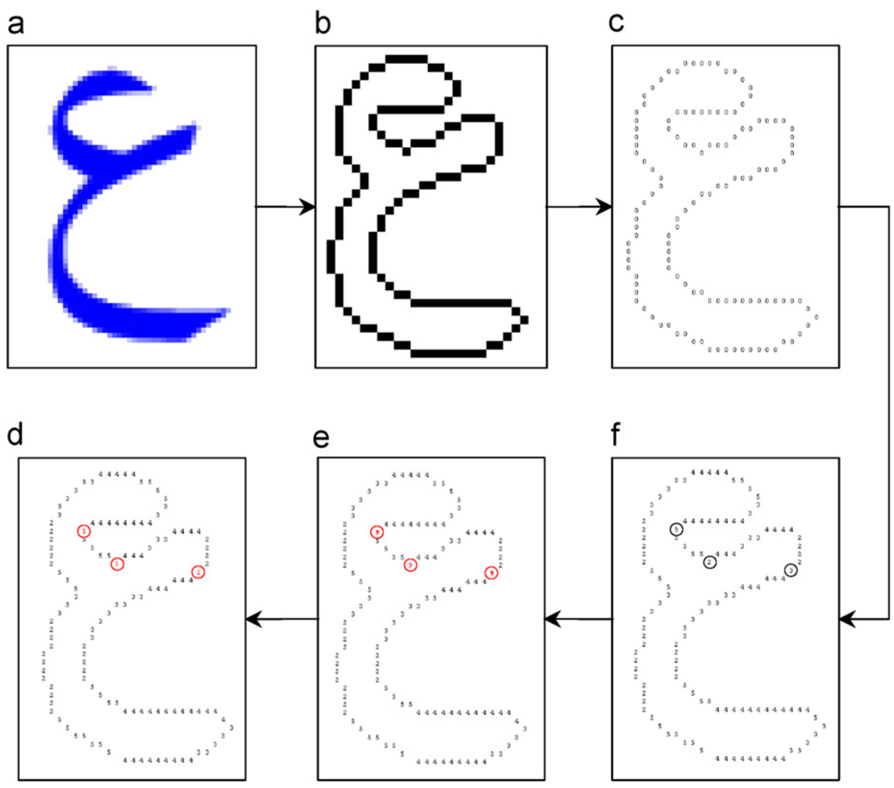


Fig. 26. Full process of distinguishing directions and normalization.

that a final 120 or 160 elements feature vector (total number of MDF features) could be formed using the following formula:

Total MDF features =

Features pair $\times$ number of transitions $\times$ number of directions $\times$ re-sampled matrix size

Where Feature pair [LT, DT] = 2
Number of transitions = 3 or 4 (in this paper = 3)
Number of directions = 4 and
Re-sampled matrix size (width) = 5

For overall transition value vectors, Fig. 28 illustrates the calculations of LT and DT feature vectors in the vertical and horizontal directions.

2.3. *Neural-based segmentation point validation*

As a result of above, after initial segmentation, a type of post-processing is employed to select only the correct segmentation points. A neural confidence-based module is used to evaluate a prospective segmentation point by obtaining a fused value from three neural confidence values: segmentation point validation (SPV), right character validation (RCV), and center character validation (CCV). Therefore it is possible to validate prospective segmentation points, rather than giving a binary (yes or no) decision to whether a segmentation point should be set in a particular region, confidence values are assigned to each segmentation point that is located through feature detection. The confidence value of any segment area can be in the range of 0 to 1 as will describe in the next section.

Therefore, an enhancement to the previously described heuristic algorithm is proposed in this section and it involves segmentation point validation. Following the over-segmentation, this method verifies each segmentation area (SA) to determine whether a particular segmentation area is valid or invalid. An overview of the technique is displayed in Fig. 29.

To test the efficiency of the segmentation point validation technique, we trained neural networks with segment area (SA) (is a point of intersection of segmentation point with the word image (ligature stroke)) right character (RC) (is a character that is located on the right of segmentation point) and center character (CC) (is the character that is located on right of segmentation point) by the training set words. The data used for training was obtained from ten different writers as mentioned previously. To obtain the best results, many configurations were investigated. In particular, classifier architecture was modified as well as the properties of the input vectors with regards to the size and feature extraction technique (or lack of). The classifiers and the configurations are discussed below in detail.

If, for example, the area immediately to the right of the prospective segmentation point (PSP) proves to be a valid character, the network will output a high confidence value RC for that character class. At the same time, if the area immediately centered on the segmentation point provides a high confidence value for the rejected neuron CC, then it is likely that the PSP is a valid segmentation point. The "reject" output of the neural network is specifically trained to recognize non-character patterns, i.e. joined characters, half characters or unintelligible primitives; Fig. 30 illustrates an overview SA, RC, and CC extraction and validation.

The rest of this section is broken down into two sub-sections: First sub-section describes neural network training. Second sub-section describes neural network testing.

2.3.1. *Neural network training*

The classifier chosen for segmentation point validation classification is a feed-forward neural network. The neural network is trained with the back-propagation algorithm [26]. As the back-propagation algorithm is supervised, it is necessary to

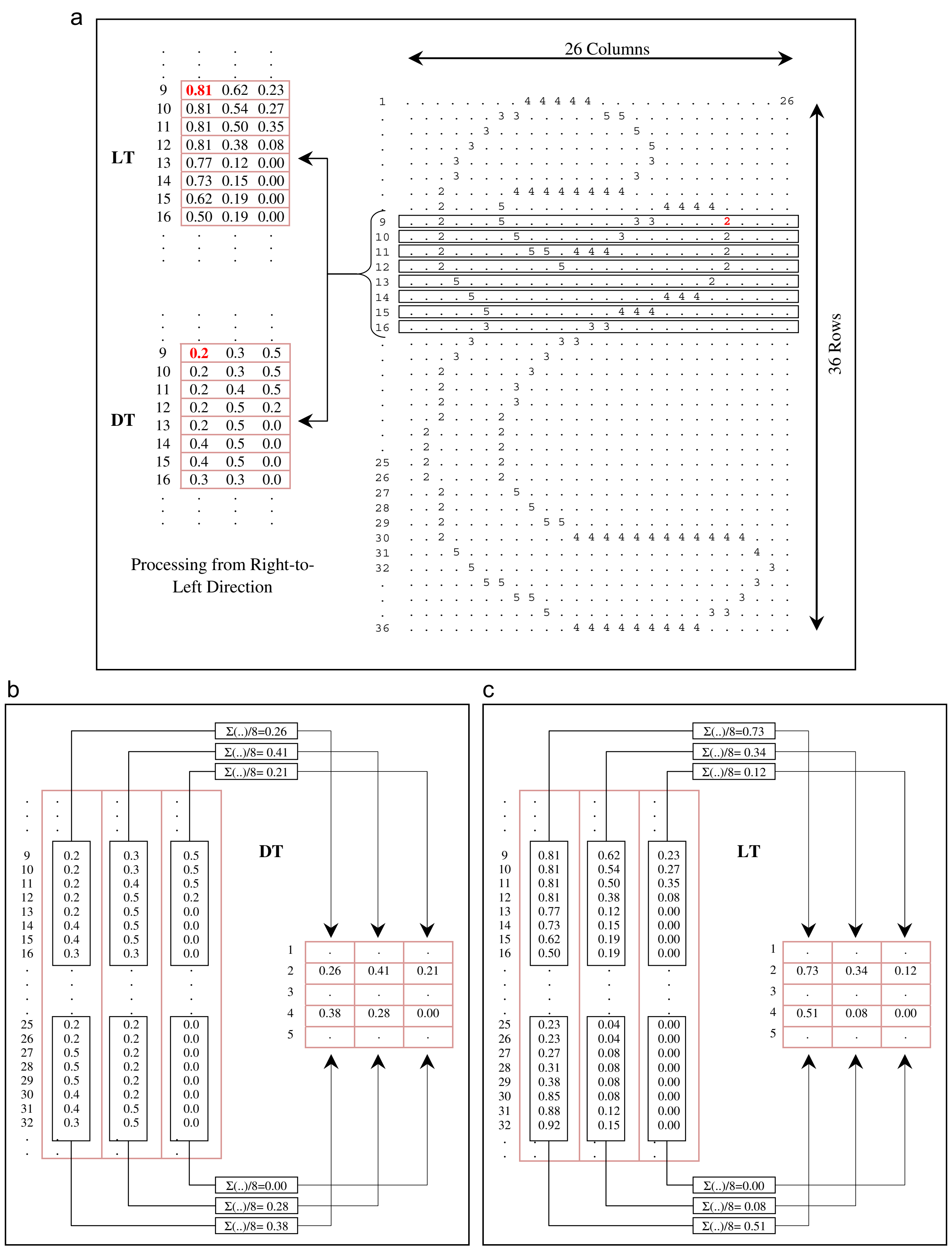


Fig. 27. Calculation and creation of MDF: (a) processing LT and DT values from the right-to-left direction, (b) calculation of DT, and (c) calculation of LT.

provide the neural network with a substantial amount of training data. Two neural networks are used; the first is trained with features extracted from SA originally located by the heuristic algorithm. The neural network verifies whether each particular area is or is not characteristic of a segmentation point. The second is trained with direction features that are extracted from RC and

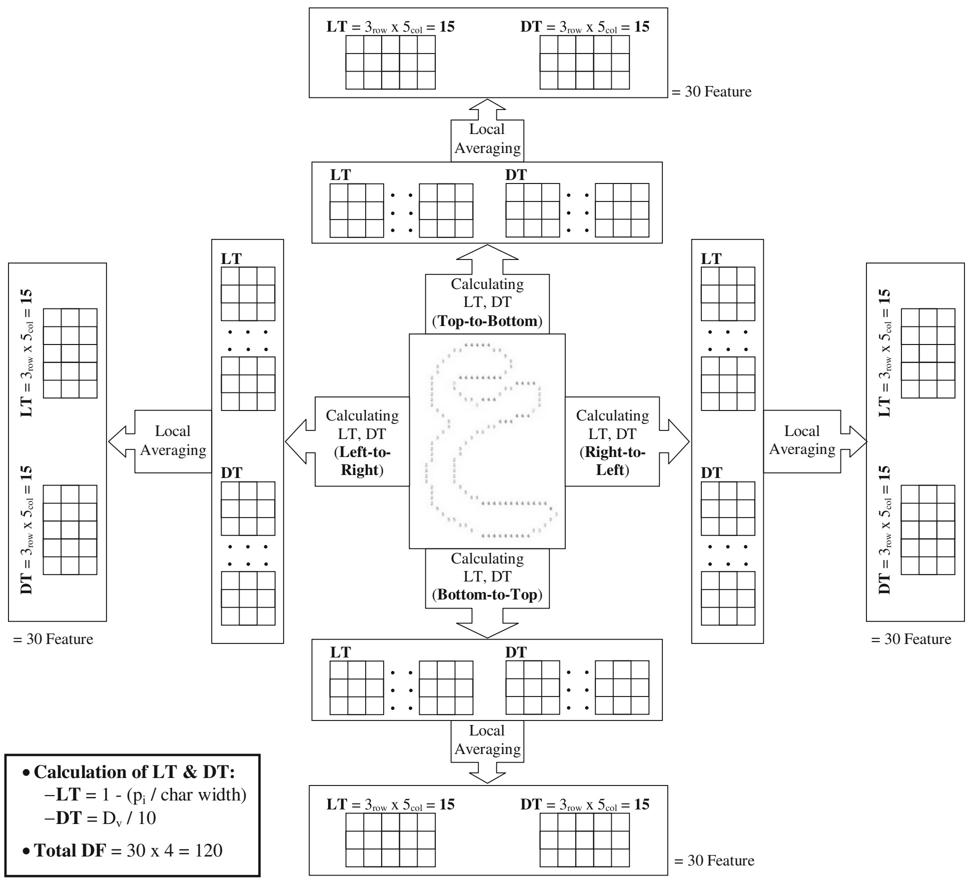

LT = 3row x 5col = 15
DT = 3row x 5col = 15
= 30 Feature
Local Averaging
LT
DT
Calculating LT, DT (Top-to-Bottom)
Calculating LT, DT (Left-to-Right)
Calculating LT, DT (Right-to-Left)
Calculating LT, DT (Bottom-to-Top)
Calculation of LT & DT:
–LT = 1 - (pi / char width)
–DT = Dv / 10
Total DF = 30 x 4 = 120

Fig. 28. All feature vector (LT, DT) in the vertical and horizontal directions.

