Offline Automatic Segmentation based Recognition of Handwritten Arabic Words

Laslo Dinges, Ayoub Al-Hamadi, Moftah Elzobi, Zaher Al Aghbari[1]
and Hassan Mustafa[2]

Institute for Electronics, Signal Processing and Communications (IESK)
Otto-von-Guericke-University Magdeburg; Germany
{Laslo.Dinges, Ayoub.Al-Hamadi}@ovgu.de

[1]Department of Computer Science
University of Sharjah; UAE
zaher@sharjah.ac.ae

[2]Faculty of Engineering
Albaha University; Saudi Arabia
mustafa_hasan47@yahoo.com

Abstract

The world heritage of handwritten Arabic documents is huge however only manual indexing and retrieval techniques of the content of these documents are available. To facilitate an automatic retrieval of such handwritten Arabic document, a number of automatic recognition systems for handwritten Arabic words have been proposed. Nevertheless, these systems suffer from low recognition accuracy due to the peculiarities of the handwritten Arabic language. Thus, in this Paper we propose a segmentation based recognition system for handwritten Arabic words. We divide a handwritten word into smaller pieces of a word and then these small pieces are segmented into candidate letters. These candidate letters are converted into their correspondence chain-code representation. Thereafter we extract discrete, statistical and structural features for classification. Additionally, we introduce a novel active contour based feature to increase the recognition accuracy of strongly deformed Arabic letters. We also use a decision tree to reduce the number of potential classes. We then use a neural network to compute weights for all statistical features and use them as input for a k-NN classifier. Our experiments show that the extracted features by our technique achieve higher recognition accuracy as compared to other features.

Keywords: *Pattern Recognition, Character Recognition, Arabic Handwriting, Handwritten Word Segmentation.*

1 Introduction

Within the last decades information becomes preserved and used more and more in digital forms. Nevertheless, there are still a huge amount of handwritten modern as well as historical documents, without digital redundancies. Even though optical scanning can preserve digital copies of such documents in a digital image form, mining the content of the image for information is impossible, unless a subsequent transformation process into a digital text form (e.g. ASCII, Unicode, and, etc.) is accomplished. A carefully designed optical character recognition (OCR) system is a vital prerequisite for achieving satisfactory results.

In the segmentation phase, words will be segmented into their constituent characters' representatives. Then the recognition of the word will be a function of the recognition of the individual characters. Unfortunately, adaption of segmentation and/or classification methods that proved successfulness for Latin text, for Arabic text is not a straightforward process. This is because the Arabic script has some special characteristics: there are 28 *letters* (Characters) in the Arabic alphabet as shown in Table 1, the letters change their shapes dramatically correspondence to their positions (isolated-, beginning-, middle- and end form); only 6 of characters have 2 different shapes, the rest of them has 4 different shapes. Further aspects are:

1. Arabic is written from right to left
2. The occurrence of characters with only two shape forms inside a word, leads to the split of the connected word into two or more parts, called *Piece of Arabic word* (PAW), consisting of the main body (connected component) and related diacritics (dots) or supplements like Hamza (') (see Fig. 3)
3. Within a PAW letters are joined to each other, whether handwritten or printed
4. Very often PAWs overlap each other
5. Sometimes one letter is written beneath the one before it, like Lam-Ya (لي) or Lam-Mim (لم) or it seems to almost vanish away in middle Form like Lam-Mim-Mim (لمّ) (compares to Kaf-Mim-Mim (كمّ)), so in addition to the basic forms, there are also special forms which can be seen as exceptions
6. Some letters like Tha (ث), Ya (ي) or Jim (ج) have one to three dots above, under or within their 'body'
7. Some letters like Ba (ب), Ta (ط), Tha (ث) only differ because of these dots

In this paper, we propose a segmentation based technique for the recognition of handwritten Arabic words. The proposed technique accepts binary images of Arabic words as input and gives back a digital representation (so called Unicode) of the recognized letters as output. We segment PAWs into *candidate letters* as described in [1, 2]. These candidate letters are converted into their corresponding chaincode representation. Thereafter we represent each candidate letter by statistical and structural features extracted from the chaincode representation. For comparison, we also use the thinned image of the main body of a letter to extract statistical features proposed in [3]. Also we extracted new types of features, namely an Active Contour based feature and Chaincode Histogram based features from the candidate letters, which resulted in an increase of the recognition accuracy, even for strongly deformed handwritten Arabic letters. Furthermore, we use a neural network to assign weights for each feature representing the candidate letter and use this weighted feature vector as input for a k-NN classifier. Our experiments prove the feasibility of the proposed technique. The main contributions of the proposed technique are:

- Use of a decision tree to reduce the number of potential classes
- Use of a Chaincode Histogram and Active Contour features to classify the candidate letters
- Use of Neural Network assigned weights for each feature in a k-NN classifier to increase the accuracy of the word recognition.

