Pattern Recognition 46 (2013) 141–154

Contents lists available at SciVerse ScienceDirect

Pattern Recognition

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

Arabic handwriting recognition using structural and syntactic pattern attributes

Mohammad Tanvir Parvez [a,*], Sabri A. Mahmoud [b]

[a] Computer Engineering Department, Qassim University, Qassim 51477, Saudi Arabia
[b] Information and Computer Science Department, King Fahd University of Petroleum & Minerals (KFUPM), Dhahran 31261, Saudi Arabia

ARTICLE INFO

Article history:
Received 14 December 2011
Received in revised form
9 June 2012
Accepted 18 July 2012
Available online 31 July 2012

Keywords:
Arabic handwriting recognition
Structural recognition
Arabic OCR
Nearest neighbors
Median computation

ABSTRACT

In this paper, we present research results on off-line Arabic handwriting recognition using structural techniques. Statistical methods have been more common in the reported research on Arabic handwriting recognition. Structural methods have remained largely unexplored in this regard. However, both statistical and structural techniques can be effectively integrated in multi-classifier based systems. This paper presents, to our knowledge, the first integrated offline Arabic handwritten text recognition system based on structural techniques. In implementing the system, several novel algorithms and techniques for structural recognition of Arabic handwriting are introduced. An Arabic text line is segmented into words/sub-words and dots are extracted. An adaptive slant correction algorithm that is able to correct the different slant angles of the different components of a text line is presented. A novel segmentation algorithm, which is integrated into the recognition phase, is designed based on the nature of Arabic writing and utilizes a polygonal approximation algorithm. This is followed by Arabic character modeling by 'fuzzy' polygons and later recognized using a novel fuzzy polygon matching algorithm. Dynamic programming is used to select best hypotheses of a sequence of recognized characters for each word/sub-word. In addition, several other key ideas, namely prototype selection using set-medians, lexicon reduction using dot-descriptors etc. are utilized to design a robust handwriting recognition system. Results are reported on the benchmarking IfN/ENIT database of Tunisian city names which indicate the robustness and the effectiveness of our system. The recognition rates are comparable to multi-classifier implementations and better than single classifier systems.

1. Introduction

Handwriting recognition is an active research area in Pattern Recognition and has many practical applications. Some of these applications include postal address and zip code recognition, forms processing, automatic processing of bank cheques etc.

Recognition of unconstrained handwriting is an extremely challenging task due to large variations in handwriting styles. In some restricted domains, the task of handwriting recognition may become less demanding. For example, in zip code recognition, the problem may reduce to the recognition of a sequence of isolated numerical digits. However, even in this restricted domain, many challenging issues arise. Touching digits, broken characters, writing errors and noise due to scanning are some of the issues that can make the 'simpler' problem of zip code recognition much harder.

Over the last few decades, numerous research results have been reported on handwriting recognition, especially for Latin script. Although there are very good results for machine-printed text recognition (with more than 99% accuracy), accuracies on handwritten text recognition lag far behind. For example, for the well-known IAM handwritten text database for Latin script [32], the state of the art word recognition accuracy is around 74% [21]. The state of the art for Arabic handwritten text recognition falls far behind compared to Latin script. There is very few work published on unconstrained Arabic handwritten text recognition [29]. Most of the work on Arabic handwriting recognition has dealt with character, digit or word recognition [13,29]. Thus a lot of work remains to be done in Arabic handwriting recognition.

Recently, comparatively large number of work has been published for Arabic handwritten word recognition, partially due to the availability of the IfN/ENIT database of Tunisian city names [43]. Most of these works are hidden Markov model (HMM) based Arabic handwritten word recognition [6,12,16,23,25,27,33,35,36,37,48]. Although HMM is popular for cursive text recognition, some researchers have used HMM for Arabic numerals recognition [4,31]. However, only one group of researchers has

* Corresponding author. Tel.: +966 551895733.
E-mail addresses: m.parvez@qu.edu.sa, tparvez@gmail.com (M. Tanvir Parvez), smasaad@kfupm.edu.sa (S.A. Mahmoud).

0031-3203/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.patcog.2012.07.012

reported results on Arabic handwritten text recognition [46]. As Lorigo and Govindaraju [29] pointed out in their survey paper, there still remains a gap in research for developing OCR systems that works on full Arabic text page (not just on isolated words/digits).

There are some other works on Arabic handwritten word recognition that utilize classifiers other than HMM, like Support Vector Machines (SVM) [5,10], Recurrent Neural networks (RNN) [22], Transparent Neural Networks [9].

HMM based approaches for Arabic handwritten word recognition have attempted to modify the basic configuration of HMM to achieve higher recognition accuracy. Below we discuss several such techniques reported recently in the literature [12,25,35]. Reference may be made to [41] for a recent and comprehensive survey on off-line Arabic Handwritten Text Recognition.

Mohamad et al. [35] used two HMM slanted windows in addition to the traditional HMM vertical window. One window is slanted to the right, one vertical, and one to the left. They proposed these sliding windows to cope with the problem of writing inclination, overlapping ascenders and descenders and shifted position of diacritical marks. The authors combined the three HMM-based classifiers at the decision level using three combination schemes: sum rule, majority vote and multilayer perceptron. The proposed method was tested on IfN/ENIT database. The authors studied different system parameters, combinations of features, performance of different combination schemes. With MLP being the combination scheme, a recognition rate of 90.26% was reported with combined classifiers for set-*d* of IfN/ENIT (the number of classes is 946).

Kessentini et al. [25] used multi-stream hidden Markov models for off-line handwritten word recognition. In this technique several different feature representations are modeled and decoded separately by individual HMM classifiers. The use of multi-stream HMMs allows the merging of different independent feature types, some weighting may be used for the different feature types based on its effectiveness, and different HMM models may be used for the different feature types. The authors reported an accuracy of 79.6% for set-*e* of IfN/ENIT.

Kundu et al. [27] used Variable Duration HMM (VDHMM) where all Arabic words are modeled by one HMM. Each character is a state in VDHMM and has a variable duration to model a character model of multiple segments. In [12], Dreuw et al. used a writer adaptive training using constrained maximum likelihood linear regression (CMLLR). The authors tested their method on IfN/ENIT database and reported recognition accuracies of 94.18% and 88.78% for set-*d* and set-*e* respectively.

Thus, there are an increasing number of efforts towards multi-stream or multi-classifiers systems for improved Arabic word recognition. However, structural approaches for Arabic handwritten word recognition is almost absent in this picture. In statistical approaches (like HMM, SVM, ANN, etc.), each object or part of an object (here characters or words) is represented by some fixed length values called feature vectors. In structural approaches, the spatial relationships between parts of an object are used to represent the object. Graphs, strings or trees are some of the ways to represent the structural relationships of an object. The power of the structural approach is that it can model complex objects and the need for fixed length feature vectors is relieved. However, there are fewer machine learning tools available for structural approach compared to statistical approach.

Developing structural methods for Arabic handwriting recognition can be useful for the following reasons:

- There are a number of efforts to incorporate structural information into the recognition system based on statistical methods to improve recognition accuracy [1,37,46]. One such example is to use structural information to decide between confusing classes (like zero and five in Arabic digit recognition) [1]. Thus there is a growing need for better structural features/representations to be used in recognition systems.
- Another aspect is the advancement of research in developing machine learning tools for structural representations. One such example is the concept and algorithms for computing median graphs [20]. Another example is the mapping of structural information into the statistical feature space using graph embedding [45]. With this advancement in research, a new horizon for developing structural techniques for recognition of Arabic handwriting is opening up.
- Multi-classifiers systems have been shown to give better results compared to single classifier system for Arabic handwriting recognition [15,25,35]. The basic goal of a multi-classifiers system is to utilize multiple sources of information to improve the performance. Many researchers have modified their basic recognition systems to get multiple sources of information for better recognition accuracy. Examples of such modifications include slanted frame HMM classifiers for handling misplaced dots/diacritics [35], contextual models for dealing with overlapping characters [17], etc. However, exploring the spatial relationships in Arabic handwriting is expected to be more robust for handling these issues. Therefore, structural approaches can provide valuable clues for improving accuracies for Arabic handwriting recognition in multi-classifiers systems.

We adopt the structural approach for Arabic handwriting recognition due to the above mentioned reasons. This paper describes the different techniques developed for Arabic handwriting recognition based on the structural attributes of the text and the syntactic relations between different parts in an Arabic word. The main contributions of this paper can be summarized as follows:

- Structural modeling of Arabic handwriting based on fuzzy polygonal approximation of Arabic text contours. The resulting models, called Fuzzy Attributed Turning Functions (FATF), tolerate variations of the handwritten text of different writers. FATF also handles (to some extent) non-uniform slants present in a word/sub-word.
- A robust Arabic character segmentation algorithm utilizing the approximated polygons of Arabic text. This algorithm does not suffer from the limitations of segmentation algorithms based on vertical projection, which is not suited for handwritten Arabic cursive text. In addition, it does not suffer from the effects of noise as is the case with the algorithms based on Arabic text contours or chain codes.
- A slant detection and correction algorithm that is adapted to the slant at the word/sub-word level. Hence, slant variation in a text line are detected and corrected. Since it is based on the polygonal approximation, it has higher resolution that the algorithms based on the chain code.
- A classification phase based on fuzzy polygon matching algorithm with an integrated segmentation algorithm of Arabic text. Hence, Segmentation and recognition phases are both carried out simultaneously.
- An Integrated lexicon reduction technique within the classification cycle. Hence, the extracted information is utilized for both recognition of words and reduction of lexicon.
- Reduction of character models by prototype selection. The simple concept of set-medians is shown to be extremely effective by incorporating into it a fuzzy description of the character models.
- Novel implementations of several techniques including dot extraction and assignments, normalization of Arabic characters etc. to suit Arabic handwriting modeling and recognition.

Several other algorithms are described in connection with these contributions. We believe that this paper introduces several key ideas for modeling and recognition of Arabic handwriting using structural techniques. It is not only that the techniques reported in this paper are novel, but also the experimental results on IfN/ENIT database are found very promising.

It is to be noted that our work in [39] introduced our polygonal approximation technique which was tested on popular shapes and Arabic isolated characters. The work in [40] introduced the fuzzy attributed turning function which was tested using Arabic isolated characters. None of the previous publications addressed Arabic cursive text recognition, the segmentation of Arabic characters, the slant correction of Arabic text, the lexicon reduction technique or prototype selection, to name a few.

The rest of the paper is organized as follows. Section 2 gives an overview of the characteristics of Arabic script. Section 3 deals with the overall recognition process. Sections 4 and 5 discuss the details of different phases of Arabic handwriting recognition. Experimental results are presented in Section 6. Finally, the conclusions are summarized in Section 7.