CC of prospective segmentation point. In many ways, this has been seen as an advantage i.e. faster and more accurate training. Neural networks have been thoroughly tested only on a 620 patterns.

For each training technique, the neural architecture was a feed-forward, fully interconnected. Weights were initially set to small random values. The number of inputs corresponded to the size of the raw. Fig. 31 illustrates a training process for SA, RC and CC validation.

As mentioned previously the number of training set was reduced after the removal of the punctuation marks (dots) from 110 shapes approximately to only 62 shapes. As a result only 620 characters were extracted from the ten writers mentioned above as a training set, as much as 62 characters from every writer. The number of input sits '*P*' depended on the feature extraction vector after normalized; the number of inputs characters set was 620 characters with 120 features extracted for each character. On the other hand, the number outputs of neural network is depended on the different character shape أ - ي that is always a constant, the number of outputs set for every character is a victor contains 62 confidence values, the maximum value of each vector represent a correct character which is the neural network recognize it, the target '*T*' is represented as (0, 1) for a 620 characters. Number of iterations was also varied to discover the best weight configuration for the back-propagation network. A sample of the inputs characters that training by neural network is shown in Fig. 32.

2.3.2. *Neural network testing*

After the neural networks were trained with small errors via training network procedures as shown in the previous section, confidence value for every character is calculated and stored in the "net" parameter; the next step is testing the classification capabilities of the neural network. The algorithm has been used for both the word image and the prospective segmentation point that are extracted from Arabic heuristic segmentor as mentioned in previous sections. Segmentation point is used to extract segment area SA, RC and CC from original word image based on segmentation point location. Direction feature is extracted with vector equal 120 features for each area, then the vector feature is input to the classifier. Three vectors that are extracted from areas outlined above were tested by classifier for validation if it is "valid or "invalid", a classifier outputs confidence value for each area, the value represents if the SP, RC or CC are "valid" or "invalid" based on maximum value of confidence for each one. Finally, SP, RC and CC were validated in such a way that in most cases, all valid segmentation points were retained and most invalid segmentation points were removed, Fig. 33 illustrates a procedure for neural network testing the validity of SA, RC and CC.

The number of iterations (Epochs) was also varied to discover the best weight configuration for the back-propagation network. A sample of the output characters that were tested by neural network is shown in Fig. 34.

2.3.2.1. Combining successive segmentation area (merging). A novel segmentation-based method for verification of segmentation points is also presented. It verifies the validity of the segmentation points set by the initial segmentor; mostly over-segmentation, as shown previously, causes many bad segmentation points. Character width of Arabic scripts are widely different, some of handwritten characters are longer than others such as 'ى', 'ـسـ',

'ب', compared with 'د', 'ه' etc, this causes more bad segmentation points. So a new technique is used to solve this problem. The technique is merging four successive segmentation area successively (first SA, first+second SA, first+second+third SA, first+second+third+fourth SA), then calculating the confidence value for each area, the benefit of the technique is to increase the accuracy and performance of segmentation. As a result, each of segmentation area (SP, RC, and CC) has four confidence values; the

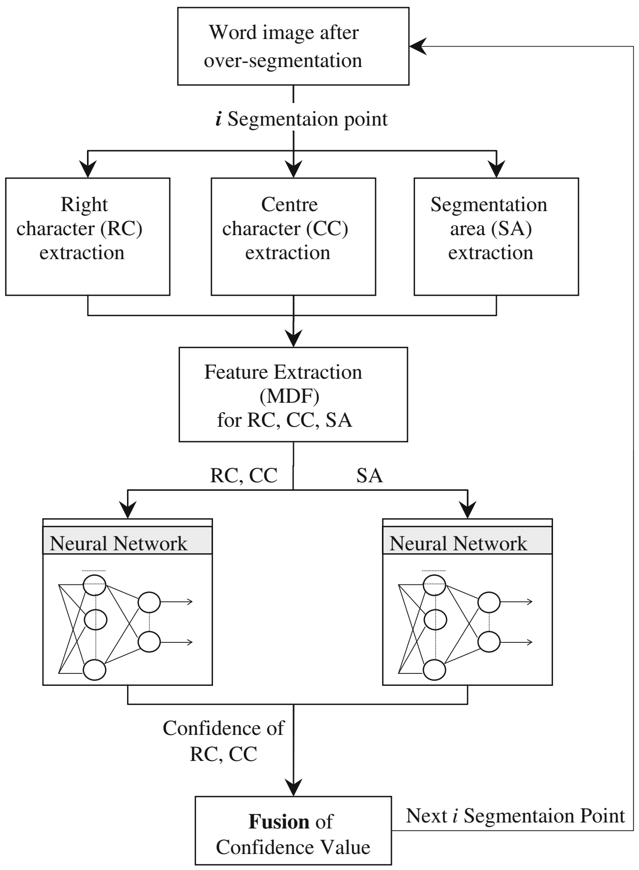

Word image after over-segmentation
i Segmentaion point
Right character (RC) extraction
Centre character (CC) extraction
Segmentation area (SA) extraction
Feature Extraction (MDF) for RC, CC, SA
RC, CC
SA
Neural Network
Neural Network
Confidence of RC, CC
Fusion of Confidence Value
Next *i* Segmentaion Point

Fig. 29. Overview of segmentation point validation procedure (training and testing phases).

maximum value is selected as a segmentation point (fusion value). Two advantages of merging more than one segmentation area; first, to donate a correct segmentation point high confidence value, second; to donate a bad segmentation point low confidence value for eliminating it to enhance overall segmentation results. Finally the real confidence values for each segmentation area are extracted. The next step is fusion confidence values as we will see in the next section. Fig. 35b illustrates a new proposed technique that combines (merges) more than one segmentation area to validation of SA, RC, and CC, and Fig. 35a illustrates a traditional technique.

Following heuristic segmentation and neural confidence value, all words were qualitatively assessed. Next section details the fusion of all SP, RC and CC.

2.4. Fusion confidence values

After heuristic and segmentation techniques were discussed. We introduce a procedure that checks SA, RC and CC activities. In the previous section, for segmentation point validation, two neural networks were used to classify the segmentation areas, the right character and the center character in a word into "valid" and "invalid" classes by analyzing the network's actual confidence value output. The segmentation procedure outlined is divided into three steps. They are outlined below:

Step 1: Heuristic segmentation of words

The Arabic heuristic segmentor described previously is initially executed to over-segment the handwritten words. In further steps each segmentation point set by the heuristic algorithm shall be referred to as a segmentation point (SP).

Step 2: The extraction and analysis of segmentation point, right character and center character

As mentioned previously, a segmentation area (SA) of small dimensions centered about each heuristic segment point is first extracted. A neural classifier trained on "valid" and "invalid" SA is used to determine the nature of the heuristic segment point. Two other areas are also extracted and analyzed. The first area is extracted from the current segmentation point and the previous segmentation point; this area is marked as representing a right character confidence (RCC) area. Second, an area centered about the heuristic segment point is also extracted. The width of the area should be set to the half on width of RC. This area is referred as the center character (CC) area; it shall be referred to as the center character confidence (CCC) area from this point on.

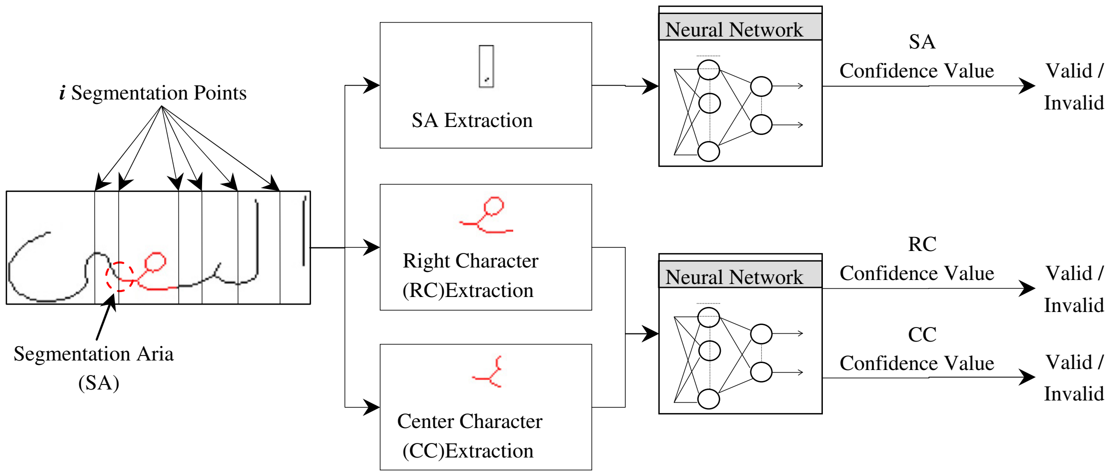


Fig. 30. Overview of SA, RC, and CC extraction and validation.

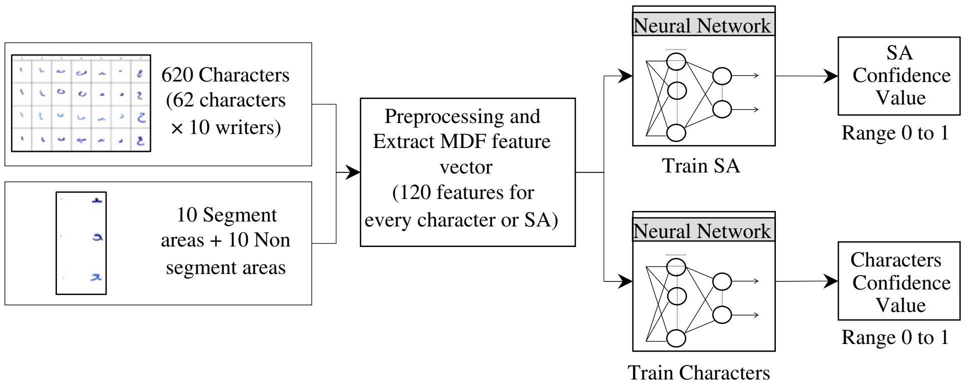


Fig. 31. Training process for SA, RC and CC validation.