The paper is organized as follow, Section 2 outlines existing approaches and problems concerning segmentation and recognition of Arabic Words. Thereafter we describe our own technique in Section 3 with focus on feature extraction. In Section 4 we discuss the recognition technique. In Section 5 we discuss the experimental results. Finally we conclude the paper in Section 6.

2 Related Works

In the published literature, approaches that addressing the problem can be classified into three main categories according to the different segmentation technique that is followed. The first category contains all approaches that completely ignore the segmentation, such methods called "holistic" based [4, 5] and [6] (Latin). Features are extracted from the word image as a hole and Hidden Markov Model (HMM) is employed as classifier. The authors reporting recognition rate of 90% of 937 different Tunisian town names, taken from the IFN/ENIT-database [7]. In [8] a couple of different features are examined for the holistic approach using this database.

Under the second category fall all approaches that apply an over-segmentation on the PAW, and then a margining strategy is followed in order to detect the optimal margining path [9, 10]. As an example for those approaches, Ding and Hailong [11] proposed an approach, in which a tentative over-segmentation is performed on PAWs, the result is what they called "graphemes", the approach differentiates among three types of graphemes namely (main, above, and under -grapheme). The segmentation decisions are confirmed upon the recognition results of the merged neighbouring graphemes; if recognition failed another merge will be tried until successful recognition. Also a HMM can be trained to handle segmentation [12]. The disadvantages of such approaches are the possibility of sequence errors and classification faults, as a result of the shape similarity between letters and fragments of letters.

The third category is what is called "explicit segmentation", in which the exact border of each character in PAW is to be found. The main features often used to identify the character's border are minima's points near or above the baseline. Shaik und Ahmed [13] proposed an approach that used some heuristic rules calculated upon the vertical histogram of the word's image. Though authors claim successfulness of their approach with printed text, they report failures cases when PAW contains problematic letters like Sin (س). In [14] a similar approach is tested for words of the IFN/ENIT-database (7.7% missed and 5.1% additional wrong segmentations).

Table 1. The Arabic Alphabet.

Letter	Isolated	End	Mid	Begin	Letter	Isolated	End	Mid	Begin
Alif	ا	ﺎ			Dhad	ﺽ	ﺾ	ﻀ	ﺿ
Ba	ﺏ	ﺐ	ﺒ	ﺑ	Taa	ﻁ	ﻂ	ﻄ	ﻃ
Ta	ﺕ	ﺖ	ﺘ	ﺗ	Dha	ﻅ	ﻆ	ﻈ	ﻇ
Tha	ﺙ	ﺚ	ﺜ	ﺛ	Ayn	ﻉ	ﻊ	ﻌ	ﻋ
Jim	ﺝ	ﺞ	ﺠ	ﺟ	Ghayn	ﻍ	ﻎ	ﻐ	ﻏ
Ha	ﺡ	ﺢ	ﺤ	ﺣ	Fa	ﻑ	ﻒ	ﻔ	ﻓ
Kha	ﺥ	ﺦ	ﺨ	ﺧ	Qaf	ﻕ	ﻖ	ﻘ	ﻗ
Dal	ﺩ	ﺪ			Kaf	ﻙ	ﻚ	ﻜ	ﻛ
The	ﺫ	ﺬ			Lam	ﻝ	ﻞ	ﻠ	ﻟ
Ra	ﺭ	ﺮ			Mim	ﻡ	ﻢ	ﻤ	ﻣ
Zai	ﺯ	ﺰ			Nun	ﻥ	ﻦ	ﻨ	ﻧ
Sin	ﺱ	ﺲ	ﺴ	ﺳ	He	ﻩ	ﻪ	ﻬ	ﻫ
Chin	ﺵ	ﺶ	ﺸ	ﺷ	Waw	ﻭ	ﻮ		
Sad	ﺹ	ﺺ	ﺼ	ﺻ	Ya	ﻱ	ﻲ	ﻴ	ﻳ
Tamabutra	ﺓ	ﺔ							

Our segmentation approach can also be categorized under this last category, since we are using topological features to identify the character border. The main problem with this category of segmentation is the varying of shape and topology within the single classes of handwritten Arabic letters. The feature extraction of the segmented letters [15] is the second important step of any recognition system. We discuss some different features in this paragraph which are applicable for the third category of segmentation.