2. Characteristics of Arabic script

Arabic alphabet is the script used for writing several languages of Asia and Africa, such as Arabic, Persian and Urdu. Arabic text is cursive in both machine-printed and hand-written text and is written from right to left, with the alphabet having 28 basic characters. Sixteen Arabic letters have from one to three dots. The number and position of the dots differentiate between the otherwise similar characters (like غ and ع). Additionally, some characters have a zigzag like stroke (Hamza ء) as in (أ, ؤ, ئ). These dots and Hamza are called secondaries and they are located above the character primary part as in Alef (أ), or below like Baa (ب), or in the middle like Jeem (ج). Within a connected component, some characters connect to the preceding and/or following characters, and some do not connect. The shape of an Arabic character depends on its position in the word. A character might have up to four different shapes depending on it being isolated, connected from the right (ending form), connected from the left (beginning form), or connected from both sides (middle form). Characters in a word may overlap vertically (even without touching). Arabic characters do not have fixed size (height and width). The character size varies according to its position in the word.

Arabic text has diacritics (short vowels) such as Fathah (ـَ), Dhammah (ـُ), Shaddah (ـّ), Maddah (ـٓ), and Kasrah (ـِ), Tanween which is formed by having double Fathah (ـً), Dhammah (ـٌ), or Kasrah (ـٍ). These diacritics are placed either on top of or below the characters. A different diacritic on a character may change the meaning of a word. Readers of Arabic are accustomed to reading un-diacritized text by deducing the meaning from the context.

Arabic text has ligatures. A ligature is a character formed by combining two or more letters with vertical or horizontal overlapping between component characters (like 'laam-alef' (لا) and 'laam-meem' (لم). In segmentation-based recognition, this property of overlapping may introduce difficulty in segmentation of words into characters. A solution to this problem is to consider ligatures as additional classes. This results in increasing the number of classes of Arabic handwriting text. Fig. 1 shows some characteristics of Arabic text and the difficulties that an Arabic text recognition system has to address in practical applications.

Many of these aspects make Arabic handwriting recognition task more difficult compared to Latin script. However, the variations between Latin printed and handwritten text is much more

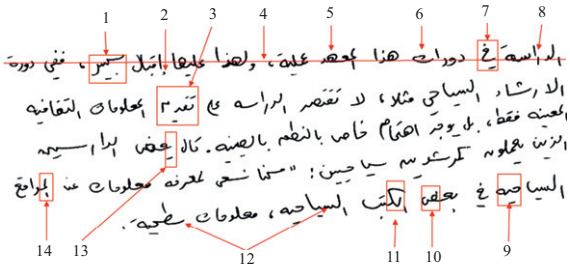

Fig. 1. Illustration of some characteristics of Arabic script. It is cursive and has right to left flow of writing: (1) a connected component having a first character (Kaf) very close to the writing line (the vertical stroke is small although it should be large), (2) a diacritic (Hamza), (3) two dots are written as a short stroke with missing character of the two dots, and (4) the approximate writing line. Different words writing- (base-) lines do not match the text line base line. Hence, the estimated base line may not match real base line of connected components. (5) A middle Hhaa written as beginning Hhaa, (6) another way of writing a Taa in one stroke including the two dots instead of a main body and two dots above it, (7) a ligature with a Yaa written as a reverse stroke, it is another way of writing an end Yaa, and (8) a Raa written like an Alef. A human without context cannot read it as a Raa. It is similar to the Alef (the first character of the same word). (9) A three character component with deformed structure. The first should have the lower stroke on the writing line and the upper stroke above writing line, the second should be on the writing line and the third should extend well above the writing line. The three are having the same height of the vertical projection. (10) A dot touching the main body of a character (Dhad) hence it may be recognized as Sad. (11) A word with 2nd character tilted to the right and the 3rd character tilted to the left. (12) Two ways of writing the same character, Seen, one with notches and the other without. (13) A dot written as two dots making the character a Yaa instead of a Baa. (14) A ligature and an Alef below it instead of being above the writing line.

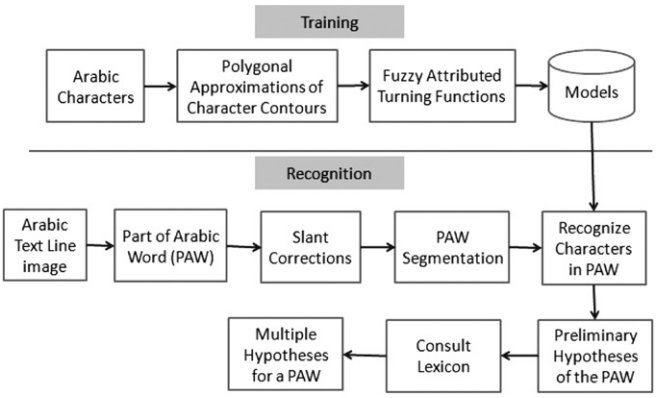

Fig. 2. Illustration of phases of the recognition of an Arabic handwritten word/sub-word.

than its Arabic counterparts. The assumption that Arabic is more difficult may be due to the fact that less effort and resources have been devoted to it and thus the state of the art is less advanced. However, in the past few years, there is an enormous increase in effort for Arabic handwriting recognition [13,41].

3. Overview of the Arabic handwriting recognition process

The recognition process, as we describe here, assumes that text lines are already extracted from pre-processed text pages using a suitable line segmentation algorithm. From each line of text, words/sub-words (called *Parts of Arabic Words*—PAWs) are extracted. For each PAW, the recognition system tries to identify the best character sequences comprising that PAW.

Fig. 2 illustrates the various phases of Arabic handwriting recognition. Our approach for recognition of an Arabic text line is

to recognize its PAWs. A PAW is recognized based on the recognition of its component characters. Therefore, in the training phase, we generate character models from Arabic handwritten characters. These models are based on Fuzzy Attributed Turning Functions (FATF) as will be discussed below. There are several reasons to take this approach as is evident from Fig. 1. The text line estimated baseline does not necessarily coincide with individual PAWs baselines. The estimation of the tilt (slant) of a text line assumes that the tilt is the same for all PAWs although some writers write different PAWs with different tilt. Although this is not the way, in general, that other researchers handle Arabic text recognition, we believe that addressing the recognition of Arabic text at the PAW level is more accurate than addressing it at the text line level. This is the approach we follow in this work.

The recognition of Arabic handwritten text consists of several steps. PAWs are extracted from the text lines, then they are slant corrected. Then we extract the main body of a PAW and segment it into smaller components. These smaller components may be valid Arabic characters or parts of Arabic characters (in the case of over-segmentation). Then the best segmentation of the PAW and its constituent characters are identified by an adaptive algorithm. We generate multiple hypotheses for each PAW. These hypotheses pass through post-processing steps, like lexicon consultation, to re-rank the hypotheses and select the best matching word. The following sections describe these phases in more details.

4. Preprocessing of Arabic text

In the preprocessing phase, assuming we have segmented Arabic text lines, we first extract the words/sub-words from the Arabic text line. These extracted words/sub-words are slant corrected. Then the slant corrected Arabic words/sub-words are segmented into possible candidate character segments. These candidate segments are later recognized and combined to form the Arabic words. The following sub-sections address these phases of preprocessing of Arabic text line.

4.1. Extraction of words/sub-words from Arabic text lines

In this section, we describe our method for extracting words/sub-words from a line of Arabic text. The goal of word/sub-word extraction is to extract *Parts of Arabic Words* (*PAWs*), where each PAW consists of one primary component (the main body) and associated dots/diacritics (the secondary bodies). For example, the Arabic word لدراسة contains four PAWS.

The proposed method for extraction of PAWs consists of several steps. We first extract all the connected components in the line. Then a rough baseline is estimated using an expectation-maximization (EM) algorithm. This knowledge of the baseline is used to locate the primary components of the PAWs. All other components are called secondary components. Each of the secondary components is assigned to a primary component. Each primary component with the associated secondary components is marked as a PAW. We now describe the above steps for extracting the PAWs in details.

4.1.1. Extraction of blobs

The connected components (blobs) of the binary image of the text line are extracted. For each blob i, we obtain the following information: centroid of the blob: $C_i=(Cx_i, Cy_i)$, bounding box containing the blob, $Box_i=(XTop_i, XWidth_i, YTop_i, YHeight_i)$, the area of the blob, $Area_i$ and the pixel list of the blob.

4.1.2. Identification of base components

The next step of our algorithm is to estimate the baseline. To estimate the baseline, we first identify *base components*. Assume that there are n blobs in a text line. Let $Area_{avg}$ be the average area of all the blobs, where$Area_{avg}=(1/n)\sum_{i=1}^{n} Area_i$. Now, a blob i is marked as a *base component* if $Area_i \geq \frac{1}{2}Area_{avg}$.

4.1.3. Baseline detection

Once the base components in a line are identified, the baseline is detected using an expectation-maximization (EM) procedure. First, a best-fitting line is obtained by linear regression using the centroids of the base components. This best-fitting line is taken as the initial estimation of the baseline. This initial estimation of the baseline is refined. We call a base component as a *refined base component* if the boundary box of that component is crossed by the initial baseline. The centroids of these refined base components are used to estimate the final baseline. Again, we use linear regression to find a best-fitting line passing through the centroids of the refined base components. This regression line is taken as the baseline of the text-line.

4.1.4. Identification of primary components

Once the baseline is estimated, the primary components of the line are identified from the base components. Base components are basically the larger components in the line. However, some isolated characters (like ا) may not be marked as a base component. On the other hand, some dots/diacritics (like Shaddah ّ) may be marked as base components. Therefore, we need to consider these two cases while identifying the primary components. Thus, the primary components are identified in two passes.

In the first pass, we check each base component i to see whether component i is possibly a dot or diacritic. Let d be the distance of the centroid of component i from the estimated baseline. Let $Area_i$ be the area of the component i. Then, component i is not a primary component if $Area_i < Area_{avg}$ and $d > n_1^* S_{Thickness}$. Otherwise, component i is marked as a primary component. Here, $S_{Thickness}$is the stroke thickness of the line. $S_{Thickness}$is estimated as the most frequent value in the vertical projection profile of the line image [36]. Constant n_1 is estimated experimentally as shown below.

In the second pass, the primary components selected in the first pass are refined. The idea is to find those components which are located above or below other larger components. This is done by computing the overlapping of the bounding boxes of the blobs.

4.1.5. Assignment of secondary components

Assume that the component i is not marked as a primary component in the previous step. For this component i, we calculate the overlapping with all the primary components. Component i is assigned to the primary component j where overlapping is the maximum over all the primary components. If there are more than one primary components with the same overlapping with i, then component i is assigned to that primary component whose centroid is closest to the centroid of i. We then mark i as a secondary component.