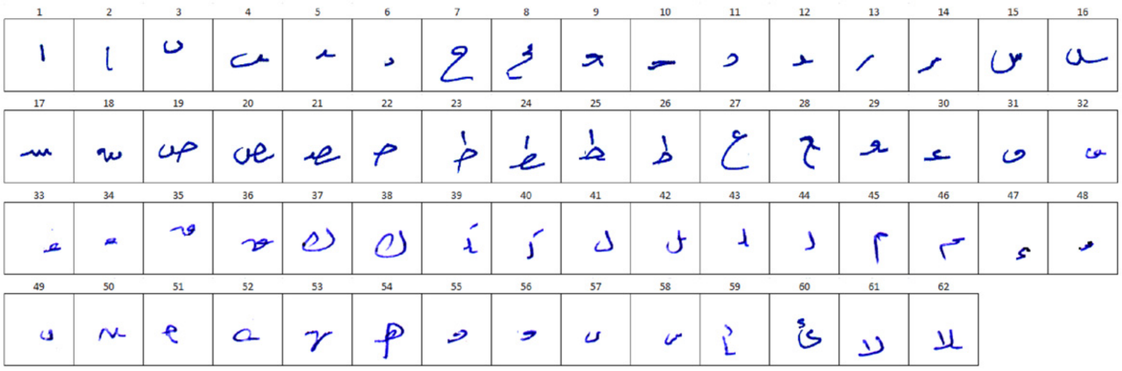


Fig. 32. Sample of 62 inputs character that were wrote by one person.

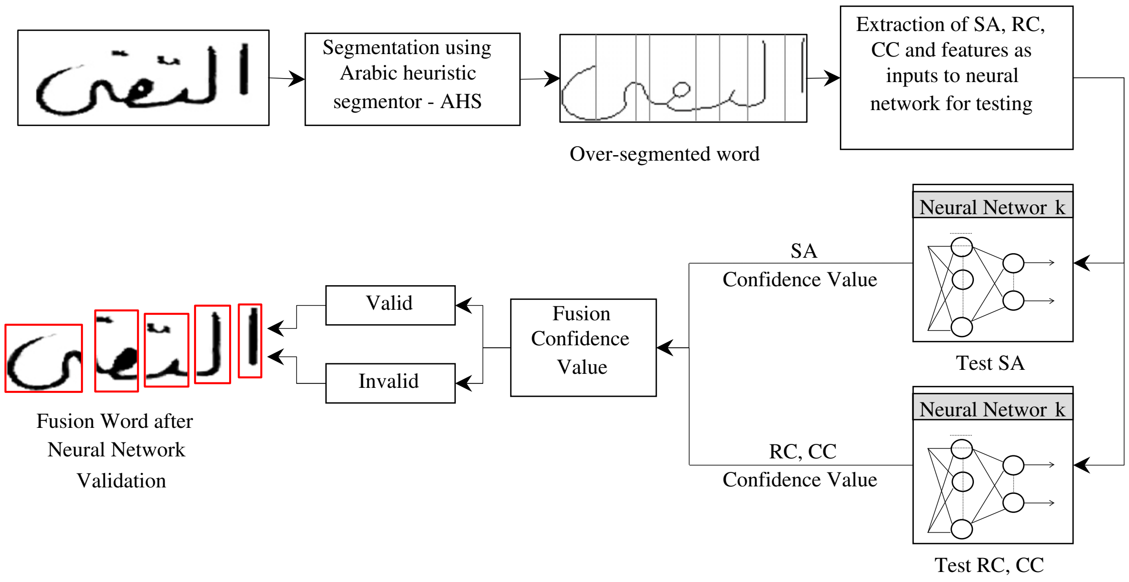


Fig. 33. Procedure for neural network testing the validity of SA, RC and CC.

The analysis of the three extracted areas outlined above (SA, RC and CC-refer to below Fig. 36) is presented in the form of rules below.

The extracted areas are analyzed as described below:

Rule 1: Following RC extraction and neural verification, the area is analyzed.

If the area is identified by the neural expert as one of 62 possible characters, then the segmentation point is more likely to be a correct segmentation point.

If the area is identified as a non-character (rejected), then the segmentation point is more likely to be an incorrect segmentation point

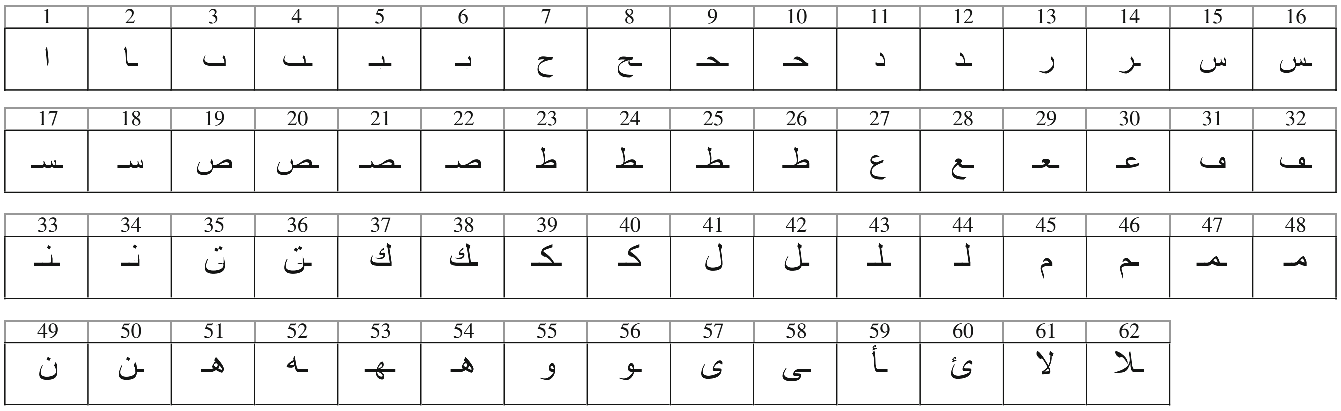

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48
49 50 51 52 53 54 55 56 57 58 59 60 61 62

Fig. 34. The 62 categories of outputs.

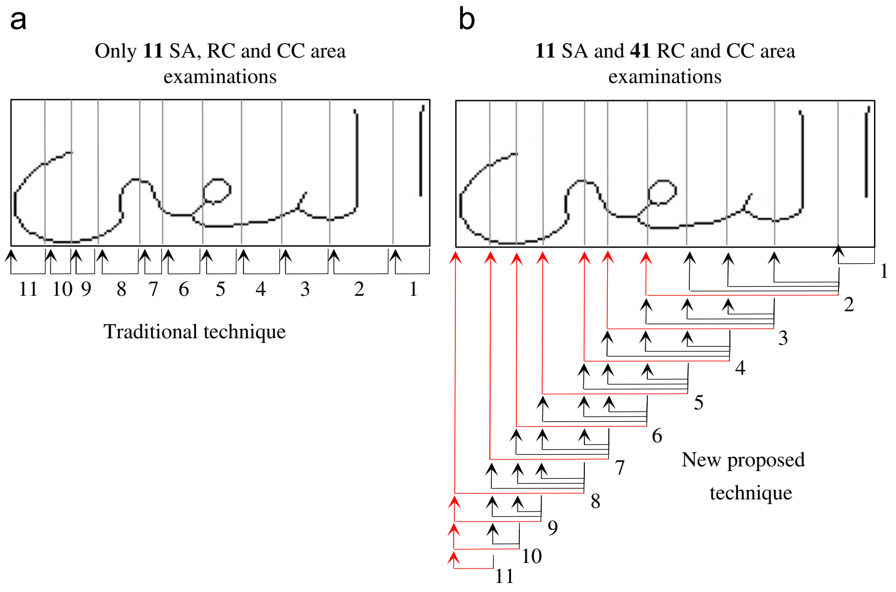

a
Only **11** SA, RC and CC area examinations
11 10 9 8 7 6 5 4 3 2 1
Traditional technique
b
11 SA and **41** RC and CC area examinations
1 2 3 4 5 6 7 8 9 10 11
New proposed technique

Fig. 35. Combination of segmentation area to validation of SA, RC, and CC.

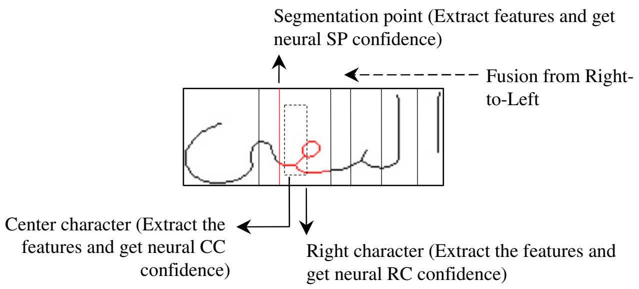

Segmentation point (Extract features and get neural SP confidence)
Fusion from Right-to-Left
Center character (Extract the features and get neural CC confidence)
Right character (Extract the features and get neural RC confidence)

Fig. 36. Indicates the areas (SA, RC and CC).

Rule 2: Following CC extraction and neural verification, the area is analyzed.
If the area is identified by the neural expert as one of 62 possible characters, then the segmentation point is more likely to be an incorrect segmentation point
If the area is identified as a non-character then the segmentation point is more likely to be a correct segmentation point

Rule 3: Following SA extraction, the area is analyzed.
If the neural expert provides a confidence ≥ 0.5, then the segmentation point is more likely to be a correct segmentation point
If the neural expert provides a confidence < 0.5, then the segmentation point is more likely to be an incorrect segmentation point

Step 3: Fusion of Experts
Fusion of the outputs given by each expert is described below, there are two possibilities for each fusion, experiments show dealing with both.

1. Correct segmentation point (CSP)

if	$f_{\text{SPV\_Ver}}(ft1)$	≥ 0.5 and
	$f_{\text{RCC\_Ver}}(ft2)$	is a high character confidence and
	$f_{\text{CCC\_Ver}}(ft3)$	is a high non-character confidence;

$$f_{\text{CSP}}(ft1,ft2,ft3 =)f_{\text{SPV\_Ver}}(ft1)+f_{\text{RCC\_Ver}}(ft2)+(1-f_{\text{CCC\_Ver}}(ft3))$$

or, if $f_{\text{RCV\_Ver}}(ft1)$ ≥ 0.5 and

	$f_{\mathrm{SPV\_Ver}}(ft2)$	is a high segmentation point confidence and
	$f_{\mathrm{CCC\_Ver}}(ft3)$	is a high non-character confidence;

$f_{\mathrm{CSP}}(ft1,ft2,ft3=)f_{\mathrm{RCC\_Ver}}(ft1)+f_{\mathrm{SPV\_Ver}}(ft2)+(1-f_{\mathrm{CCC\_Ver}}(ft3))$

where,	$f_{\mathrm{SPV\_Ver}}$(features)	Confidence value from the segmentation point validation neural network which was tested on correct (0.9) and incorrect (0.1) SA confidence value.
	$f_{\mathrm{RCC\_Ver}}$(features)	Right character confidence (RCC) value from character neural network (highest confidence from 62 characters neuron outputs) which was tested with valid characters and non-character components (incomplete characters)
	$f_{\mathrm{CCC\_Ver}}$(features)	Center Character Confidence (CCC) from character neural network (reject neuron output).