3 Proposed Technique

Our proposed technique consists of 4 steps (see Fig. 1). The first step is the pre-processing of the input word image. In the second step, the pre-processed word is divided into PAWs and then each PAW is segmented into single candidate letters. Subsequently the feature extraction from these candidate letters is performed in the third step and the classification of letters is accomplished in the fourth step. An input to the first step is a word image and the output of the fourth step is a string of recognized Arabic letters.

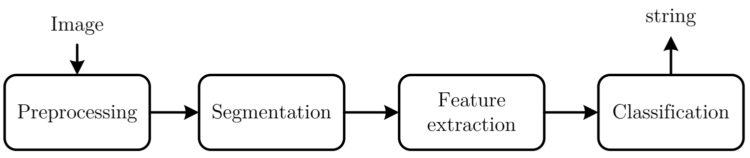


Figure 1. Overview of our proposed segmentation based system for Arabic handwritten recognition.

3.1 Pre-Processing

In the pre-processing phase noisy pixels are suppressed, in order to improve the gain of important information that will be processed in the subsequent recognition phases. Also some Arabic handwriting specifics pre-processing issues, like Baseline estimation are performed. To reduce noise and close little white gaps, that sometimes occur within strokes, we apply a 5x5 *median filter* on the image. Thereafter a global threshold is used to convert gray text images into *binary bitmap images*. To reduce the number of pixels to the minimum necessary for subsequent operations and also to ease the extraction of features, a *thinning* [16] version is created from the binary image.

3.1.1 Chaincode Representation

We get an initial contour *sequence* (data structure which contains neighbouring pixels of a word image) by following the contour of word in clockwise order from the highest point p_0 of a not-yet visited segment of a thinned image . For the purpose of feature extraction, we generate two representations for PAW's

3	2	1
4		0
5	6	7

Figure 2. Chaincode mask.

main contour, clockwise and counter-clockwise representations. They are generated for a PAW, in order to recover temporal information that can be used to generate a pseudo on-line version from the off-line one. The recognition of online version is proven to be more accurate and less expensive than in case of offline version. To generate the aforementioned temporal representations, we start first by calculating the pen-down point p_s and the pen-up point p_e Then starting from p_s , trajectory is traced until p_e in the clockwise direction, in order to generate the clockwise representation. The counter clockwise representation is generated in the same way, but by tracing the trajectory in a counter clockwise direction.

To achieve accurate classification, it is important to identify the pen-down (p_s) and the pen-up (p_e) of the PAWs in the sequence, and extract the sequence from pen-down to pen-end in clockwise and counter-clockwise directions. The pen-down point is usually the rightmost **EP** and the pen-up point the leftmost **EP** Key Feature in the segment. Such a sequence can also be converted to a list of numbers between 0 and 7 that represent the direction in which the successor is located This translation invariant representation called chaincode.

3.1.2 Baseline Estimation

Baseline estimation [17] is proved to be of critical importance to determine the position of possible ascenders and descenders of a letter, to select a minima to be a border point, and also to differentiate between diacritic dots according to their position from the baseline (above or under). To estimate the

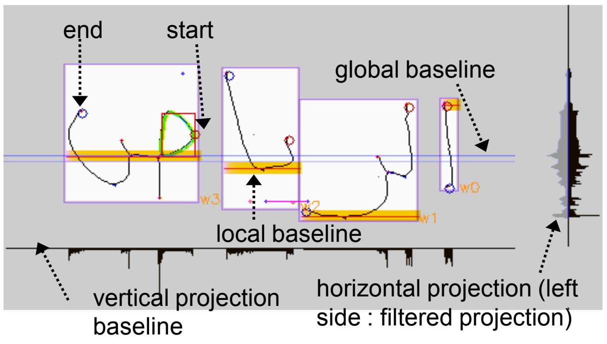


Figure 3. Example result after the pre processing. The second PAW from right creates a strong wrong peak in the horizontal projection of the word, but the filter dulls it sufficiently for correct baseline estimation.

baseline, we filter the horizontal projection with a (0.3, 0.7, 0.3) filter kernel and select the index with the biggest value. The method works fine for letters as can be seen in Fig. 3.