Note that, there may be some components with no overlapping with any of the primary components. Assume that i is such a component. Here, component i is either a misplaced dots/diacritics or it is a small character (like isolated ه). Let d be the distance of the centroid of component i from the baseline. Let $Area_i$be the area of the component i. Then, component i is marked as a primary component if $Area_i \geq \frac{1}{2}Area_{avg}$ and $d < n_2^* S_{Thickness}$. Otherwise, i is assigned to the primary component whose centroid is closest to the centroid of i. Constant n_2 is estimated experimentally as shown below.

The parameters or heuristics of the above PAW extraction procedure are estimated by experimenting with the text database described in [26]. Text lines were extracted from 47 pages of Arabic text, where each page contains 15–19 lines and was written by different writers. The result of the PAW extraction algorithm was verified manually. The parameters of the algorithm which gave the best PAW extraction results were selected as the final estimated parameters. For example, n_1 and n_2 in the above description are both set to 3 through these experimentations.

Once all the primary components are identified and the secondary components are assigned to their respective primary components, each primary component with its associated secondary components forms one PAW.

4.2. Polygonal approximation of word contours

In many of the algorithms described in this paper, we work with the contour of an Arabic word. The contour is conveniently represented by a polygonal approximation [42]. In principle, any polygonal approximation algorithm can be used for this purpose. However, in this work we use the algorithm described in [39] due to its suitability and robustness for handwritten shapes. The idea of collinear-points suppression in [39] is used in several phases of our current system. Therefore, a brief description of the techniques in [39] follows.

The polygonal approximation (a.k.a. *dominant points* detection) algorithm [39] first selects an initial set of dominant points from the contour $C=P_i(x_i, y_i)$, $i=1, 2, \ldots, n$. These initial set of dominant points (called *break points*) are the points on the contour where the contour takes a turn. The approximation defined by the break points is affected by noise on the contour. This approximation can be improved by removing redundant break points. This removal process is called *constrained collinear-points suppression*. Fig. 3 illustrates the process. Here, three consecutive points P_i,P_j and P_k are considered at a time. Point P_j is suppressed if the perpendicular distance d_{per} from P_j to the line joining P_i and P_k is less than some threshold d_{col} and the following two conditions are satisfied: (i) in the triangle formed by P_i,$P_j$andP_k, the angles at P_i and P_k are acute, and (ii) for each unsuppressed point P_l other than P_i,P_j and P_k, the minimum distance from P_l to the line segment joining P_i and P_k is greater than d_{col}.

Constrained collinear-points suppression essentially removes redundant dominant points from contour. However, the best set of dominant points for a particular contour, which also defines the outline polygon, is selected using an optimization procedure. The optimization procedure selects the final set of dominant points by optimizing some performance measures of the fitted polygon. An illustration of the algorithm is given in Fig. 4. As can be seen in Fig. 4, the final approximation uses more dominant points where the contour has more details, whereas the straighter parts of the contour are approximated with less number of points. The local and global optimizations of the polygonal approximation algorithm make it more suitable for handwritten characters/words.

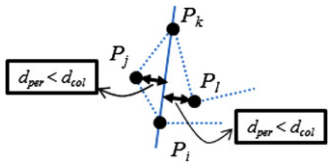

Fig. 3. Illustration of the process of constrained collinear-points suppression.

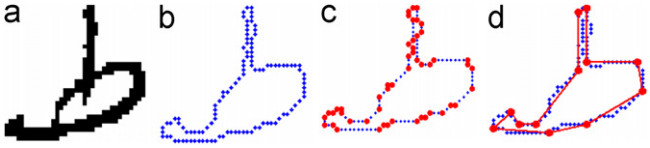

Fig. 4. Illustration of the polygonal approximation algorithm: (a) Arabic character Twaa (ط), (b) contour after smoothing, (c) break points of (b), (d) polygonal approximations of (b) after suppression of points.

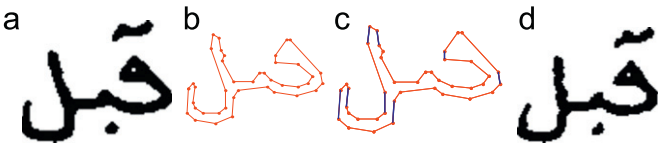

Fig. 5. Illustration of slant correction of an Arabic word: (a) original word image, (b) the polygonal approximation of the word main body, (c) the near-vertical strokes marked as thick lines and (d) final slant corrected word.

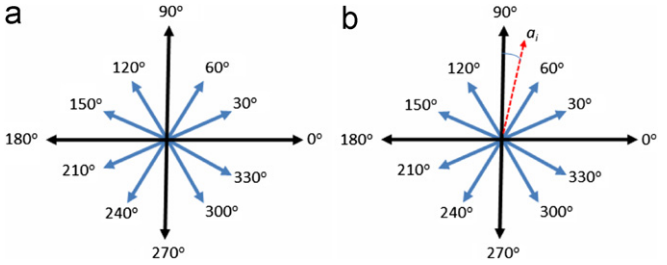

Fig. 6. Illustration of mapping an angle to a standard direction: (a) the set of standard directions and (b) an edge (dotted arrow) is being mapped to the closest standard direction (90°).

4.3. Slant correction

There may be some slant in the handwritten words (or PAWs). These slants should be corrected before the PAWs are recognized for higher recognition rates. Slant correction algorithms look for *near-vertical strokes* (NVSs) to estimate the slant angle. Then a shear transformation is applied to the PAWs text images to correct the slant. Here, we describe how the polygonal approximation based representation of an Arabic word lends itself to an intuitive slant correction algorithm. Fig. 5 illustrates the proposed method for slant correction of an Arabic word.

Assume that the word (or PAW) image I is a binary image. Let D_1,D_2,...,D_n be the sequence of n dominant points obtained for the contour C of the main body of I using the algorithm in [39]. These n dominant points define a polygonal approximation of the contour C with $n+1$ edges, $e_i=\langle D_iD_{i+1}\rangle$,$i=1,2,...,n$ where $D_{n+1}=D_1$. Let$a_i$be the counter-clockwise angle of the edge e_i with the x-axis. For each edge e_i, the angle a_i is mapped to a closest *standard direction* (see Fig. 6(b)). Standard directions virtually limit the directions of writing strokes and hence serve as a quantization of stroke angles.

To estimate the overall slant of the word image I, the NVSs in the image are identified. This is done as follows. Let SD_j,$j=1,2,...,k$ be the k standard directions. Let S be the standard direction to which angle a_i is mapped. Here, $S=SD_t$ where $t=\arg\min_{1\le j\le k}|SD_j-a_i|$. Now, edge e_i is marked as an NVS if it is mapped to a vertical standard direction. For example, assume that a set of standard directions is given by $SD_j=\langle 0,30,60,90,120,150,180,210,240,270,330\rangle$ (see Fig. 6(a)). Here each standard direction is separated by 30°. Then e_i is identified as a near-vertical stroke if the angle S is equal to either 90° or 270°. Fig. 6(b) illustrates this mapping of the angle a_i to a standard direction (here the closest direction is 90°).

Let l_i be the length of e_i. Then the slant $slant_{e_i}$ associated with e_i is estimated as: $slant_{e_i}=(|a_i-S|)\times l_i$. The overall slant angle of the word image is estimated as follows. Let e_t, $t=1,2,...,m$ be the m near-vertical strokes (NVSs) in the word image I. Then the overall

slant θ of the image is estimated as: $\theta = (\sum_{t=1}^{m} slant_{e_t})/(\sum_{t=1}^{m} l_t)$. Here, $slant_{e_t}$ and l_t are the slant and edge-length associated with the edge e_t. Once θ (in degree or radians) is calculated, the slant of the PAW image I is corrected by applying a shear transformation to it. The following shear transformation is applied to each pixel (x, y) of I:

$$x' = x - y \tan\theta, \quad y' = y$$

The slant correction algorithm described above corrects the slant uniformly across the word image. Non-uniform slat corrections [7,49] are expected to improve the performance. However, we use the simpler uniform slant correction for our system, since the recognition system itself can tolerate the slant present in the individual characters to some extent. This is discussed later in this paper. In addition, correcting slant at the word level is more effective for real data where different words may be written with different slants in the same line.

Once a PAW image is slant corrected, we segment the main body of the PAW and also extract the dot information (if any) present in the PAW image. We discuss the details of these steps below.

4.4. Segmentation of PAWs

In this section, we describe our method for segmenting the main body (primary component) of a PAW into characters/graphemes.

Our goal here is to segment a PAW into component characters or graphemes (part of a character). However, segmenting a word into characters is difficult for cursive text unless the word has been recognized already. Thus there is a circular dependency between segmentation and recognition, which is sometimes referred to as *Sayre's paradox* [47]. One of the approaches adopted by many researchers is to carry out the segmentation and recognition at the same time [21]. In this approach, a segmentation of a PAW is corrected and verified by a recognizer. In this work, we have adopted this approach.

Therefore, the segmentation obtained by our algorithm is refined and corrected by the recognition system. Occasionally, our algorithm may generate an over-segmentation of a PAW. However, the recognition process described in the next section takes care of any over-segmentation and finds out the best sequence of characters in a PAW. In addition, lexicon consultation phase may further improve any over- or under-segmentations of Arabic handwritten words.

Here again, we utilize the polygonal approximation of an Arabic word contour in a novel way to obtain the segmentation of the word. The basic idea is to locate segmentation points by finding '*valleys*' on the word contour (Fig. 7(a)). To avoid finding false valleys, collinear-points suppression technique [39] is used as a smoothing operator. In addition, several rules are used to find segmentation points not located in valleys (Fig. 7(b)).

We apply collinear-points suppression on the upper contour of the PAW's main body with an adaptive threshold. This threshold is selected adaptively in such a way that the smoothing operation gives us a polygonal approximation of the PAW's upper contour with very low resolution (very few dominant points). We then consider the dominant points only and locate the segmentation

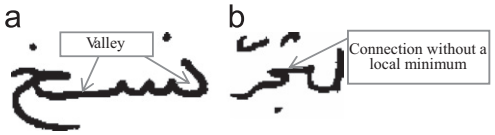

Fig. 7. Illustration of different cases in locating segmentation points: (a) characters connected on valleys (on the baseline) in the Arabic word (نسخ) and (b) characters are joined without a valley (above baseline).

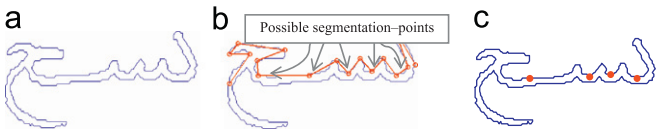

Fig. 8. Illustration of the procedure for obtaining segmentation points: (a) contour of the main body of Arabic word نسخ, (b) possible segmentation points (the circles) in the upper contour and (c) final segmentation points (in bullets) selected by the proposed technique.

points from among them. Fig. 8 illustrates the process for obtaining the segmentation points. The detail of this process is given below.

The following algorithm captures our novel method for segmenting a PAW.