2. Incorrect Segmentation Point (ISP)

if	$f_{\mathrm{SPV\_Ver}}(ft1)$	$<$ 0.5 AND
$f_{\mathrm{RCC\_Ver}}(ft2)$		is a high non-character confidence AND
$\mathrm{f}_{\mathrm{CCC\_Ver}}(\mathrm{ft}3)$		is a high character confidence;

$f_{\mathrm{ISP}}(ft1,ft2,ft3(1=)-f_{\mathrm{SPV\_Ver}}(ft1))+f_{\mathrm{RCC\_Ver}}(ft2)+f_{\mathrm{CCC\_Ver}}(ft3)$

or, if	$f_{\mathrm{RCV\_Ver}}(ft1)$	$<$ 0.5 AND
$f_{\mathrm{SPV\_Ver}}(ft2)$		is a high non-segmentation point confidence AND
$f_{\mathrm{CCC\_Ver}}(ft3)$		is a high character confidence;

$f_{\mathrm{ISP}}(ft1,ft2,ft3(1=)-f_{\mathrm{RCC\_Ver}}(ft1))+(1-f_{\mathrm{SPV\_Ver}}(ft2))+f_{\mathrm{CCC\_Ver}}(ft3)$

where,	$f_{\mathrm{SPV\_Ver}}$(features)	Confidence value from segmentation point validation neural network.
$f_{\mathrm{RCC\_Ver}}$(features)		Right character confidence (RCC) value from character neural network (reject neuron output).
$f_{\mathrm{CCC\_Ver}}$(features)		Center character confidence (CCC) value from character neural network (highest confidence from characters neuron outputs).

3. Finally, the outcome of fusion is decided by the following equation

$f(\mathrm{confidence})=\max\ [(\mathrm{CSP}), f(\mathrm{ISP})]$

where,	CSP	Correct segmentation point.
	ISP	Incorrect segmentation point.

The previous equations indicate a procedure for computing a final confidence value for each SP that exists in a word. A calculation is performed for the two possible cases: CSP and ISP. The greater confidence out put of the two decides the final identity of the SP being examined. If the CSP confidence is greater, the segmentation point will be set as being correct (CSP). Conversely, if the ISP confidence prevails as being larger, the SP is discarded and no longer used in further processing. The entire technique involves an analysis of each word from right to left. The fusion equations used static value equal 0.5 for both of segmentation point validation and right character validation, the value 0.5 is considered as a threshold between the valid or invalid tested area (SPV, RCV). Modifying this value by another value such as (0.4, 0.6 or 0.8) will change the fusion results and, later, the overall segmentation accuracy. Using a new value needs a lot of the studies to determine if it is suitable or not. Many researches have been used value of 0.5 as a basis for fusion equations. Finally, the full program which reflects all the techniques explained previously was built by using MATLAB v.7 release 14; Fig. 37 illustrates the full system of the neural-based Arabic heuristic technique.

3. Experimental results

This section outlines all the relevant results that were conducted by using the techniques of previous sections and before.

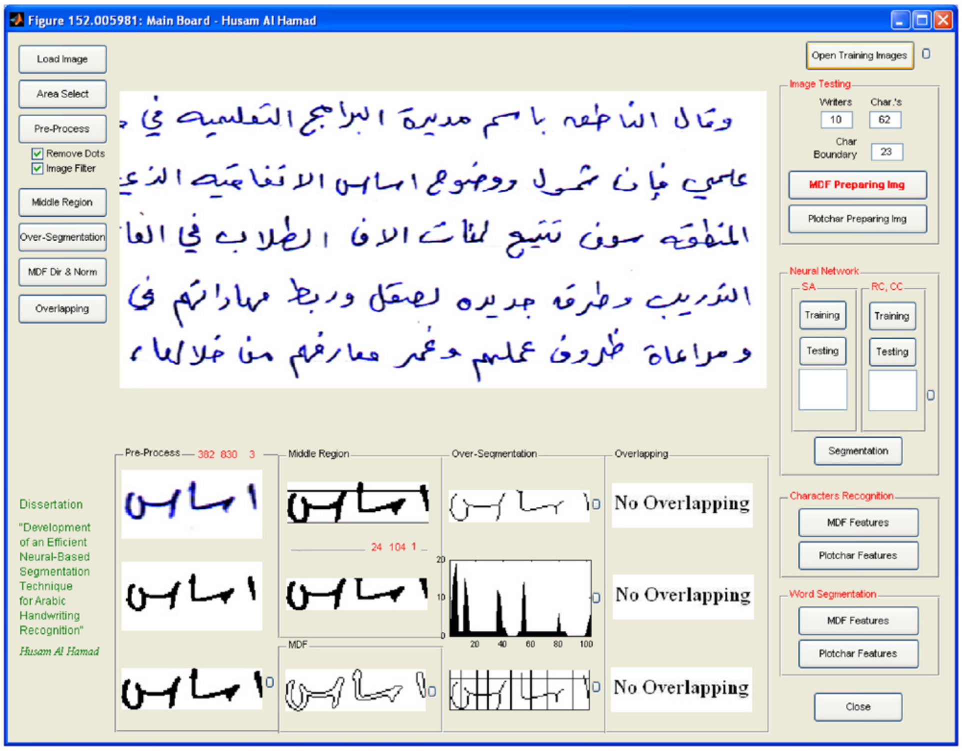
3.1. Feature-based Arabic heuristic segmentor (AHS) results

The results shown below covers the accuracy of the first criterion referred to as *over-segmentation*. If a touching character is divided into more than three components, this is regarded as an error. The second criterion is *missed segmentations.* This refers to the eventuality that two touching characters have not been separated at all. The last criterion is *bad segmentations.* This refers to segmentations that are neither correct nor missed. A "bad" segmentation occurs if for example two touching characters have been split in such a way that either one or both of the characters have been disfigured or particular character components have been incorrectly separated.

The total number of segmentation points at the 500 word samples in the database was 3349. Table 1 shows the number of segmentation errors that were obtained solely for characters. It may be noted that for the results listed below, an "over-segmentation" was recorded if the body of the character was divided into more than three segments; however ligature segments surrounding the character were not taken into account.

As may be seen from Table 1, AHS performed very well with 84.56% accuracy. Error rates were only 0.27% for missed segmentation point, this result is very promising since a set of efficient heuristic techniques were used, Fig. 38 provides samples of handwriting with segmentation points found by AHS.

As may be seen in Fig. 38h, the results of the segmentor missed one "valid" segmentation point at character "ـنـ" in the word "السنوات". In Fig. 39g, the dissections at character "و" was inaccurate and so on in (e, f). More accuracy in Fig. 38d is retained by the location of all "valid" points. Fig. 38(a–c) illustrates an example where, heuristic segmentor retains accuracy by locating all "valid" points and at the same time contains less "invalid" segmentations.



Dissertation

"Development of an Efficient Neural-Based Segmentation Technique for Arabic Handwriting Recognition"

Husam Al Hamad

Fig. 37. GUI program of neural-based Arabic heuristic technique.

Table 1
Over-segmentation performance of AHS with 3349 segmentation points.

SP	Correct segmentation	Over-segmentation error rates			
		Over-segmented	Missed	Bad	Total
3349	**2832**	49	9	459	**517**
	84.56%	1.46%	0.27%	13.71%	**15.44%**

Different writers, and the same writers under different conditions, write some Arabic characters in completely different ways. This decreases the performance of the technique. Good writing of handwritten words is one great reason for reducing missed and bad segmentation points. Writing with avoidance of an overlap and writing the characters shapes clearly are considered a good effect to enhance the overall segmentation processes. Table 2 illustrates results of over-segmentation that is mention according to ten writers. Writers (2, 4–8, 10) with 0.00% "missed" errors indicate clear writing and consistency in fonts.

3.2. Character training and recognition results

In the context of word recognition, character classifiers must be trained with a variety of characters. Certain segmented characters are extracted from handwritten words; only 620 training set patterns are extracted from database. This number of patterns poses a real challenge for training any type of classifier. In an attempt to improve current segmented character results, large number of experiments were conducted on characters extracted from twenty paragraphs written by ten different people. All characters for training were extracted from words inside mentioned paragraphs. Characters were obtained by manually segmenting each word. Characters for testing were extracted automatically segmenting each word using the heuristic algorithm and extracting characters bounded by the segmentation points found. The segmentation points SP, RC and CC validation was conducted following heuristic segmentation.

The majority of character classification experiments were conducted using a neural network trained with the back-propagation algorithm. Three feature extraction techniques SP, RC and CC were used to create appropriate input vectors to each network. Two networks are used to deal with both areas character, one for segment area SA and another one for right character RC and center character CC. In other words, one network is trained on SA and a separate network is trained on RC and CC. The results are displayed in tabular form for each set of experiments. Each table contains columns detailing the number of epochs the network was trained for, the training error, the training performance and the classification rate on the test set.

In this research the CPU time for training a neural network with back-propagation was calculated for each experiment (CPU time is a MATLAB keyword that returns the CPU time in seconds for any process". The system was written in MATLAB v.7 release 14. The PC was used for experiments has a P4 3.42 GHz dual core CPU, 1.50GB memory and uses windows XP x32 operation system.

Before discuss the experiments results of the proposed feature technique, and in order to do comparisons with the experiments results with MDF techniques. *Plotchar* feature algorithm is implemented. The *Plotchar* feature extraction technique is based on the calculations of pixel density features of character image; each character/area image is re-scaled (resized) to 5×7 dimensions, so that, each input vector has 35 features extractions. Fig. 39 illustrates the *Plotchar* feature extraction technique. Later in the next section, Table 11 summarizes the results that are obtained by various researchers compared with final proposed segmentation technique.