3.2 Feature Extraction

During this step several features are extracted from the candidate letter structures. The extracted features must be robust against variations of handwriting style. An example of the clockwise and counter clockwise sequences, which is stored for all letter prototypes, is shown in Fig. 4. Both of the sequences are stored XML files in addition to the position of KF points, the number of dots and loops and the global coordinates of the bounding box. For Classification we use mainly the following three different types of features.

3.2.1 Discrete Features

We define Discrete Features as set of features specific to the Arabic letters; namely the number and position of DPs, the number of LPs and existence of a hamza or stroke such as the fragment of Taa (ط). Those four features are employed in a pre-classification phase and lead to huge reduction in number of classes. Discrete Features f^d are Structural Features [18], which can described by a limited number of states $f^d \in (0, 1, 2, 3)$.

3.2.2 Statistical Features

Our Statistical Features [19] are extracted from the stored chaincode sequence representation. All statistical features of a letter can be represented as a simple vector $\vec{m} \in \mathbb{R}^n$; thus, a candidate letter can be easily and efficiently compared with the prototypes for each possible class.

We develop a set of statistical features called *chaincode histogram.* Histograms are generated as a result of counting the occurrence $N(i)$ for each possible digit $i \in \{0, 1 \ldots 7\}$in the chaincode (for the *clockwise* S_c and *counter-clockwise* S_{cc} order).

$$I(i) = \left(\sum_{k=0}^{7} n_k\right)^{-1} n_i \tag{1}$$

With a normalized histogram of each possible value of the chaincode, every $I(i)$ represents the intensity of a direction of the estimated trajectory, so Lam(ل) for example has a very high intensity for $I(6)$, Sin(س) have in contrast similar intensities for $I(6)$ and $I(4)$.

The invariant Hu *moments* are computed upon the sequences S_c and S_{cc} of the segmented letters. The moments are calculated as in [20] . An overview of various moments offers [21]. In order to normalize all statistical features, we find the maximal and minimal value for each feature, and normalizing according to $f(x) = {}^{x-\min}/_{\max - \min}$.

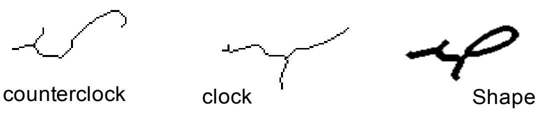


Figure 4. Stored sequences of an image instance of Sad (ص).

The last group of features is calculated based on the *Key Features* (KF) and the bounding box of the candidate letters. As very simple feature, we compute the slope of the line passing through the pen-up and the pen-down points. It is possible, to classify some handwritten letters like Alif(ا) in isolated form only with this feature because no other letter has such a high slope. The rest of features are the number of EP, LP, minima and the variance of the x and y coordinate of the EPs.

3.2.3 Structural Features

Structural features are more complex than statistical ones. Usually a model for each class of letter must be designed, that contains all significant information. The normalized chaincode is a typical structural feature that approximates the chaincode for clockwise S_c and counter-clockwise order sequences S_{cc} in u parts. To keep as much information as possible, convert each element $c \in \{0,\ 1, \ldots 7\}$ of the chaincode into a corresponding vector $\mathbf{c} \in \mathbb{R}^2$.

Our normalized chaincode must have fixed length of u in order to compare two codes component wise. To normalize a sequence of length n elements, where $n > u$, we compute $\triangle = \lfloor n/u \rfloor$. By summing $\triangle$vectors $\mathbf{c}_i$and normalizing the result we get:

$$\mathbf{v}_j = \frac{1}{\triangle} \sum_{i=\triangle \cdot j}^{\triangle \cdot (j+1)} \mathbf{c}_i \,, \quad cause\ \triangle = \left\| \sum_{i=\triangle \cdot j}^{\triangle \cdot (j+1)} \mathbf{c}_i \right\| \tag{2}$$

With $j \in \{0, 1, ...u-1\}$. We set $u = 10$ and compare the angle between every 10 vectors v_j of the candidate letter l and a prototype P and get the correlation error:

$$f_{ang} = \sum_{j=0}^{10} \sphericalangle \left(\mathbf{v}_j^l - \mathbf{v}_j^P \right) \tag{3}$$

Because the minimum angle is 0 and the maximum $180°$, f_{ang} can be easily normalized *after* the classification and in our approach the result f_{ang} can be used like a statistical feature. We also compute the difference of the position for every pair$\left(v_j^l, v_j^P\right)$.