Algorithm. *getCharacterSegmentation*

- Estimate the stroke-width of the PAW.
- If the stroke-thickness is greater than PAW width, then no segmentation is done.
- Otherwise,
  - Obtain the break-points of the PAW upper-contour.
  - Obtain a set of dominant points on the contour by using collinear-points suppression.
  - Select a possible set of cut-points from this set of dominant points.
  - Refine the set of cut-points using some heuristics.

Assume that there are m points on the PAW upper contour in counter-clockwise order. From these m points, a set of dominant points is selected using collinear-points suppression technique [39]. Collinear-points suppression may sometimes remove possible segmentation points when the contour is relatively straight. To alleviate this problem, extra dominant points are inserted by checking the horizontal length of the contour between two consecutive dominant points. If the length is greater than t times the stroke-thickness, then a new dominant point is inserted at that contour segment. Here, the value of t $(=4)$ is selected through experimentation with the PAWs extracted from the database in [26]. These PAWs are extracted using our PAW extraction algorithm described in the previous section. Fig. 9(b) and (c) illustrates this process.

Let the n dominant points on the upper contour, selected in the previous step, be denoted by $D_i=(x_i, y_i)$. It is assumed that the segmentation points of the word image are a subset of these n dominant points. As a first step of obtaining those segmentation points, we select a subset of points from the dominant points called *cut-points*. Cut-points are selected as follows.

As shown in Fig. 9(d), each dominant point obtained in the previous step can be marked as either *convex* or *concave* point. It is assumed that a segmentation point cannot be a convex point. Let D_i, D_{i+1} and D_{i+2} be three consecutive dominant points. Without loss of generality, we assume that the point D_{i+1} is shifted to the origin. Now, we divide the space into eight regions as illustrated in Fig. 10(a). The points D_i and D_{i+2} are each given a label according to the region in which they fell. Let R_1 and R_3 be the labels of D_i and D_{i+2} respectively. Here R_1 $(R_3) \in \{1, 2, 3, 4, 5, 6, 7, 8\}$. We say that $\langle R_1, R_3 \rangle$ is the configuration of D_{i+1}. Now, $\langle R_1, R_3 \rangle$ is *admissible* if it takes one of these values: $\langle i, j \rangle$, $i \in \{2, 3, 4, 5\}$ and $j \in \{5, 6, 7\}$. We select D_{i+1} as a *cut-point* if D_{i+1} is a *concave* point and the configuration of D_{i+1} is *admissible*.

Fig. 10(b) illustrates some of these admissible configurations. Once the set of cut-points are obtained, a set of rules are applied on these cut-points to select the final set of segmentation points. These rules are summarized below.

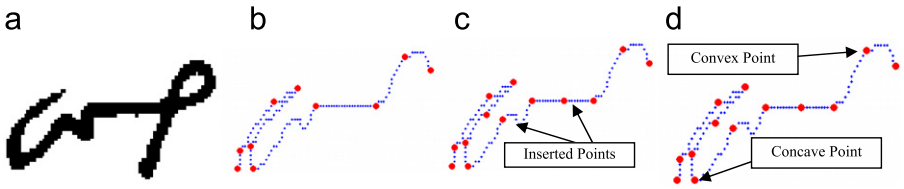

Fig. 9. Illustration of dominant points on upper contour: (a) word image, (b) dominant points on the upper contour, (c) extra dominant points inserted and (d) concave and convex dominant points.

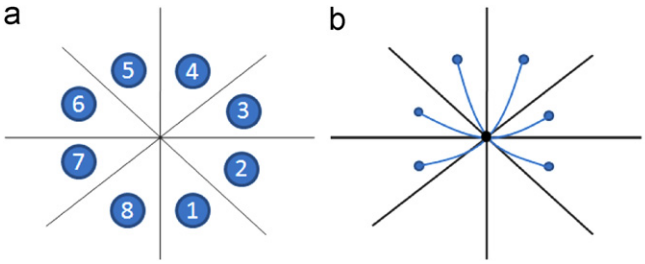

Fig. 10. Illustration of admissible configurations: (a) division of space into eight regions and (b) location of points for admissible configurations.

Table 1
Illustration of some dots and diacritics used in Arabic handwriting.

Dots/diacritics	Examples	Location	Dot count
Single dot		Above/ below	1
Double dots		Above/ below	2
Triple dots		Above	3
Shaddah		Above	–
Hamza		Above/ below	–
Fathah/double Fathah	/	Above	–
Kasrah/double Kasrah	/	Below	–
Dhammah/double Dhammah	/	Above	–
Madda		Above	–

- Replace consecutive cut-points by one cut-point (preferably the middle one). This indicates a *kashida* (elongation of the characters' interconnection). A set of cut-points are *consecutive* if there are no dominant points in between them.
- Remove any cut-point which cut a hole in the original word image as it will cut a character.
- Remove any cut-point if the stroke-thickness at that point in the word image is greater than 1.5 times the estimated stroke-thickness. This indicates within character segmentation point.

The refined sets of cut-points define a possible segmentation of the PAW contour. Segmented characters of a PAW image are called *candidate characters*, since the segmentation step may over-segment the word (Fig. 8(c)). These candidate characters are used in the word recognition system to locate the best character boundary, as described later.

4.5. Extraction of dots

Assume that the character segmentation algorithm returns $n-1$ segmentation points for a PAW image I. Note that some of these segments may be part of a character in the word image due to the over-segmentation of I. An example of this is shown in Fig. 8(c), where the character Seen (سـ) is cut into three segments. For each of these segments, several features like the (x, y) co-ordinates of the boundary of C, centroid of C, position (*isolated*, *beginning*, *middle* or *end*) and number of holes (if any) are extracted.

Dots can provide valuable information for the recognition of Arabic characters, as some characters share the same common body and differ by the number and position of dots (like ـبـ and ـخـ). Table 1 shows different dots and diacritics used in Arabic handwriting. Some dots can be written in different ways, making the extraction of dot information itself a challenging task. Fig. 11 illustrates some cases where extracting dot information may become very challenging. In many cases, higher level information is needed (like context, lexicon etc.) to correctly extract dot information. To correctly recognize the characters in a word (or PAW), we need to identify both the number and location of dots.

Therefore, we extract dot information from a word or PAW using the procedure described below.

4.5.1. Identify main body

In a PAW or word, all the connected components are identified. The component having the maximum area is identified as the main body of the PAW or word. Dot information is extracted with respect to this main body. All other components in the PAW image are called secondary components.

4.5.2. Identify dots

The secondary components of a PAW can be dots, Shaddah, Hamza, part of a broken character etc. Therefore, we need to identify those secondary components that are possibly written as dots. Morphological properties of the secondary components can be utilized to identify dots. This is done as follows.

The location of each secondary component can be estimated from baseline of the PAW image I. Since this PAW image is part of a text line, the baseline of this PAW is approximately the same as the baseline of the line itself. The baseline estimation of a text line is discussed earlier. A secondary component SC_i is *above* the PAW main body, if the centroid of SC_i is located above the baseline. SC_i is *below* the main body of I if the centroid of it is below the baseline.

Let (W_i,H_i) be the width and height of the bounding box of SC_i. Let H be the height (in pixels) and T be the stroke-thickness of the image I. The classification of the secondary components is done as follows.

- *Part of a character*: SC_i is part of a character or an overlapped Alef, if H_i is more than one-third of H. This is the case with the vertical part of the Ttaa (طـ) character, which is written by some writers isolated from the main body (Fig. 12(a)).
- *Shaddah or triple dot* ( ): The upper contour C of SC_i is extracted. Note that C is an open curve. A polygonal approximation is obtained for this upper contour C by applying collinear-suppression technique with a distance threshold of $2T$. If the number of dominant points in the polygonal

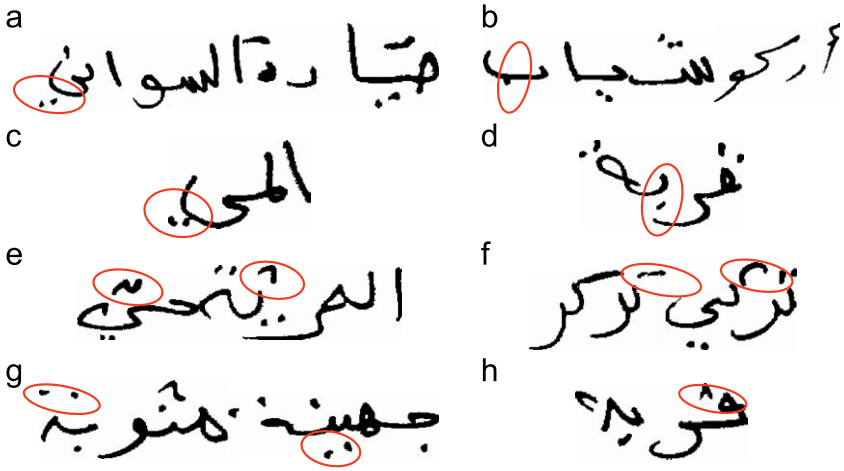

Fig. 11. Illustration of cases where extracting dot information is not straightforward (images taken from set-*e* of IfN/ENIT database).

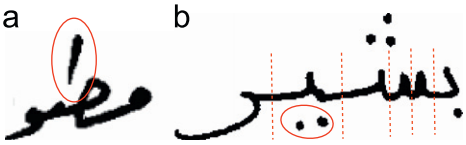

Fig. 12. Identifying dots from different secondary components of a PAW.

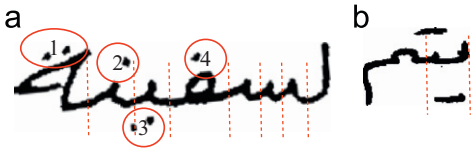

Fig. 13. Assignment of dots to PAW segments.

approximation of C is 3, then SC_i can either be a triple dot ( ) or a Shaddah ( ). Let (y_1,y_2,y_3) be the y-coordinates of the three dominant points on C. Then SC_i is identified as a Shaddah if y_2 is less than y_1 or y_3. Otherwise it is marked as a triple dot.

- *Double dot* ( ): In this case, the width W_i of the component SC_i is much larger than the height H_i. Therefore, SC_i is identified as a double dot, if $W_i/H_i > n,\ n > 1$.
- *Single dot* ( ): If SC_i is not categorized by any of the above mentioned criteria, then SC_i is marked as a single dot.

In the above description for extracting dots and diacritics, several heuristics are used. The values of these heuristics are optimized by experimenting with the PAWs extracted from the database in [26]. The results of the dot extraction process are verified manually and the values of the heuristics are adjusted so as to get better results.

Assume that the main body of the PAW image is segmented into m segments. Then, single dots which are above (below) the same segment of the main body are grouped together into double dots ( ) or triple dots ( ). Fig. 12(b) shows an example where two single dots are grouped together into double dots. Further grouping of dots is possible while searching for the best combination of characters of a PAW.