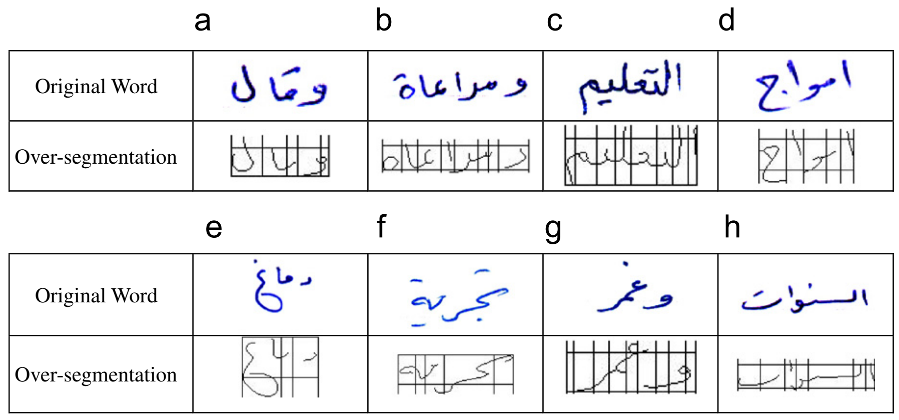


Fig. 38. Samples of over-segmentation word images using the AHS, (a–d) successful words, (e–h) unsuccessful words.

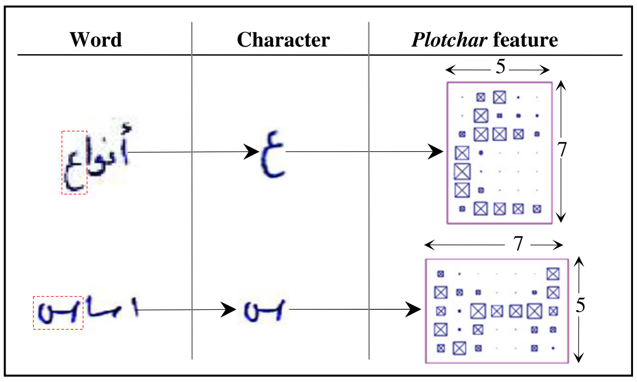


Fig. 39. Extracting the *Plotchar* feature-35 features for each character/area.

3.2.1. *Plotchar feature results*

The *Plotchar* feature experiments were conducted for comparing it with MDF technique as will be seen at next section. All training characters as mentioned previously were obtained from ten different persons databases. It was used to create a training file for the neural classifier. The character images were re-scaled using *Plotchar* feature so that each input vector had the following dimensions: 5×7 or 35 input features. The input vector of character features were passed to the neural classifier. The number of outputs chosen for the neural network was 62 confidence values. As might be expected, the accuracy was not high comparing with other technique; highest result obtained was only 76.77%. The Average CPU time for ten experiments of SP training was 0.4063 s. Table 3 displays results of characters recognition with 620 training/testing pairs using *Plotchar* feature with only 35 inputs for each character/area, and the CPU time of training characters.

This concludes the *Plotchar* feature experiment result section for character recognition. The results above were not promising comparing with the next section modified feature extraction results.

3.2.2. *Modified feature extraction (MDF) results*

All training characters were used to create a training file for the neural classifier. The testing set totaled 620 character patterns. The character images were re-scaled using MDF feature so that each input vector had the following dimensions: [(15 LT+15 DT) features $\times$ 4 directions] or 120 inputs. For these experiments 120 feature extractions were used as mentioned previously, then the matrices of character features were passed to the neural classifier. The number of outputs chosen for the neural network was 62 confidence values. As might be expected, the accuracy was better than *Plotchar* feature, and the top result obtained was just equal to 81.45% recognition on the test set. The Average CPU time for ten experiments of SP training was 0.4844 s. Table 4 illustrates results of characters recognition with 620 training/testing pairs using MDF feature with 120 inputs for each character/area, and the CPU time of training characters.

The results above were promising, especially since a MDF feature extraction technique was used for all experiments. On the other hand, to enhance the classification rates, the number of testing set must be increased at least two-fold or three-fold, this number of testing patterns aids to improve the accuracy of overall segmentation as mentioned previously on literature. Fig. 40 illustrates the effects of altering the epoch's number for characters recognition using MDF and *Plotchar* feature.

As we have seen previously, the *Plotchar* algorithm requires significantly less CPU time than MDF for training each of SP and characters. This is because each vector in *Plotchar* has 35 input features, conversely, the MDF algorithm has 120 input features in each vector. Fig. 41 illustrates the CPU times (in seconds) for training characters using MDF and *Plotchar* feature."

3.3. *Word segmentation results*

The accuracy of the technique can affect the efficiency and accuracy of the word segmentation module that is employed in subsequent sections. Therefore, the Arabic heuristic technique was, instead, evaluated qualitatively in terms of ability to locate all "valid" segmentation points. The results presented here are divided into two sub-sections. The first discusses experiments results for the *Plotchar* feature extraction techniques. The second section deals with describing experiments results for the modified feature extraction MDF techniques.

3.3.1. *Plotchar feature result*

The experiments presented here are using *Plotchar* feature to locate segmentation points from word image based on the neural network algorithm. These experiments give a more accurate view of the execution of the segmentation technique, as it deals with the same input patterns for training. Tables in this section report segmentation point validation accuracy. Fig. 42 displays successfully segmented word images using heuristic segmentation with and without segmentation point validation. As may be seen in Fig. 42h, enhanced neural-based segmentation technique missed two "valid" segmentation points at character

Table 2
Over-segmentation performance of AHS by ten writers.

Writers	SP	Correct segmentation	Over-segmentation error rates			
			Over-segmented	Missed	Bad	Total
Writer (1)	348	**302**	1	4	41	**46**
		86.78%	0.29%	1.15%	11.78%	**13.22%**
Writer (2)	332	**273**	7	0	52	**59**
		82.23%	2.11%	0.00%	15.66%	**17.77%**
Writer (3)	387	**322**	5	1	59	**65**
		83.20%	1.29%	0.26%	15.25%	**16.80%**
Writer (4)	309	**270**	12	0	27	**39**
		87.38%	3.88%	0.00%	8.74%	**12.62%**
Writer (5)	302	**249**	5	0	48	**53**
		82.45%	1.66%	0.00%	15.89%	**17.55%**
Writer (6)	353	**304**	1	0	48	**49**
		86.12%	0.28%	0.00%	13.60%	**13.88%**
Writer (7)	312	**273**	5	0	34	**39**
		87.50%	1.60%	0.00%	10.90%	**12.50%**
Writer (8)	350	**296**	5	0	49	**54**
		84.57%	1.43%	0.00%	14.00%	**15.43%**
Writer (9)	355	**282**	5	4	64	**73**
		79.44%	1.41%	1.13%	18.03%	**20.56%**
Writer (10)	301	**261**	3	0	37	**40**
		86.71%	1.00%	0.00%	12.29%	**13.29%**

Table 3
Experiment results of characters recognition with 620 training/testing pairs using *Plotchar* feature-35 inputs.

Experiment	Training performance	Epochs	Training error (%)	CPU time (S)	Classification rate (%)	Classification rate set
1	0.0031180	200	19.19	15.2813	75.65	469/620
2	0.0029664	300	17.93	22.1250	76.45	474/620
3	0.0031210	500	19.35	38.1094	76.13	472/620
4	0.0030170	700	19.31	53.7188	75.97	471/620
5	0.0029390	1000	17.76	79.0781	76.77	476/620

Table 4
Experiment results of characters recognition with 620 training/testing pairs sing MDF feature and 120 inputs.

Experiment	Training performance	Epochs	Training error (%)	CPU time (S)	Classification rate (%)	Classification rate set
1	0.0014476	200	8.87	27.3906	81.13	503/620
2	0.0011650	300	7.10	40.9375	82.26	510/620
3	0.0013000	500	8.06	68.2813	79.68	494/620
4	0.0011966	700	7.42	94.7188	81.45	505/620
5	0.0013520	1000	8.39	141.9375	78.55	487/620

"ـسـ" and "ـا" in the word "باسم", and so on in (e–g). At (a–d) more accuracy is retained by the location of all "valid" points. (a–d) illustrates example where enhanced neural-based segmentation technique retains accuracy by locating all "valid" points and at the same time contains less "invalid" segmentations.

The results of the enhanced neural-based segmentation technique are calculated based on the number of correctly and incorrectly identified segmentation points in word samples. Neural network verifies whether segmentation points are valid or invalid based on neural confidence-based module for validating the prospective segmentation points. If network outputs a height confidence value this indicates that a point is a valid segmentation point; a low confidence value indicates that a point should be ignored, Table 5 shows the overall performance results of the enhanced neural-based segmentation technique using SPV fusion.

Also, Table 6 shows the overall performance results of the enhanced neural-based segmentation technique using RCV fusion.

Table 7 illustrates experiment results of neural-based segmentation technique divided according to ten writers using RCV fusion.

3.3.2. *Modified feature extraction (MDF) result*

The final experiments presented here are for using MDF feature to locate segmentation points from word image based on the neural network algorithm. These experiments give more accurate view of the execution of the segmentation technique, as it deals with the same input patterns for training. Tables demonstrate segmentation point validation accuracy. Fig. 43 displays successfully segmented word images using heuristic segmentation with and without segmentation point validation.

Table 8 shows the overall performance results of the enhanced neural-based segmentation technique.

Table 9 shows the overall performance results of the enhanced neural-based segmentation technique using RCV fusion.

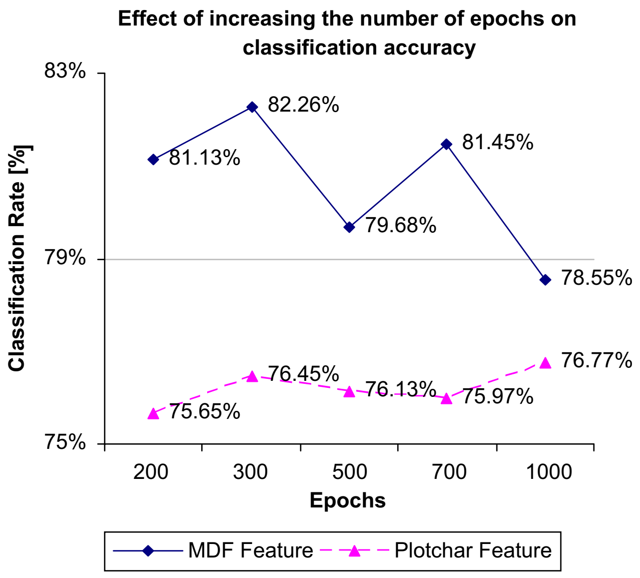


Fig. 40. Effect of increasing the number of epochs on classification accuracy for characters recognition using MDF and *Plotchar* features.

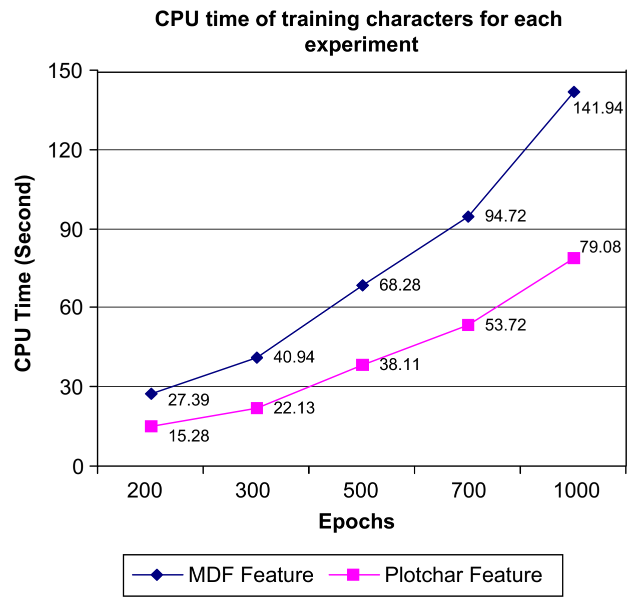


Fig. 41. CPU times (second) of training characters using MDF and *Plotchar* feature.