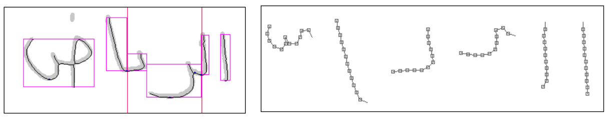


Figure 5. On the left: The segmented word "الريلض". On the right: the normalized chaincode (counterclockwise).

3.2.4 Deformable Models

The last structural feature is based on a modified *active contour*. To employ topological information in our model, we use an approximation that is based on the gradient, in order to keep significant information of Arabic letters like areas near a BP that can get lost by using a regular approximation such as normalized chaincode (see Fig. 5). But this irregular kind of approximation converts S_c and S_{cc} into polygons that mostly have different numbers of points, even within the same class. Therefore an Active Shape Model (ASM) cannot be created from such polygons; we decided to try a different approach to avoid losing information. Thus, we normalize the polygon, so that every element is now a image point $p \in \mathbf{B}^{b \times b}$and convert every p to $p \in \mathbb{R}^2$ with $p_x, p_y < b \wedge p_x, p_y \geq 0$.

Active Contour based Approach The following method uses the approximated sequences of a prototype for a deformable model. Before the basic algorithm can start, the model has to be initialized. Thereafter within several iterations, which consisting of a phase in which external forces deform the model followed by a phase in which the internal forces try to restore the original shape, the model adapt to the background, which is given by the sequences of a candidate letter.

The first initialisation of the model (concerning translation and scaling) is done by the normalisation $\Xi(S) = S \odot c$ of the contour for prototype P and candidate letter l. We use the approximated and normalized sequences S_c and S_{cc} to get P' and l':

$$\begin{aligned} &\forall p \in P' \wedge \forall q \in l' \left| P' \subseteq (P \odot c) \wedge l' \subseteq (l \odot c) \right. \\ &\Rightarrow p, q \in \mathbf{B}^{b \times b}. \end{aligned} \tag{4}$$

In some cases an advanced initialisation can be useful, so we move now the first point (*pen-down* point) p_s and the last point (*pen-up* point) p_e of P' (in S_c and S_{cc}) according the vectors $\mathbf{p}_{end}$ and $\mathbf{p}_{start}$to their corresponding positions in the background . We move all other points p_i by interpolating the vectors

$$\mathbf{t}_i = \lambda \mathbf{p}_{start} + (1 - \lambda) \mathbf{p}_{end} \tag{5}$$

where λ is inversely proportional to the distance of p_i to p_s. Now we have fixed the model on its start and end point. The technique can also adjust the orientation of the model and the background. As traditional

Active Contours (ACs) our models need the calculation of the so called Intern Energy and Extern Energy. Additional we use the intensity of the translation during the initialisation as energy. There are many ways to define the energies of a traditional AC, but our approach defers significantly because of the unusual application of our AC. The normalized sequences of l', which can be approximated in order to boost the algorithm, are used as fix background. The Extern Energy is based on two different forces, which depend on all points q_i of these sequences. We call the first force gravity. Similar to the physical gravity, we compute for all $p_i \in P'$ distance vectors $\mathbf{d}_{ij}$ to all $q_j \in l'$ that indicates the direction in which p_i is accelerated by q_j . The external force that influences a p_i can be computed by:

$$f_{ext}(p) = \sum_{j=0}^{N(l')} \mathbf{v}_j \left(1 - \frac{\|\mathbf{v}_j\|}{\sqrt{2b^2}}\right),\ \mathbf{v}_j = \begin{pmatrix} q_x^j \\ q_y^j \end{pmatrix} - \begin{pmatrix} p_x \\ p_y \end{pmatrix}, \tag{6}$$

where $N(l')$ is the number of elements in l'.