4.6. Assignment of dots

Dots extracted in the previous step are assigned to the different segments of the PAW main body. Let the PAW main body be divided into m segments, $S_i, i = 1,2,...,m$. Extracted dots are processed from left to right, as shown in Fig. 13. For each dot dot_i, the overlappings of the bounding box of dot_i with each segment S_i, $i = 1,2,...,m$ are computed. Dot dot_i is assigned to the segment S_i if the overlapping with S_i is maximum among all the segments S_i, $i = 1,2,...,m$.

Note that, there may be cases where two dots with conflicting locations may be assigned to the same segment by the above criteria. Fig. 13(b) illustrates one such case. These cases can be handled by assigning the dot on the right to the nearest segment other than the current segment. If this is not possible, then we assign the left dot the nearest segment other than the current segment. If none of the dots can be assigned to another segment, then the dot which has least overlapping with the current segment is discarded.

5. Recognition of PAWs

The basic idea of recognition of a PAW is to recognize its characters. Then the recognized characters are combined to find the possible hypotheses of a PAW.

5.1. Arabic character modeling

The core of the recognition system, as described in this paper, is the recognition of Arabic characters. Therefore, in this section, we describe how the models for Arabic characters are built. We also present the novel fuzzy dissimilarity measure for recognition of characters.

Each sample of the Arabic characters is binarized and smoothed using a statistical average based smoothing [30]. Then the contour of the character image is extracted [40] and represented by a polygonal approximation [39]. Let $D_i=(x_i,y_i)$, $i=1, 2,..., n_d$ be the sequence of dominant points of the polygonal approximation of the character contour C. These points define a polygon P of n_d edges, where each edge d_i of P is a vector D_iD_{i+1} with length $l_i=|d_i|$, where $d_i=d_{i+nd}$. A suitable representation of this polygon P is the *turning function* [3]. The turning function, or cumulative angle function, $\theta_A(s)$ of a polygon or polyline A gives the angle between the counterclockwise tangent and the x-axis as a function of the arc length s. In modeling the character contour C, we describe the vector d_i as a fuzzy direction. Fuzzy directions are fuzzy sets that have membership functions similar to those of fuzzy numbers that are characterized by possibility distributions. In the modeling of Arabic handwritten characters, π numbers [2,40] are used to model the directions d_i. These numbers are denoted by $\pi=(P_1/\beta_1;\theta;P_2/\beta_2)$, where $P_1=\theta-\gamma_1$, $\gamma_1>0$, and$p_2=\theta+\gamma_2$, $\gamma_2>0$.

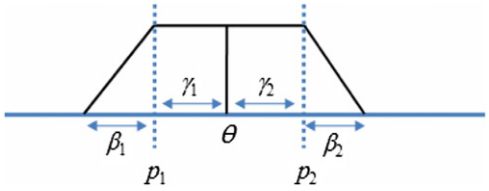

Fig. 14. Illustration of a π number.

Fig. 14 illustrates the concept of πnumbers. In a πnumber, the membership value is 1 in $[p_1, p_2]$, and decreases linearly in $(p_1, p_1-\beta_1)$ and $(p_2, p_2+\beta_2)$. This simple modeling of directions can effectively handle variations in strokes' directions.

With this definition of a π number, we now describe a novel fuzzy modeling of handwritten characters. We call $\Phi_A(S)$ as *Fuzzy Attributed Turning Function* (FATF), where the angles in $\Phi_A(S)$ are modeled as π numbers $(p_1/\beta_1;\theta;p_2\backslash\beta_2)$. The dissimilarity measure $d(A,B)$ for two FATF $\Phi_A(S)$ and $\Phi_B(S)$ is defined as follows: $d(A,B)=\left(\int_0^l |\Phi_A(s)-\Phi_B(s)|^p ds\right)^{1/p}$, where l is the length of the perimeter of the scaled polygons.

Now for L_2 metric, the integral $\int_0^l |\Phi_A(s)-\Phi_B(s)|^2 ds$ is computed by adding up the value of the integral within each *strip*. Here a strip is defined by a consecutive pair of discontinuities in $\Phi_A(S)$ and $\Phi_B(S)$. The integral within a strip is computed as $w_s \times d_s^2$, where w_s is the width of the strip and d_s is the 'fuzzy' difference between $\Phi_A(S)$ and $\Phi_B(S)$ within that strip. To compute d_s, assume that, within a strip $\Phi_A(S)=\varphi_1$ and $\Phi_B(S)=\varphi_2$. Here, φ_1 is represented by a π number $(p_1/\beta_1;\theta;p_2\backslash\beta_2)$. Since d_s is used as the dissimilarity measure for writing directions, it is computed as follows.

$$d_s = \begin{cases} 0, & \text{if } \varphi_2 \in [\theta-\gamma_1, \theta+\gamma_2] \quad \{\text{Case 1}\} \\ 1, & \text{if } \varphi_2 > (\theta+\gamma_2+\beta_2) \text{ or } \varphi_2 < (\theta-\gamma_1-\beta_1) \quad \{\text{Case 2}\} \\ \frac{\varphi_2-p_2}{\beta_2}, & \text{if } p_2 < \varphi_2 < p_2+\beta_2 \quad \{\text{Case 3}\} \\ \frac{p_1-\varphi_2}{\beta_1}, & \text{if } p_1-\beta_1 < \varphi_2 < p_1 \quad \{\text{Case 4}\} \end{cases}$$

For a detailed discussion of Fuzzy Attributed Turning Functions (FATF) and its use in handwriting recognition, readers are referred to [40].

5.2. Prototype selection

In our experimentations on Arabic characters, we have used nearest neighbor (NN) as the classifier. NN is a simple and powerful classifier and is used as a benchmarking classifier in many cases [20]. However, the main drawback of the nearest neighbor based approach is the lack of machine learning or generalization. Here all the samples in the training set are compared with an unknown character sample to decide its class. In the case of large training set, the classification may become very slow. To alleviate this problem of NN, two main approaches have been proposed by researchers:

In *Template reduction*, the number of samples (called *templates*) in the training set is reduced by discarding some samples. The idea is that many samples in the training set are very similar and may not affect the recognition performance if removed from the training set. There are a number of algorithms for template reductions proposed by researchers [19,50]. However, these algorithms have their own limitations. For example, effects of the reduction on the classification performance may not be predicted. The amount of reduction depends on the parameters of the algorithms, and all these template reduction algorithms are computationally intensive.

In *Prototype selection*, one or more prototype samples (also called *allographs*) are selected as the representative of the samples of a particular class. There are a number of algorithms for prototype selections [8,38]. Automatic selection of allographs needs clustering of the training samples. Then prototypes are selected for each cluster. This approach may be difficult if the number of clusters is not known.

The problem of prototype selection becomes easier when the number of classes is known. In our case, the number of classes is the same as the number of different characters' shapes in Arabic script. Thus, the computational overhead of the NN approach can be reduced by selecting appropriate prototypes for each character class. Therefore, in this work, we follow this approach.

In the domain of structural pattern recognition, the concept of *set-median* can be used to select prototypes from a particular class [24]. A set-median of a character class is a sample in that class which has minimum sum of distances (SOD) to all other samples in that class. Each sample in a character class is represented by the fuzzy attributed turning functions (FATF). A dissimilarity measure has been defined for FATFs previously. This dissimilarity measure is used to calculate the SOD of each sample in a character class. The sample having the minimum SOD in a character class is taken as the median of that class. Thus, in this work, the set-median of a class is used as the representative of that particular character class.

5.3. Recognition of characters in a PAW

In this section, we describe our method for the recognition of the best sequence of characters in a PAW.

Assume that a PAW main body is segmented into n segments using the algorithm described earlier. Fig. 15 illustrates a PAW divided into five segments (S_1–S_5). The Sheen in the middle (ـشـ) is divided into three segments (S_2, S_3, S_4). The first two segments (S_2 and S_3) do not represent any valid Arabic characters, although the third segment (S_4) may seem like Thaa (ـثـ). However, by combining these three segments, we can recognize the Sheen correctly.

Assume that the n segments of the PAW image I are represented by $(S_1,S_2,\ldots,S_n)$. For each segment S_i, there is a candidate character C_i associated with it. Note that C_i contains additional information, like the number and location of dots in S_i, presence of holes, etc.

Let $S_{i..j}$ be the *compound segment* formed by concatenating the segments $S_i,S_{i+1},\ldots,S_j$. Fig. 15 illustrates the compound segments $S_{2..3}$ and $S_{2..4}$. Let $C_{i..j}$ be the compound candidate character associated with $S_{i..j}$. Note that not all the adjacent segments in a PAW can be concatenated together. Two adjacent segments S_i and S_{i+1} can be concatenated together if the following conditions are satisfied.

- C_i and C_{i+1} have dots (if any) with non-conflicting locations. For example, if C_i contains a dot above and C_{i+1} contains a dot below, then S_i and S_{i+1} cannot be joined together.

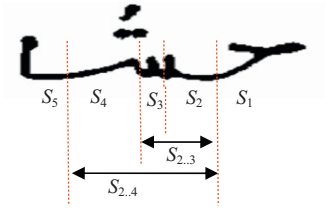

Fig. 15. Illustration of a PAW divided into segments.

- Number of dots in C_i and C_{i+1} are non-conflicting. For example, if C_i contains a double dot and C_{i+1} contains a triple dot, then S_i and S_{i+1} cannot be joined together.

Note that, the position of $C_{i..i+1}$ depends on the positions of C_i and C_{i+1}. Table 2 shows the rules to determine the position of $C_{i..i+1}$. As an illustration of the rules given in Table 2, consider the candidate characters C_4 and C_5 (corresponding to the segments S_4 and S_5) in Fig. 15 which have positions 'middle' and 'end' respectively. Therefore, the position of the compound candidate character $C_{4..5}$, which is formed by con catenating C_4 and C_5, is 'end'.

Now, we determine the best sequence of characters in a PAW as described below. Let *Score* be a $n \times n \times d$ matrix of scores. Here, *Score*[i, j, k] is kth best matching score for the candidate character $C_{i..j}$ with the models in the training set. Here,$C_{i..i}$ is same as C_i. The corresponding matching character is stored in the *MatchedChar*[i, j, k] matrix. Fig. 16 shows the structure of *Score* and *MatchedChar* matrices for a PAW with four segments. For each value in *Score* and *MatchedChar*, Fig. 16 illustrates the candidate character $C_{i..j}$ and the segment $S_{i..j}$ that is mapped to that particular entry, $i=1, 2, 3, 4$ and $j=1, 2, 3, 4$. In Fig. 16, only one hypothesis is generated for each $C_{i..j}$.