The incorrectly identified of valid and invalid segmentation points were due to the neural classifier misrecognising two joined characters as a single character. The solution is to employ a better neural classifier and increase size of training set. Training set was extracted from ten writers (different in age and gender); it consisted only of 620 characters, which is not enough to train the neural network. Increasing the training set size will be used to enhance the performance of the neural classifier and then to improve the overall segmentation. Another reason for improving the segmentation performance is to use MDF technique and removing punctuation marks (dots). Since word without dots can be extracted, and MDF provides better features for the single classifier, hence, the performance of RCV and CCV are improved. Table 10 illustrates experimental results of neural-based segmentation technique according to ten writers with RCV fusion.

Fig. 44 illustrates the tabulated results above in Tables 6 and 9. In particular it compares the performance of segmentation validation between MDF and *Plotchar* feature using RCV fusion.

Fig. 45 illustrates the comparison between MDF and *Plotchar* based on RCV and SPV fusion

Finally, Fig. 46 illustrates the final comparison between AHS results and performance results of neural-based segmentation validation using *Plotchar* and MDF feature direction for ten writers.

4. Analysis and comparison of results

The purpose of this section is to analyze the experimental results obtained in this research and to discuss their significance. The section also presents a comparison of the results with those of other researchers in the field. As may be seen in previous section, there are three main sections. The first section deals with the results of over-segmentation, second section deals with character recognition results and the final section deals with segmentation results. The current discussion is also sectioned in the same manner. Each section provides insight into the nature of the results obtained. At the conclusion, a brief comparison between the results of other researchers is presented.

4.1. *Analysis of feature-based Arabic heuristic segmenater (AHS)*

Words segmented by the heuristic algorithm were visually inspected and were required to meet two important criteria. First, it was imperative that the algorithm located all “valid” segmentation points. Second, it was preferable that the number of over-segmentations or “invalid” segmentations was kept at a minimum. Fig. 38 illustrates some words that have been segmented by the heuristic algorithm. In the examples shown, the segmentor performed quite well, locating all valid points. It is also interesting to note that in certain cases such as the words “أمواج” and “التعليم”, the number of over-segmentations is quite low. This is important for further processing, i.e. segmentation point validation, (SPV) as it may reduce processing time.

In most cases the criteria outlined above were met very successfully. However, problems did arise in segmenting very noisy word images and those that contained tightly coupled characters or characters with large interfering or ornamental strokes. One such example may be seen in Fig. 38e in the word “دماغ”. The characters “ـا” and “غ” are very tightly coupled. In general, the proposed technique of the algorithm performed quite well, but still did not manage to fully separate the two characters. The main reason that separation is difficult with tightly coupled characters is that in certain instances vertical segmentations are not sufficient to perform a perfect or “clean” split. The second problem, which is large interfering or ornamental strokes, was the cause of the following errors: “missed” or “bad” segmentations. The former means that no segmentation point was set in a “valid” segmentation zone. The latter refers to characters that are segmented but are not “cleanly” separated. An example of this may be seen in Fig. 38f, “تجربة”. An interfering “ر” stroke protrudes across to the “ـبـ”. This instances is quite minor and would not affect segmentation greatly, however in some cases the interfering stroke may be large resulting in an enlarged primitive that contains a piece of the interfering stroke. These problems were not very frequent; however missed segmentations did affect the end character segmentation rates.

There are many reasons for increasing accuracy overall performance results of segmentation word. First, writing the word image with clear font and clear strokes between characters. Second, avoid writing with overlap character and writing properly

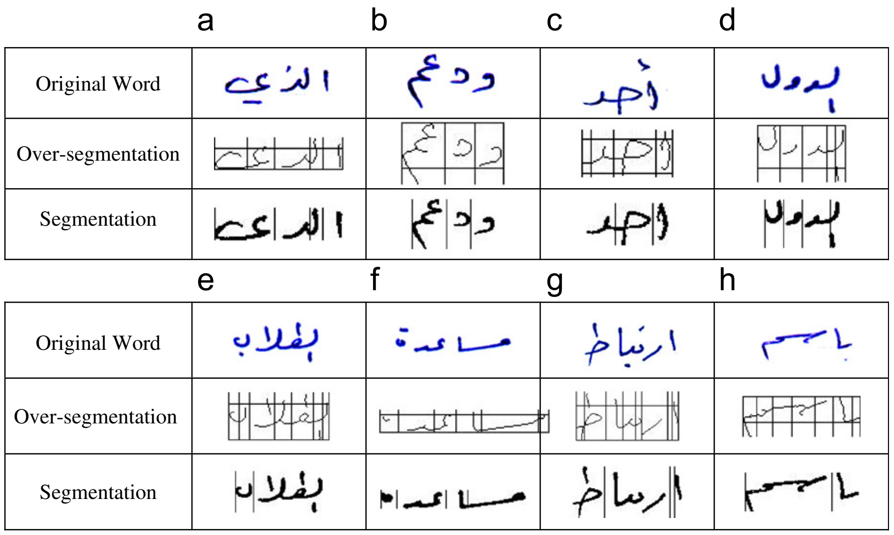


Fig. 42. Sample word images segmented by the feature-based Arabic heuristic segmentor (a–d) successful words, (e–h) unsuccessful words.

Table 5
Neural-based segmentation technique results, using SPV fusion.

Result	Correctly identified		Incorrectly identified	
	Valid	Invalid	Valid (bad)	Invalid (missed)
Subtotals	1433	771	207	938
%	42.79%	23.02%	6.18%	28.01%
Total	**2204**		**1145**	
%	**65.81%**		**34.19%**	

Table 6
Neural-based segmentation technique results, using RCV fusion.

Result	Correctly identified		Incorrectly identified	
	Valid	Invalid	Valid (bad)	Invalid (missed)
Subtotals	2215	198	682	254
%	66.14%	5.91%	20.36%	7.58%
Total	**2413**		**936**	
%	**72.05%**		**27.95%**	

Table 7
Neural-based segmentation technique results for ten writers, using RCV fusion.

Writer	SP	Correctly identified			Incorrectly identified		
		Valid	Invalid	Total	Valid (bad)	Invalid (missed)	Total
Writer (1)	348	238	12	**250**	70	28	**98**
		68.39%	3.45%	**71.84%**	20.11%	8.05%	**28.16%**
Writer (2)	332	199	20	**219**	82	31	**113**
		59.94%	6.02%	**65.96%**	24.70%	9.34%	**34.04%**
Writer (3)	387	235	39	**274**	90	23	**113**
		60.72%	10.08%	**70.80%**	23.26%	5.94%	**29.20%**
Writer (4)	309	215	23	**238**	48	23	**71**
		69.58%	7.44%	**77.02%**	15.53%	7.44%	**22.98%**
Writer (5)	302	214	16	**230**	50	22	**72**
		70.86%	5.30%	**76.16%**	16.56%	7.28%	**23.84%**
Writer (6)	353	223	21	**244**	74	35	**109**
		63.17%	5.95%	**69.12%**	20.96%	9.92%	**30.88%**
Writer (7)	312	216	20	**236**	61	15	**76**
		69.23%	6.41%	**75.64%**	19.55%	4.81%	**24.36%**
Writer (8)	350	242	19	**261**	63	26	**89**
		69.14%	5.43%	**74.57%**	18.00%	7.43%	**25.43%**
Writer (9)	355	223	18	**241**	91	23	**114**
		62.82%	5.07%	**67.89%**	25.63%	6.48%	**32.11%**
Writer (10)	301	210	10	**220**	53	28	**81**
		69.77%	3.32%	**73.09%**	17.61%	9.30%	**26.91%**

font. This is clearly evident in Fig. 38(a–d) as an example of the word written clearly; that were led to reduce missed or bad segmentation points. On the other hand, the words (e–h) written with overlap and bad font, which means there is numerous number of missed or bad segmentation point. The final results of located prospective segmentation points from word image are extracted by using AHS algorithm. As may be seen from Table 1, AHS performed very well with 84.56% accuracy. Error rates were only 0.27% for missed SP, this result is very promising because a set of efficient heuristic techniques were used. Also the over-segmented result with only 1.46% is very promising as may be seen from Table 1. Finally, and as may be seen from Table 1, bad segmentation points with 13.71% is a large rate, the reason of this result is referred to the characteristic of some characters. Writing some of characters by some people is done with wide width compared with average characters width that calculated i.e. ‘صــــ’, ‘ب’, ‘ســـ’, ‘ص’, ‘ب’, ‘س’ and ‘ـــــ’ etc., most of it like two or three characters such as ‘ـبـ’, ‘ـب’, another reason refers to similarity part of some characters with some other characters. For example, ‘ر’ is like a part of ‘س’ or ‘ص’ and ‘ـبـ’/‘ـب’ also is like a part of ‘س’ or ‘ســ’, character similarity can seen clearly from handwriting characters.

4.2. Analysis of character recognition results

There were quite a large number of experiments conducted. It is never easy to compare results for handwritten character recognition with other researchers in the literature. The main problems that arise are the differences in experimental methodology, the different experimental settings and the difference in the handwriting database used. The primary goal of the experiments was to achieve the best possible recognition accuracy. Therefore as could be seen in the previous section, a two feature extraction techniques were examined along with only one neural classifier. The results are examined in the same order as they were presented before. The training set contained 620 characters, and hence the neural network was configured to have 62 outputs (characters without dots). The noise or local distortions that the

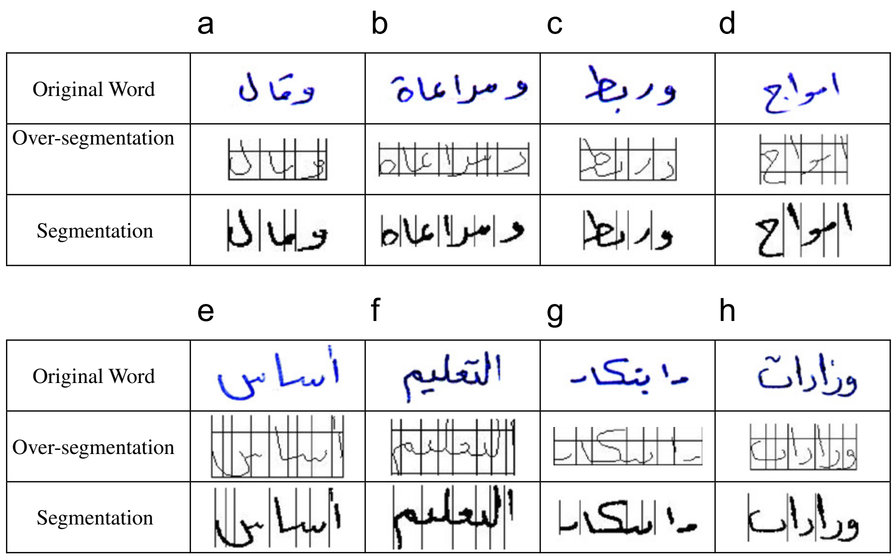


Fig. 43. Sample word images segmented by the feature-based Arabic heuristic segmentor (a–d) successful words, (e–h) unsuccessful words.