The second external force is more traditional. If a model point p is close to a point q, p will be slowed by a linear damping field of q that affects model points within a radius r. Without these forces the model points will oscillate around the background. We link all neighbour points with internal forces that build

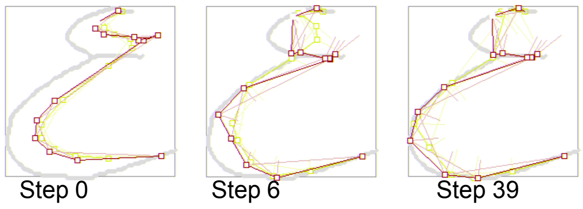


Figure 6. Example for the AC for the class Ayn(ع).

the *Intern Energy* with the intention to create a model that maintain its original form as good as possible after it is affected by the external forces. For all p_i with $0<i< n$ we link p_{i-1} and p_{i+1} in order to keep the gradient of the original model. All internal forces have an initial energy of 0.

After the external forces have deformed the model, the internal forces must be computed. These forces depend on the links between the model points. For every p_i we compute for all p_x, which are connected to p_i, the vector $v_{ix} = p_x - p_i$ and set

$$p_i{}' = p_i + c \sum \mathbf{v}_{ix}. \tag{7}$$

That means all model points try to restore their original relations. Moreover, we use a weak force that is directed on the original position of p_i.

To estimate the *correlation* of the model and the background we use three errors. The first error is caused by the initialisation, so we use the length of $\mathbf{p}_{end}$ and $\mathbf{p}_{start}$ as indicator for Err_{init}. The extern force creates a deformation almost proportional to the dissimilarity of model and background. So we use this deformation as Err_{intern}. The different forces are developed with the intention to create a model, which is able to adapt itself to variances of letter in the same class. At the same time, the model should not adapt to letters of other classes with different shape. Thus, we compare how similar the shape of the model with the background is after the last iteration. We use the distance between p and the next point q of the background, and compare the strengths of gradients because corresponding points should have a similar gradient. Now we get the last part Err_{extern} and can compute the total Err 8. We compare a candidate letter l' with models of all possible classes and match l' with the class that has the smallest Err.

$$Err = \lambda Err_{init} + \mu Err_{\text{int}\,ern} + \psi Err_{extern} \tag{8}$$

4 Recognition

After extracting all the features, the proposed technique starts the classification of candidate letters in three phases: pre-classification, classification and post-classification. In pre-classification phase, we use a decision tree (based on Discrete Features) to exclude some of the 28 Classes. Thereafter in the classification phase, we compute the probability that a candidate letter belongs to a class by comparing its feature vector with some prototypes. Finally, in the post-classification phase, we use a priori knowledge $P(C_x)$, which is the probability of occurrence of classes.

4.1 Pre-Classification

After the feature extraction we use the Discrete Features to reduce the number of potential classes the candidate letter may belong to. This pre-classification is very important, because in contrast to the Latin (English) alphabet only 9 of the 28 Arabic letters have a distinct shape and could be identified clearly without knowledge of the dots. Therefore, pre-classification saves us from computing statistical features from the dots to distinguish the different 28 letters. With pre-classification we can assign a letter to one of 12 groups where each group contains 1 to 10 classes. All members of the same group have the same number of loops and dots where the dots are on the same side of the baseline. That is a group G_i can be represented by a vector v_d which contains all Discrete Features. Our pre-classification assigns 70% of the letters to the right class. Reasons for failures are the occurrence of wrong positioning of dots in many handwritten words (usually shifted to the left) as well as the fact that a single dot and also a group of two dots are often written as horizontal stroke.

Further only the letters (ر, و, ز, ذ, ا, ر) always assume the end form or isolated form. This a priori knowledge allows us to reduce the number of classes. By using manual generated information about number and position of the dots in the prototypes, a recognition rate of 22% of all letters is achieved (without pre-classification only 3%).

4.2 Classification

Now the candidate letter must be compared with at least one prototype P of every potential class using a *k-Nearest-Neighbour* (k-NN) classifier. We assign weights to all statistic features f_i by computing the difference d_i between a letter l and all possible prototypes P and use them as input for a Single Layer Perceptron (SLP). We use the SLP to find sensible weights for all features due to the fact that not all features have the same validity. Thus, we compute the propagation function $p(\mathbf{d})$ of the SLP and subsequently use the sigmoid activation function $f(\mathrm{x})$:

$$\begin{aligned} x &= p(\mathbf{d}) = d_1 w_1 + d_2 w_2 + \ldots d_n w_n \\ f(x) &= \frac{1}{1 + e^{-(x+\theta)}} \quad |f(x) \in [0,1] \wedge x \in \mathbb{R} \end{aligned} \tag{9}$$

If the function $f(\mathrm{x})$ has a value near 1, the letter and the prototype seem to be very dissimilar ($\rightarrow$ rejection of the prototype). A value of 0.5 means, it's not possible to estimate if the letter has the same class as the prototype or not. Otherwise if $f(\mathrm{x})$ is almost 0, it is highly probable that they correlate. Since the output $f(\mathrm{x})$ is still unreliable, so we put all neighboring prototypes in a priority queue, using $f(\mathrm{x})$ as a key (see Table 4). Then, the class of the majority prototypes is selected.