Note that, both *Score* and *MatchedChar* are upper triangular matrices. To reduce computation, it is assumed that an Arabic character can be segmented into a maximum of three parts. Therefore, we concatenate two or three consecutive segments to get compound candidate characters. Moreover, $C_{i..j}$ is matched with a character model if the model and $C_{i..j}$ have the same position and the same number and position of dots.

From the *Score* and *MatchedChar* matrices, top n hypotheses for a PAW are generated. To generate the best hypothesis for a PAW, dynamic programming can be used. Using the following recurrence relation, we can generate the best matching hypothesis for a PAW:

$$AggScore(i,j) = \min(AggScore(i-1,i-1)+Score(i,j),AggScore\ (i-1,j)).$$

Here, *AggScore*(i,j) stores the best matching score for a compound candidate character $C_{i..j}$. As the basis, we take *AggScore* (i,i) = *Score*(i,i). Each hypothesis for a PAW states a probable sequence of characters comprising the PAW. These hypotheses are sorted by their matching score. Note that, not all combinations of entries from *MatchedChar* are possible.

Table 2
Rules for determining the position of a compound candidate character.

C_i	C_{i+1}	$C_{i..i+1}$
Beginning	Middle	Beginning
Beginning	End	Isolated
Middle	Middle	Middle
Middle	End	End

	1	2	3	4
1	$C_{1..1}$, $S_{1..1}$	$C_{1..2}$, $S_{1..2}$	$C_{1..3}$, $S_{1..3}$	$C_{1..4}$, $S_{1..4}$
2		$C_{2..2}$, $S_{2..2}$	$C_{2..3}$, $S_{2..3}$	$C_{2..4}$, $S_{2..4}$
3			$C_{3..3}$, $S_{3..3}$	$C_{3..4}$, $S_{3..4}$
4				$C_{4..4}$, $S_{4..4}$

Fig. 16. Illustration of the structure of *Score* and *MatchedChar* matrices for a PAW segmented into four parts.

6. Experimentations and discussion

In this section, we present our results for the recognition of Arabic handwritten words. We present comparative results with other techniques and analyze the performance of the proposed technique.

6.1. Recognition of digits

Before we discuss our experimental results on Arabic words recognition, we demonstrate the performance of FATF by recognizing Arabic numerals or digits. For this purpose, we use an Arabic handwritten database of isolated digits called ADBase [18]. The ADBase is composed of 70,000 digits written by 700 writers. Each writer wrote each digit (from '0' to '9') ten times. The database is partitioned into two sets: a training set (60,000 digits samples) and a test set (10,000 digits samples). An average accuracy of 97.18% is obtained for this database. The recognition accuracies for each numeral are shown in Table 3.

In the experimentation of digits, a simple nearest neighbor classifier is used along with FATF without any prototype reduction. After analyzing the erroneous cases manually, we have found that around 41% of the errors occurred due to bad handwriting, which is difficult to be recognized even by humans. Further 22% of the error occurred due to broken digits, which was not handled by the model.

6.2. Generation of models

We now describe our experimentations with Arabic words recognition. The character models in the training set are built from characters and are reduced by selecting setmedians. Therefore, in this work we use pre-segmented Arabic characters for generating the character models. For this purpose, we utilize two databases of Arabic handwritten characters [2,26]. The database in [2] contains around 2000 isolated Arabic character shapes without dots written by four writers. However, some character shapes are missing in [2]. Therefore, we utilize the database in [26] to get more character samples. This database contains all the Arabic character shapes where each character shape has 48 samples written by 48 writers. The characters in [26] were obtained by segmenting manually the Arabic words written by the 48 writers. Character samples in [26] which are broken or difficult to recognize (by human) are not included in our training set. For each of these characters in the training set, we build models based on FATF. These models are later reduced by selecting the set-medians from the character classes.

In handwriting recognition, handwritten text is size normalized to make the samples of the same character written by different writers comparable. Normalizing all the characters to the same size can destroy valuable information, like the relative heights of the characters. This issue is particularly important for Arabic script. For example, the handwritten samples of Dal (د) and Lam (ل) may look similar if their height is same (see Fig. 17(a)). However, Lam is written with much higher height compared to Dal. We can easily separate Lam from Dal by using their relative heights. Blind normalization of the Arabic characters may destroy this valuable information and may become an additional source of errors in the system. Therefore, in this work we adopt the following normalization process.

The text lines extracted from a page are normalized such that the height of a line is arbitrarily set to 96 pixels. The width of a line is normalized keeping the aspect ratio of the original image of the line. Since we normalize a line itself, the relative height of the characters in the normalized line remain same as in the original image.

The issue of normalization is more challenging for the segmented characters in the training set. Since we cannot normalize

Table 3
Recognition results on Arabic handwritten digits.

Digit	٠	١	٢	٣	٤	٥	٦	٧	٨	٩
Accuracy (%)	92.50	98.40	98.80	96.40	97.80	95.50	98.10	99.10	99.00	96.20

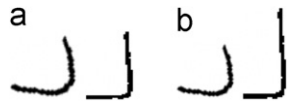

Fig. 17. Illustration of cases where relative height of Arabic characters is used to avoid confusions.

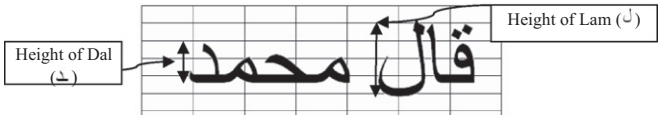

Fig. 18. Illustration of the process of estimating the relative heights of Arabic character shapes.

all the characters to the same height (say 96 pixels), a more robust normalization process is required. To accomplish this, we first estimate the ideal relative heights of Arabic characters. For this purpose, we take some printed text containing all the shapes of Arabic characters. From this text, the relative heights of all the Arabic characters are estimated manually. Fig. 18 illustrates this process. Therefore, an isolated character in the training set is normalized with respect to its relative height. For example, the relative height of Kaf (ك) is estimated as 0.5. Therefore, all samples of Kaf are normalized to the height 96*0.5=48 pixels keeping the aspect ratio of the original image.

The normalization process described above is an approximate method. However, this approximation is adequate for our purpose in this work. This is because the dissimilarity measure between two character samples in this work is robust enough to handle some variations in the size of the character samples.

6.3. Experimentation with IfN/ENIT database

Since no bench marking database is available for Arabic text recognition, we present our experimental results with IfN/ENIT database [43]. This database consists of city names and not real Arabic handwritten text pages and lines. Currently, this database is the most widely used Arabic handwritten text database. Many researchers have published results on IfN/ENIT. Thus, experimental results on IfN/ENIT database can enable us to compare our system with others.

This is a database of handwritten Tunisian city names. The current version (v2.0p1e) available to the researchers contains a total of 32,492 handwritten city names by more than 1000 writers. The vocabulary is limited to 946 city names. Each city name is composed of one or more words. IfN/ENIT database is divided into five sets: *a*, *b*, *c*, *d* and *e*.

Generally, researchers train their system on sets *a*–*d* (four sets) and test on set-*e*. Therefore, we have tested our text recognition approach on set-*e* of IfN/ENIT database. This set contains 6033 samples of city names. We trained our system on segmented Arabic characters of other databases [2,26] as producing segmented isolated characters from IfN/ENIT has two problems. Segmentation takes time and error prone. In addition, since the database consists of city names, we may not be able to produce enough samples for some of the Arabic character shapes.

Since IfN/ENIT is a closed vocabulary database, we can use a lexicon of city names and map the recognized city names to the entries in the lexicon. This practice is used by other researchers using IfN/ENIT database. Therefore, in our experimentation, a lexicon of 946 city names is used to refine the recognition results.

6.3.1. Lexicon consultation and reduction

In our experimentations with IfN/ENIT database, the lexicon of Tunisian city names is used in two ways. First, a recognized city name is mapped to the entries of the lexicon to refine the recognition results. Second, to avoid the consultation of the entire lexicon for each hypothesis of a recognized PAW, a *lexicon reducer* is used to reduce the lexicon to a smaller one. The design of a lexicon reducer is explained below.

As part of our recognition process, we extract the dot information for a word image. This information for PAWs can be utilized to *reduce* the lexicon of city names of IfN/ENIT database. Given a lexicon $L=\{l_1,l_2,...,l_n\}$ of n words, a *lexicon reduction system* determines a subset $L' \subset L$ of words in the recognition of an unknown word. The other entities from the set $L-L'$ are no longer taken into consideration for the current word under test.

Using dot-descriptors to reduce a lexicon was introduced by Mozaffari et al. [36]. They applied their method to a lexicon of 200 city names. In this work, we have integrated this lexicon reducer to the recognition system and applied it to a much larger lexicon.

Dot-descriptor of an Arabic word lists the number and position of the dots in the word image. For example, for the word قليغير, the dot-descriptor would look like 1U2D1U2D. Here, each dot is represented by a number (1/2/3) and the location (U/D) and processed from right to left.

We generate and store the dot-descriptors of all the lexicon entries (here, 946 city names). The dot-descriptor of an unknown word image is matched with that of a lexicon entry using *Damerau–Levenshtein* (DL) distance [11,28]. Damerau–Levenshtein algorithm measures the distance between two strings by computing the minimal number of 'edit operations' needed to transform one string into the other. Edit operations are usually defined as insertions, deletions, and substitutions. In this work, all three edit operations have the same cost of one. Table 4 shows the different errors that may happen while extracting dot-descriptors. It also lists the edit operations needed to correct those errors and the associated costs.

DL distance is used in two ways in our system. First, DL distance is used to compute the edit distance between the character strings of a recognized word and a lexicon entry. Second, it is used to re-rank the lexicon entries by comparing the dot-descriptors.

With this lexicon reducer at hand, the aggregate score of a word hypothesis is calculated as follows. Assume that we have m hypotheses for a word image I. For each hypothesish_i, we have a confidence level l_i. Now, we have three matching scores for each h_i with a lexicon entry W:

- DL_{char}: This is calculated by finding the DL distance between the character sequence of h_i with that of W.
- DL_{dot}: Distance between the dot-descriptors of h_i and W.
- p: Difference in the number of PAWs in I and W. We count the number of PAWs in a city name image while extracting the PAWs (in the word extraction phase).

Then the aggregate score $Agg_{h_i/W}$ that measures the distance from I to W (for the hypothesis h_i) is estimated as: $Agg_{h_i/W}=(DL_{char}+DL_{dot}+p)\times l_i$. The final score Agg_W of W for all the m hypotheses is taken as the minimum of all $Agg_{h_i/W}$. That is,

Table 4
Different cases of errors in extracting dot-descriptors and costs for correction.

Description of errors	Correcting operation	Cost for correction	Example
Error in finding the location of a dot	S	1	1U→1D
Error in estimating the number of dots	S	1	1U→2U
Touching dots in the word image	2×I	2	3U→2U1U
Identifying a non-dot component as a dot	2×D	2	3U1U→1U
A dot is identified as a non-dot component	2×I	2	1U→3U1U

Here, S≡Substitute, I≡Insert and D≡Delete.