Table 8
Neural-based segmentation technique results, using SPV fusion.

Result	Correctly identified		Incorrectly identified	
	Valid	**Invalid**	**Valid (Bad)**	**Invalid (Missed)**
Subtotals	1908	730	240	471
%	56.97%	21.80%	7.17%	14.06%
Total	**2638**		**711**	
%	**78.77%**		**21.23%**	

Table 9
Neural-based segmentation technique results, using RCV fusion.

Result	Correctly identified		Incorrectly identified	
	Valid	**Invalid**	**Valid (bad)**	**Invalid (missed)**
Subtotals	2334	445	416	154
%	69.69%	13.29%	12.42%	4.60%
Total	**2779**		**570**	
%	**82.98%**		**17.02%**	

Table 10
Neural-based segmentation technique results for ten writers, using RCV fusion.

Writer	SP	Correctly identified			Incorrectly identified		
		Valid	**Invalid**	**Total**	**Valid**	**Invalid (bad)**	**Total (missed)**
Writer (1)	348	**261**	36	297	34	17	**51**
		75.00%	10.34%	85.34%	9.77%	4.89%	**14.66%**
Writer (2)	332	**216**	53	269	49	14	**63**
		65.06%	15.96%	81.02%	14.76%	4.22%	**18.98%**
Writer (3)	387	**255**	63	318	60	9	**69**
		65.89%	16.28%	82.17%	15.50%	2.33%	**17.83%**
Writer (4)	309	**231**	39	270	29	10	**39**
		74.76%	12.62%	87.38%	9.39%	3.24%	**12.62%**
Writer (5)	302	**213**	34	247	33	22	**55**
		70.53%	11.26%	81.79%	10.93%	7.28%	**18.21%**
Writer (6)	353	**234**	50	284	49	20	**69**
		66.29%	14.16%	80.45%	13.88%	5.67%	**19.55%**
Writer (7)	312	**222**	41	263	40	9	**49**
		71.15%	13.14%	84.29%	12.82%	2.88%	**15.71%**
Writer (8)	350	**245**	39	284	38	28	**66**
		70.00%	11.14%	81.14%	10.86%	8.00%	**18.86%**
Writer (9)	355	**236**	48	284	59	12	**71**
		66.48%	13.52%	80.00%	16.62%	3.38%	**20.00%**
Writer (10)	301	**221**	42	263	25	13	**38**
		73.42%	13.95%	87.38%	8.31%	4.32%	**12.62%**

characters had been removed by filter algorithm. This noise was not very beneficial for training of neural network.

Refer to Fig. 40; the first feature extraction technique which was tested was the *Plotchar* feature. Initially epoch numbers were set to size of 200 iterations, training error was 19.19%, and a low test set recognition rate of 75.65% was obtained. As a test, the epoch numbers were increased to 1000 epochs. The following step was modification, the top recognition rate on the test set moved up slightly to 76.77%. Therefore, what could be observed was an increase in recognition accuracy upon increasing the number of Epochs (iterations). Using the same configuration, but instead of using the modified direction feature MDF for input vector creation, top recognition rates of 82.26% and low recognition rates of 78.55% were obtained with 300 and 1000 epochs, respectively.

As is quite often true with neural classifiers, an increase in the number of iterations for training gives higher accuracy for *Plotchar* feature. This was seen to be the case with the recognition experiments as shown in Fig. 40. Conversely, training gives higher accuracy for MDF win decreasing number of iterations. However, the random weight that generated during training the classifier was a main reason of these variances. One final observation with respect to the back-propagation network was that the recognition rate will be tended to be highest when a larger number of patterns are used for training as seen in the literature.

Since each vector in *Plotchar* algorithm has 35 input features, and the MDF algorithm has 120 input features in each vector, the *Plotchar* needs less CPU time than MDF for training each of SP and characters as shown in Fig. 41.

4.3. Analysis of word segmentation results

The Analysis of the observed results for segmentation is fairly general. This means that the discussion does not dwell on particular experiments, rather it discusses the overall findings. Until recently, most handwritten Arabic segmentation procedures imbedded as part of word recognition techniques, have not been evaluated in terms of performance or accuracy. Instead, the performance has been measured in terms of the entire word recognition system. In most of papers, the researchers spoke of segmentation modules that have already or may be integrated into a word recognition system at a later time. The analyses that will be presented here are divided into two sub-sections. The first details experiments analyses for the *Plotchar* feature extraction techniques. The second section deal with describing experiments analyses for the modified feature extraction MDF techniques.

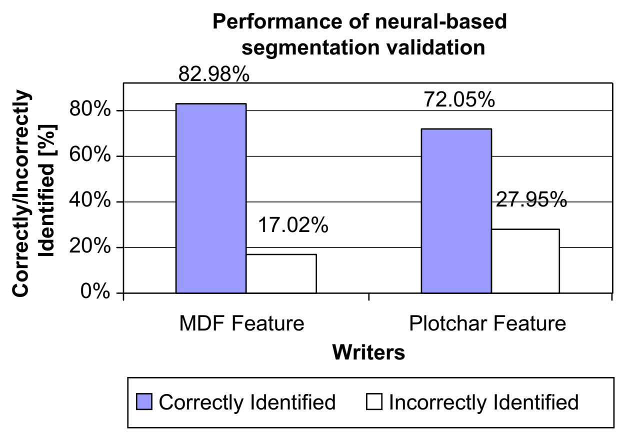


Fig. 44. Performance of neural-based segmentation validation between MDF and *Plotchar* feature using RCV fusion.

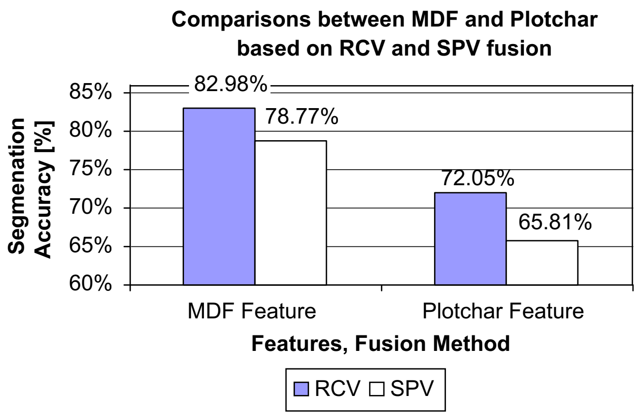


Fig. 45. Comparisons between MDF and Plotchar based on RCV and SPV fusion.

4.3.1. *Plotchar feature analysis*

As may be seen from Tables 1 and 5, the results using SPV fusion, the segmentation technique was successful for discarding bad segmentation points, from 13.71% to 6.18% that indicate to incorrectly segmenting of valid point, the technique wasn't successful for discarding or keep missed segmentation points, it was increased from 0.27% to 28.01% that indicates to incorrectly segmenting of invalid point, the over-all implemented system achieved segmentation accuracy was 65.81%, which is not promising result. Furthermore, the results also showed that the Arabic heuristic segmentor was able to produce better inputs to increase overall segmentation results, but using *Plotchar* feature have not assist reaching to the good results. Also, and as may be seen from Tables 1 and 6 using RCV fusion, the segmentation technique unsuccessful for discarding or keep bad segmentation points, it was increased from 13.71% to 20.36% that indicate to incorrectly segmenting of valid point. Also, the technique was not successful for discarding or keeps missed segmentation points, it was increased from 0.27% to 7.58% that indicate to incorrectly segmenting of invalid point, the over-all implemented system achieved segmentation accuracy was 72.05%, which is a promising result but there is better. Furthermore, the results also showed that the Arabic heuristic segmentor was able to produce better inputs to increase overall segmentation results.

4.3.2. *Modified feature extraction (MDF) analysis*

As may be seen from Tables 1 and 8 using SPV fusion, the segmentation technique was successful for discarding bad segmentation points from 13.71% to 7.17%. That indicates incorrectly segmenting of valid points, also, the technique wasn't successful in preventing missed segmentation points, and it was increased from 0.27% to 14.06% that indicates incorrectly segmenting of invalid point. The over-all implemented system achieved segmentation accuracy was 78.77%, which is not promising result. Furthermore, the results also showed that the Arabic heuristic segmentor was able to produce better inputs to increase overall segmentation results, but using *Plotchar* feature have not assist reaching to better

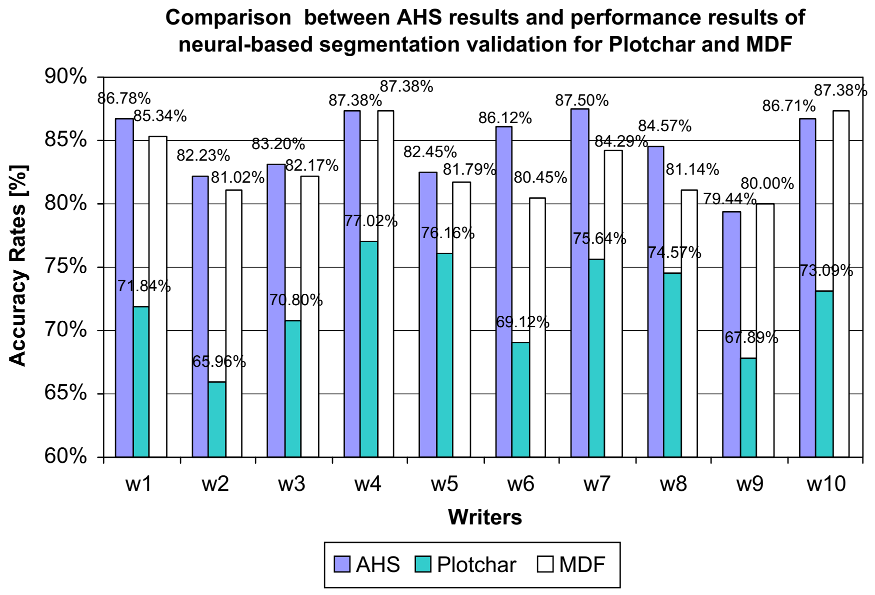


Fig. 46. Comparison between AHS results and results of performance of neural-based segmentation validation for (*Plotchar* and MDF).

Table 11
Comparison of segmentation results in the literature.

Author reference	Accuracy (%)	Language	Year	Data set used	Method used	Reference
Myer Blumenstein	75.28	Cursive English handwriting	2000	CEDAR database	Neural-based heuristic	[2]
Hamid and Haraty	69.72	Arabic handwriting	2001	Local database (360 addresses or 4000 words)	Neural-based heuristic	[5]
Nicchiotti et al.	86.9	Cursive English handwriting	2000	CEDAR database	Heuristic algorithm	[19]
Blumenstein et al.	85.74 (Testing 1031 from 1718 SP)	Cursive English handwriting	2005	CEDAR database	Neural-based heuristic	[24]
Han and Sethi	85.7	Cursive English handwriting	1995	50 real mail envelopes	Heuristic algorithm	[27]
Jawad et al.	85 (Sub-words segmentation)	Arabic handwriting	2008	Local database (200 images)	Heuristic Word Segmentation	[28]
Blumenstein and Verma	81.21	Cursive English handwriting	1997	Griffith University database	Neural-based conventional method	[29]
Eastwood et al.	75.9	Cursive English handwriting	1997	CEDAR database	Neural-based method	[30]
Lee et al.	90	Printed English handwriting	1996	alphanumeric characters	Neural-based method	[31]
Husam Al Hamad et al.	**82.98**	**Arabic handwriting**	**2008**	**Local database (500 words)**	**Neural-based method**	**[this research]**

results. Also, and as may be seen from Tables 1 and 9, and using RCV fusion, the segmentation technique was little successful in reducing bad segmentation points from 13.71% to 12.42%. This is an indication of incorrect segmentation of valid points. The over-all implemented system achieved segmentation accuracy was 82.98%, which is a very promising result. Furthermore, the results also showed that the Arabic heuristic segmentor was able to produce better inputs to increase overall segmentation results.

As mentioned previously, the results obtained for segmentation may not be easily compared with other researchers in the literature, as in most cases segmentation accuracy for Arabic handwritten words is either not measured at all, or is measured with respect to the number of correct or incorrect segmentations found. In other words, there have not been many researchers who have tested techniques for verifying segmentation points in over-segmented Arabic words. Table 11 summarizes the results obtained by various researchers compared with the proposed segmentation technique.

5. Conclusions and future research

5.1. Conclusions

The feature-based Arabic heuristic segmentor performed very well in the experiments presented in this research. Its objective was to correctly over-segment handwritten Arabic character. It was able to perform this task with minimal error. Upon comparison with other researchers, the error for the missed segmentation criterion was favorable being within approximately 0.27% and the over-segmentation rate was also comparable within approximately 1.46%. The main drawback of the proposed technique was the presence of bad segmentations. This however was attributed to the absence of a scheme to construct a path to separate connected characters with minor disfigurations. The 13.71% bad segmentation rate seems excessive. The reason they were labeled as "bad" was because the criterion demanded a very neat character boundary separation and great difference with characters from one to another. The modified direction feature extraction technique with 120 inputs for character recognition was compared to *Plotchar* feature extraction technique with 35 inputs. Upon comparison, it was found that it produces good acceptable recognition accuracy; the feature extraction techniques were tested on real world handwritten data by ten different writers that were used to train and test two neural classifiers. The results for handwritten character classification were favorable, especially considering the difficulty of the benchmark database set. In this research, the character recognition rates were obtained employing back-propagation networks. A recognition accuracy of 82.26% obtained was very favorably. An integral part of the segmentation process in this research was the segmentation point validation technique. It was developed to remove invalid segmentation points and retain valid ones to increase overall segmentation accuracy. The final results obtained for segmentation points validation using MDF feature and RCV fusion is 82.98%. The results are were very favorable and compared well with other researchers in the literature. The accuracy of the entire segmentation technique, encompassing the heuristic segmentor and segmentation point validation, was measured in terms of specific criteria set out in the literature. Overall, this segmentation accuracy was quite favorable.

5.2. Future research

The research presented in this paper has produced a number of useful observations and hence some possible directions that may be followed to improve word segmentation rates. The proposed over-segmentation technique proved to be successful in determining accurate vertical segmentations through feature inspection. However, the problem that was found to be most prominent was that of bad segmentations. It was concluded that a large proportion of these errors occurred due to interfering marks and strokes. These problems may be overcome in the future by the proposal of a scheme to construct segmentation paths for character separation. By improving on character recognition rates, an improvement shall most likely result. Character recognition accuracy was found to be very important for segmentation. Many strategies shall be attempted to raise recognition rates as mentioned previously. A better preprocessing methodology shall be used. It may include the introduction of a character slant correction module. This was not applied to the characters in this paper as it was assumed that slant correction applied to the entire word would be sufficient. Other strategies may include are increasing size of training set at least two-fold or three-fold for improving accuracy of overall segmentation processes. Finally, employing a better neural classifier and examination of different classification techniques shall also be explored i.e. statistical classifiers and other neural-based classifiers.

References

[1] R. Plamondon, S.N. Srihari, On-line and off-line handwriting recognition: a comprehensive survey, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (2000) 63–84.

[2] M. Blumenstein, Intelligent techniques for handwriting recognition, Ph.D. Dissertation, School of Information Technology, Griffith University-Gold Coast Campus, Australia, 2000.

[3] Ethnologue: Languages of the World, fourteenth ed, SIL International, 2000.

[4] L. Lorigo, V. Govindaraju, Off-line Arabic handwriting recognition: a survey, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (5) (2006) 712–724.

[5] A. Hamid, R. Haraty, A neuro-heuristic approach for segmenting handwritten Arabic text, ACS/IEEE International Conference on Computer Systems and Applications, AICCSA 1 (2001) 1–10.

[6] A. Nouh, A. Sultan, R. Tolba, An approach for Arabic characters recognition, Journal of Engineering Science, University of Riyadh 6 (1980) 185–191.

[7] B. Parhami, M. Taraghi, Automatic recognition of printed farsi texts, Pattern Recognition 14 (1981) 395–403.

[8] A. Amin, G. Masini, Machine recognition of multifont printed Arabic texts, in: Proceedings of International Conference on Pattern Recognition, Paris, France, 1986, pp. 392–395.

[9] T.S. El-Sheikh, R.M. Guindi, Computer recognition of Arabic cursive scripts, Pattern Recognition 21 (1988) 293–302.

[10] S. Sami El-Dabi, R. Ramsis, A. Kamel, Arabic character recognition system: a statistical approach for recognizing cursive typewritten text, Pattern Recognition 23 (1990) 485–495.

[11] A. Ymin, Y. Aoki, On the segmentation of multi-font printed uygur scripts, in: Proceedings of 13th International Conference on Pattern Recognition, vol. 3, 1996, pp. 215–219.

[12] B. Al-Badr, R. Haralick, A segmentation-free approach to text recognition with application to Arabic text, International Journal on Document Analysis and Recognition 1 (1998) 147–166.

[13] J. Alherbish, R.A. Ammar, M. Abdalla, Arabic character recognition in a multi-processing environment, in: Proceedings of IEEE Symposium on Computers and Communications, Alexandria, Egypt, 1997, pp. 286–292.

[14] M.S. Khorsheed, W.F. Clocksin, Structural features of cursive Arabic script, in: Proceedings of British Machine Vision Conference, 1999, pp. 422–431.

[15] L. Hamami, D. Berkani, Recognition system for printed multi-font and multisize Arabic characters, The Arabian Journal for Science and Engineering 27 (2002) 57–72.

[16] S.A. Al-Qahtani, M.S. Khorsheed, A HTK-based system to recognise Arabic script, in: Proceedings of 4th IASTED International Conference on Visualization, Imaging, and Image Processing. ACTA Press, Marbella, Spain, 2004.

[17] S.N. Srihari, G. Ball, An assessment of Arabic handwriting recognition technology, in: IWFHR, CEDAR Technical Report TR-03-07, 2007.

[18] X. Fan, B. Verma, Segmentation vs. non-segmentation based neural techniques for cursive word recognition: an experimental analysis, International Journal of Computational Intelligence and Applications 2 (4) (2002) 377–384.

[19] G. Nicchiotti, C. Scagliola, A simple and effective cursive word segmentation method, in: Proceedings of the 7th International Workshop on Frontiers in Handwriting Recognition, Amsterdam, 2000, pp. 499–504.

[20] B. Yanikoglu, P.A. Sandon, Segmentation of off-line cursive handwriting using linear programming, Pattern Recognition 31 (1998) 1825–1833.

[21] L. Lorigo, V. Govindaraju, Segmentation and pre-recognition of Arabic handwriting, in: The 8th International Conference on Document Analysis and Recognition, ICDAR 2005, August 29–September 01, 2005, Seoul, Korea, 2005.

[22] M. Blumenstein, X.Y. Liu, B. Verma, An investigation of the modified direction feature for cursive character recognition, Pattern Recognition 40 (2) (2007) 376–388.

[23] A. Hamid, R. Haraty, Segmenting handwritten Arabic text, ACIS International Journal of Computer and Information Science, IJCIS 3 (4) (2002).

[24] C.K. Cheng, M. Blumenstein, The neural-based segmentation of cursive words using enhanced heuristics, in: ICDAR, 8th International Conference on Document Analysis and Recognition, vol. 5, 2005, pp. 650–654.

[25] M. Blumenstein, X.Y. Liu, B. Verma, A modified direction feature for cursive character recognition, in: Proceedings of the International Joint Conference on Neural Networks, Budapest, Hungary, vol. 4, 2004, pp. 2983–2987.

[26] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning internal representations by error propagation, Parallel Distributed Processing 1 (1986) 318–362.

[27] K. Han, I.K. Sethi, Off-line cursive handwriting segmentation, in: Proceedings of the 3rd International Conference on Documents Analysis and Recognition, 1995, pp. 894–897.

[28] J.H. AlKhateeb, J. Jiang, J. Ren, S.S. Ipson, Component-based segmentation of words from handwritten Arabic text, in: Proceedings of World Academy of Science, Engineering and Technology, ISSN, vol. 31, 2008, pp. 1307–6884.

[29] M. Blumenstein, B. Verma, A segmentation algorithm used in conjunction with artificial neural networks for the recognition of real-world postal addresses, in: International Conference on Computational Intelligence and Multimedia Applications, ICCIMA, Gold Coast, vol. 97, Australia, 1997, pp. 155–160.

[30] B. Eastwood, A. Jennings, A. Harvey, A feature based neural network segmentor for handwritten words, in: Proceedings of the International Conference on Computational Intelligence and Multimedia Applications, ICCIMA, Gold Coast, Australia, vol. 97, 1997, pp. 286–290.

[31] S.-W. Lee, D.-J. Lee, H.-S. Park, A new methodology for gray-scale character segmentation and recognition, IEEE Transaction on Pattern Analysis and Machine Intelligence 18 (10) (1996) 1045–1051.

About the Author—HUSAM A. Al HAMAD Assistant Professor of Computer Science at Qassim University, Qassim, Saudi Arabia. BS, MS, and Ph.D. in computer science in 2000, 2002, and 2008, respectively. His research interests are image processing and artificial intelligence, and neural network.

About the Author—RAED ABU ZITAR BS, MS, Ph.D. in computer engineering 1988, 1989, and 1993, respectively. Professor at New York Institute of technology, Amman, Jordan, branch. Research interests include image processing, pattern recognition and artificial intelligence.