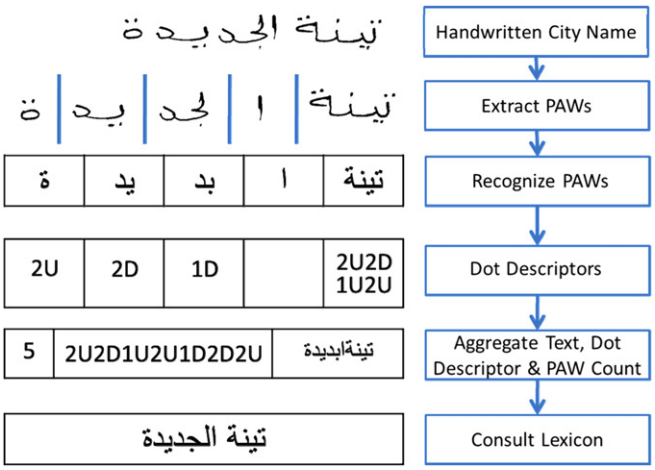

Fig. 19. Illustration of steps to recognize handwritten city names from IfN/ENIT database.

$Agg_W = \min\left(Agg_{h_i/W}\right)$, $i = 1,2,\ldots,m$. Once the final scores of the lexicon entries are calculated, top k lexicon entries are selected as the final hypothesis for the word image I.

Fig. 19 summarizes the steps explained in this section for the recognition of Arabic handwritten city names from IfN/ENIT database.

6.3.2. Results and discussions

We now present our results on set-*e* of IfN/ENIT database. For this set, we have achieved word recognition accuracy of 79.58%. Table 5 gives the comparative results on set-*e* of IfN/ENIT database. Fig. 20 shows the comparative results in a graphical form. A detailed analysis of the results is given below.

As can be noted from Table 5, the proposed system is the first attempt to experiment with IfN/ENIT database with structural techniques. Most of the previous results on IfN/ENIT database are based on statistical methods [13], with HMM being the most popular. The technique presented in this work is the first of its kind to be applied on a large dataset.

Table 5 also shows the training sets used by the other researchers. Most of the previous work on IfN/ENIT database had trained their systems on the sets *a–d* and tested on the set-*e*. This is because of the need for large training data for learning by the statistical classifiers. However, the proposed approach has not used the data from IfN/ENIT for training at all. We have generated our character models from completely different databases of isolated characters. Therefore, the testing phase in our system is completely independent from the training phase. This illustrates the robustness of the proposed system to handle unknown data. Note that the size of the training set in our system is much less compared to IfN/ENIT. Set *a–d* of the current version of IfN/ENIT contains 26,459 city names written by more than 400 writers, where each city name consists of one or more words. A total of 212,211 characters and ligatures are written in these 26,459 samples of city names. However, the two databases of isolated characters [2,26] used for generating our character models contain around 7900 isolated characters written by 52 writers. We consider that our system does not need training on new data as an advantage for several reasons. There is no real Arabic handwritten database that is freely available with enough data. We need not segment IfN/ENIT database to produce Arabic isolated characters, which is time consuming and error prone. We consider this as a step towards addressing one of the future challenges as stated in [44]: "*Handwriting Recognition systems could, for example, in the future also be trained on machine printed text and later only be adapted to handwritten. The problem of limited data sets for handwritten script would then be largely alleviated*". Using limited handwritten isolated characters to recognize cursive text is a step in that direction and address the lack of limited data for Arabic handwritten text research.

However, due to the usage of different training sets in our system, the comparisons of performance presented in Table 5 can be regarded as the comparisons of *overall systems*, rather than the comparisons of *individual algorithms*.

As we have discussed before, the character models generated in our system have been reduced by selecting set-medians. This means, we retain only one sample per character class. There are two implications of this approach. First, using set-medians reduce the number of character models to match with an unknown sample. This in turn increases the recognition speed. Second, keeping only one sample per character class reduces the recognition rates. Thus the recognition accuracy obtained by our approach has been improved due to the fuzziness introduced into the model itself. The fuzzy attributed turning function (FATF), used in this work, has proved itself to be extremely powerful in handling variability in writings. Hence, using more models for characters is expected to improve the recognition rates further.

IfN/ENIT is a closed vocabulary database. Thus, most researchers have used a lexicon of IfN/ENIT city names to improve recognition accuracy. In our method, we have utilized the dot-descriptors and PAW counts to reduce the lexicon. This reduction approach is seamlessly integrated into the system, as the information needed for lexicon reduction is already available in the system. This benefit is due to the use of structural approach. Integrating lexicon reduction techniques with an HMM based system may require a completely separate feature extraction and classification stage [36].

As for the recognition accuracy itself, our approach gives performance comparable with the state of the art results on the same data. As can be seen in Table 5, the proposed system has accuracies comparable to the multi-classifiers systems. However, the accuracies of individual systems of these multi-classifiers systems are much lower. For example, the multi-stream HMM based system proposed by Kessentini et al. [25] had an accuracy of 79.6% for set-*e*. However, the individual systems in [25] had accuracies ranging from 63.5% to 70.5%. These accuracies are much lower compared to our approach, which is basically a single classifier system. Therefore, the proposed method shows significantly superior performance when compared to the single classifier systems. The performance of our system is comparable with that of multi-classifiers systems. It has been shown by researchers that multi-classifiers systems generally show better performance compared to the component single classifiers [14]. Thus the performance of our algorithm suggests that integrating other

Table 5
Comparative results for set-*e* of IfN/ENIT database.

Authors	Classifier	Training data	Lexicon usage	Word recognition rate (%)	
				Single-classifier	Multi-classifiers
Märgner et al. [33]	HMM	Set *a–d*	Yes	74.69	–
Kundu et al. [27]	Variable Duration HMM	Set *a–c*	Yes	60	–
Elbaati et al. [16]	HMM	Set *a–d*	Not mentioned	54.13	–
Hamdani et al. [23]	Multiple HMM	Set *a–d*	Not mentioned	49.48–63.90	81.93
Kessentini et al. [25]	Multi-stream HMM	Set *a–d*	Yes	63.5–70.5	79.6
Proposed	FATF with set-medians	Isolated characters	Yes	79.58	–

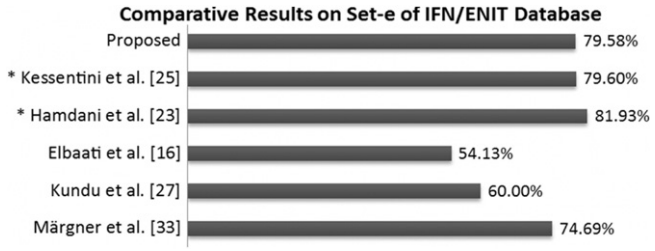

Fig. 20. Comparative results on set-*e* of IfN/ENIT database.

statistical classifiers with the proposed method may significantly improve the performance for IfN/ENIT database.

It can be noted here that, IfN/ENIT database has been used in several competitions recently [34]. In the most recent competitions, set-*d* and set-*e* are used as part of the training set. New sets (set-*f* and set-*s*), which are unknown to the researchers, are used to evaluate performance. In [34], the best result reported for set-*f* is 92.20% and for set-*s* is 84.55%. Due to the unavailability of the new testing sets of IfN/ENIT to us, we have to resort to the '*older*' testing sets to evaluate and compare our system.

7. Conclusions

In this paper, we present our research results on off-line Arabic handwriting text recognition using structural techniques. Developing structural methods for Arabic handwriting recognition can be incorporated into recognition systems based on statistical methods to improve recognition accuracy. It may also be used in multi-classifier systems. This paper introduces several novel ideas and techniques for structural recognition of Arabic handwriting. An Arabic text line is segmented to words/sub-words and dots are extracted leaving the parts of Arabic words (PAW). The dots' number and location are used in the recognition phase to discriminate between different characters with the same primary part. The PAW is then slant corrected using a novel slant estimation and correction technique based on our polygonal approximation that is robust and have higher resolution than chain code/contour based techniques. Our novel segmentation algorithm is integrated into the recognition phase of handwritten text. The segmentation algorithm is based on the characteristics of Arabic writing. The technique is robust and does not suffer from the limitations of projection-and contour-based techniques.

Modeling of Arabic characters is done by 'fuzzy' polygons. These models are generated from Arabic characters not related to the tested text. Recognition of Arabic PAWs is done using a novel fuzzy polygon matching algorithm. Dynamic programming is used to select the best hypothesis of a sequence of recognized characters of each PAW, as all possible segments' combinations to characters are investigated. In addition, we utilize several other key ideas, namely prototype selection using set-medians, lexicon reduction using dot-descriptors etc. to design a robust handwriting recognition system.

Results are reported on the benchmarking IfN/ENIT database of Tunisian city names. These results are compared with published work using the same database. The results indicate the robustness and the effectiveness of our system. Our recognition rates are better than all single classifier systems, although we did not use IfN/ENIT database for training in modeling Arabic characters and all other techniques used IfN/ENIT database sets *a–d* for training their systems.

The authors are extending their techniques by integrating the structural relations between the different segments of Arabic characters and by integrating their structural features in a multi-classifier system.

Acknowledgments

The authors would like to thank King Fahd University of Petroleum & Minerals (KFUPM) for supporting this research and providing the computing facilities. This work was supported by KACST NSTIP project 08-INF99-4 "Automatic Recognition of Handwritten Arabic Text (ARHAT)". The authors also thank the anonymous reviewers for their comments that improved this work.

References

[1] S. Abdelazeem, E. El-Sherif, Arabic handwritten digit recognition, International Journal on Document Analysis and Recognition 11 (3) (2008) 127–141.
[2] I.S.I. Abuhaiba, S.A. Mahmoud, R.J. Green, Recognition of handwritten cursive Arabic characters, IEEE Transaction on Pattern Analysis and Machine Intelligence 16 (6) (1994) 664–672.
[3] E. Arkin, P. Chew, D. Huttenlocher, K. Kedem, J. Mitchel, An efficiently computable metric for comparing polygonal shapes, IEEE Transactions on Pattern Analysis and Machine Intelligence 13 (3) (1991) 209–215.
[4] S. Awaidah, S.A. Mahmoud, A multiple feature/resolution scheme to Arabic (Indian) numerals recognition using hidden Markov models, Signal Processing 89 (6) (2009) 1176–1184.
[5] N. Azizi, N. Farah, M. Sellami, A. Ennaji, Using diversity in classifier set selection for Arabic handwritten recognition, in: Multiple Classifier Systems, Lecture Notes in Computer Science, vol. 5997, 2010, pp. 235–244.
[6] A. Benouareth, A. Ennaji, M. Sellami, Semi-continuous HMMs with explicit state duration for unconstrained Arabic word modeling and recognition, Pattern Recognition Letters 29 (12) (2008) 1742–1752.
[7] R. Bertolami, S. Uchida, M. Zimmermann, H. Bunke, Non-uniform slant correction for handwritten text line recognition, in: 9th International Conference on Document Analysis and Recognition (ICDAR), 2007, pp.18–22.
[8] K. Chellapilla, P. Simar, A. Abdulkader, Allograph based writer adaptation for handwritten character recognition, in: 10th International Workshop on Frontiers in Handwriting Recognition (IWFHR), 2006, pp. 423–428.
[9] I. Ben Cheikh, A. Belaïd, A. Kacem, A novel approach for the recognition of a wide Arabic handwritten word lexicon, in: 19th International Conference on Pattern Recognition (ICPR), 2008.
[10] J. Chen, H. Cao, R. Prasad, A. Bhardwaj, P. Natarajan,Gabor features for offline Arabic handwriting recognition, in: 9th IAPR International Workshop on Document Analysis Systems (DAS), 2010, pp. 53–58.
[11] F.J. Damerau, A technique for computer detection and correction of spelling errors, Communications of the ACM 7 (3) (1964) 171–176.
[12] P. Dreuw, D. Rybach, C. Gollan, H. Ney, Writer adaptive training and writing variant model refinement for offline Arabic handwriting recognition, in: 10th

International Conference on Document Analysis and Recognition (ICDAR), 2009, pp. 21–25.
[13] H. El Abed, V. Märgner, Arabic text recognition systems—state of the art and future trends, in: 5th International Conference on Innovations in Information Technology (Innovations'08), Al Ain, UAE, 2008.
[14] H. El Abed, V. Märgner, Reject rules and combination methods to improve Arabic handwritten word recognizers, in: 11th International Conference on Frontiers in Handwriting Recognition (ICFHR), 2008, pp. 180–185.
[15] H. El Abed, V. Märgner, Improvement of Arabic handwriting recognition systems: combination and/or reject? Proceedings of SPIE 7247 (2009) 1–10.
[16] A. Elbaati, H. Boubaker, M. Kherallah, A.M. Alimi, A. Ennaji, H. El Abed, Arabic handwriting recognition using restored stroke chronology, in: 10th International Conference on Document Analysis and Recognition (ICDAR), 2009, pp. 411–415.
[17] R. El-Hajj, C. Mokbel, L. Likforman-Sulem, Recognition of Arabic handwritten words using contextual character models, Document Recognition and Retrieval XV, Proceedings of the SPIE 6815 (2008) 681503.
[18] E. El-Sherif and S. Abdelazeem, A two-stage system for Arabic handwritten digit recognition tested on a new large database, in: International Conference on Artificial Intelligence and Pattern Recognition (AIPR-07), 2007, pp. 237–242.
[19] H.A. Fayed, A.F. Atiya, Novel template reduction approach for the nearest neighbor method, IEEE Transactions on Neural Networks 20 (5) (2009) 890–896.
[20] M. Ferrer, E. Valveny, F. Serratosa, K. Riesen, H. Bunke, Generalized median graph computation by means of graph embedding in vector spaces, Pattern Recognition 43 (4) (2010) 1642–1655.
[21] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, J. Schmidhuber, A novel connectionist system for unconstrained handwriting recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (5) (2009) 855–868.
[22] A. Graves, J. Schmidhuber, Offline hand writing recognition with multidimensional recurrent neural networks, in: Daphne, Koller, Dale, Schuurmans, Yoshua, Bengio, Léon, Bottou, (Eds.), Advances in Neural Information Processing Systems, vol. 21, NIPS'22, Vancouver, 2008, pp. 545–552.
[23] M. Hamdani, H. El Abed, M. Kherallah, A.M. Alimi, Combining multiple HMMs using on-line and off-line features for off-line Arabic handwriting recognition, in: 10th International Conference on Document Analysis and Recognition (ICDAR), 2009, pp. 201–205.
[24] X. Jiang, A. Munger, H. Bunke, On median graphs: properties, algorithms, and applications, IEEE Transaction on Pattern Analysis and Machine Intelligence 23 (10) (2001) 1144–1151.
[25] Y. Kessentini, T. Paquet, A. Ben Hamadou, Off-line handwritten word recognition using multi-stream hidden Markov models, Pattern Recognition Letters 31 (1) (2010) 60–70.
[26] M. Khedher, G. Abandah, Arabic character recognition using approximate stroke sequence, in: Proceedings of the Arabic Language Resources and Evaluation-Status and Prospects Workshop, 3rd International Conference on Language Resources and Evaluation (LREC 2002), 2002.
[27] A. Kundu, T. Hines, J. Phillips, B. Huyck, L. Van Guilder, Arabic handwriting recognition using variable duration HMM, in: 9th International Conference on Document Analysis and Recognition (ICDAR), 2007, pp. 644–648.
[28] V.I. Levenshtein, Binary codes capable of correcting deletions, insertions, and reversals, Soviet Physics Doklandy 10 (8) (1966) 707–710.
[29] L. Lorigo, V. Govindaraju, Offline Arabic handwriting recognition: a survey, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (5) (2006) 712–724.
[30] S.A. Mahmoud, Arabic character recognition using Fourier descriptors and character contour encoding, Pattern Recognition 27 (1994) 815–824.
[31] S.A. Mahmoud, Recognition of writer-independent off-line handwritten Arabic (Indian) numerals using hidden Markov models, Signal Processing 88 (4) (2008) 844–857.
[32] U. Marti, H. Bunke, The IAM-database: an English sentence database for offline handwriting recognition, International Journal on Document Analysis and Recognition 5 (2002) 39–46.
[33] V. Märgner, H. El Abed, M. Pechwitz, Offline handwritten Arabic word recognition using HMM—a character based approach without explicit segmentation, in:9th Colloque International Francophone sur l'Ecrit et le Document (CIFED), 2006.
[34] V. Märgner, H. El Abed, ICDAR 2011—Arabic handwriting recognition competition, in: 11th International Conference on Document Analysis and Recognition (ICDAR), 2011, pp. 1444–1448.
[35] R.A. Mohamad, L. Likforman-Sulem, C. Mokbel, Combining slanted-frame classifiers for improved HMM-based Arabic handwriting recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (7) (2009) 1165–1177.
[36] S. Mozaffari, K. Faez, V. Märgner, H. El-Abed, Lexicon reduction using dots for off-line Farsi/Arabic handwritten word recognition, Pattern Recognition Letters 29 (6) (2008) 724–734.
[37] P. Natarajan, K. Subramanian, A. Bhardwaj, R. Prasad, Stochastic segment modeling for offline handwriting recognition, in: 10th International Conference on Document Analysis and Recognition (ICDAR), 2009, pp. 971–975.
[38] R. Niels, L. Vuurpijl, L. Schomaker, Automatic allograph matching in forensic writer identification, International Journal of Pattern Recognition and Artificial Intelligence 21 (1) (2007) 61–81.
[39] M.T. Parvez, S.A. Mahmoud, Polygonal approximation of digital planar curves through adaptive optimizations, Pattern Recognition Letters 31 (13) (2010) 1997–2005.
[40] M.T. Parvez, S.A. Mahmoud, Arabic handwritten alphanumeric character recognition using fuzzy attributed turning functions, in: FAHR 2010: Workshop in Frontiers in Arabic Handwriting Recognition, in conjunction with 20th International Conference in Pattern Recognition (ICPR), 2010, pp. 9–14.
[41] M.T. Parvez, S.A. Mahmoud, Off-line Arabic handwritten text recognition: a survey, ACM Computing Surveys, in press.
[42] T. Pavlidis, Algorithms for Graphics and Image Processing, Computer Science Press, Rockville, MD, 1982.
[43] M. Pechwitz, S.S. Maddouri, V. Märgner, N, Ellouze, H. Amiri, IfN/ENIT-database of handwritten Arabic words, in: 7th Colloque International Francophone sur l'Ecrit et le Document, CIFED 2002, October 21–23, 2002, Hammamet, Tunis, pp. 127–136.
[44] T. Plötz, G.A Fink, Markov Models for Handwriting Recognition, SpringerBriefs in Computer Science, Springer, 2011.
[45] K. Riesen, M. Neuhaus, H. Bunke, Graph embedding in vector spaces by means of prototype selection, in: 6th IAPR-TC-15 International Workshop (GbRPR), Lecture Notes in Computer Science, vol. 4538, 2007.
[46] S. Saleem, H. Cao, K. Subramanian, M. Kamali, R. Prasad, P Natarajan, Improvements in BBN's HMM-based offline Arabic handwriting recognition system, in: 10th International Conference on Document Analysis and Recognition (ICDAR), 2009, pp. 773–777.
[47] K.M. Sayre, Machine recognition of handwritten words: a project report, Pattern Recognition 5 (3) (1973) 213–228.
[48] M. Schambach, J. Rottland, T. Alary, How to convert a Latin handwriting recognition system to Arabic, in: 11th International Conference on Frontiers in Handwriting Recognition (ICFHR), 2008.
[49] S. Uchida, E. Taira, H. Sakoe, Nonuniform slant correction using dynamic programming, in: 6th International Conference on Document Analysis and Recognition (ICDAR), 2001, pp. 434–438.
[50] D.R. Wilson, T.R. Martinez, Reduction techniques for instance-based learning algorithms, Machine Learning 38 (3) (2000) 257–286.

Mohammad Tanvir Parvez obtained his B.Sc. and M.Sc. Engineering in Computer Science and Engineering (CSE) from Bangladesh University of Engineering and Technology (BUET), Dhaka in 2004 and 2006 respectively and a Ph.D. in CSE from King Fahd University of Petroleum & Minerals (KFUPM), Dhahran, Saudi Arabia in 2010. His research interests include Pattern Recognition, Image Processing and Machine Learning with special interest in handwriting recognition using structural approach. He has received several awards including Best Graduate Students' Award (KFUPM, 2010), First Prize in Graduate Seminar Day (KFUPM 2010), BUET Academic Performance Scholarship (1999–2003) etc. Dr. Parvez is currently working as an Assistant Professor in Computer Engineering Department at Qassim University, Saudi Arabia.

Sabri A. Mahmoud is a Professor of computer Science in the Information and Computer Science Department, King Fahd University of Petroleum and Minerals. Dr. Mahmoud received his B.S. in electrical engineering from Sind University, Pakistan in 1972, received his M.S. in Computer Sciences from Stevens Institute of Technology, U.S.A., in 1980 and his Ph.D. degree in Information Systems Engineering from the University of Bradford, U.K., in 1987. His research interests include Arabic Document Analysis and Recognition, Image Analysis (including Time Varying Image Analysis), Arabic Natural Language Processing, and Applications of Pattern Recognition in Software Engineering. He published over 70 papers in refereed journals and conference proceedings in his research areas of interest. Dr. Mahmoud is a senior member of IEEE.