4.3 Post-Classification

In order to minimize the risk of a wrong classification, we consult the a priori knowledge $P(C_x)$ that is the probability of occurrence for a class C_x. To get $P(C_x)$ we use our Training database with often used Arabic words and compute for each form (isolated, begin, etc.) the number $N(C_x)$ of occurring letters of a class C_x averaged over all letters of the specified form $N(\mathit{form})$

$$P(C_x, \mathit{form}) = \frac{N\left(C_x\right)}{N\left(\mathit{form}\right)} \tag{10}$$

We multiply $P(C_x)$ with the output of our classifier $f(\mathrm{x})$ (correlation of candidate letter with C_x). So we can increase the probability of correct classification, especially if a letter has nearly the same correlation with different classes, which corrects the results of the second phase of classification.

5 Experimental Results & Discussion

For our implementation we use C++ in combination with the image processing library OpenCV.

Because it's possible to summarize the features into the groups Chaincode Histogram (**CH**), Moments of Hu (**Hu**), Key Features (**KF**), Normed Chaincode (**NC**), Baseline (**BL**) and Active Contour (**AC**), we used a MLP, in order to evaluate the quality of these groups and compute corresponding weights (using a node with one input in the hidden layer for every group). We use pre classification based on manual generated Discrete Feature (**DF**) information before using any of other features. The results are shown in table 2. In order to get comparable results we built a database which will be available on http://www.iesk-ardb.ovgu.de soon. We built our own database of 323 different words. Every word is written by 12 subjects from Arabic speaking countries. In order to be able to use automatic training and validation techniques, we have built Ground Truth (GT) files for the database (training and test database).

For our experiments we used up to 40 samples of the first writer as prototypes for each letter class and tested them on 200 words (910 letters) of the second writer. The *Recognition Rate* means how many candidate letters are classified correctly by using the specified group of features. To compute the quality

Table 2. Characteristics of the feature Hu, B, CH,KF,NC and AC. In order to compare our results, the best features of [3] are tested using the same training and testing database.

Feature	Rec. R. in %	Intersection All	Intersection Group
Best of [3]	85.92		
Hu	68.4	60.2	52.8
BL	65.45		
KF	76.9	34.5	29.8
CH	86.47	32.1	27.3
NC angle	79.1		
NC eukl	79.76		
AC	80.39		
CH+B+KF	88.01 [1]		

of the features we use the *intersection* between the pair wise density distribution of two different classes C_x and C_y for every feature. The quality of a feature for classification is inversely proportional to its average intersection. We compute it for each of the 28 classes but also for only that classes, which are in the same

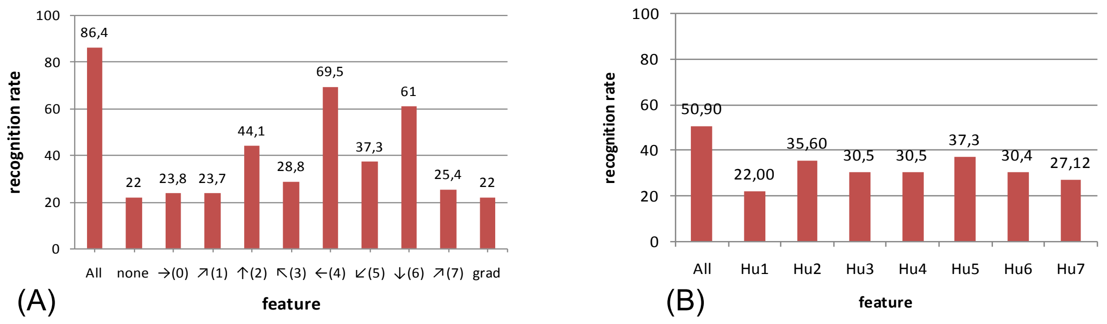


Figure 7. Recognition Rate (A)Chaincode Histogram, (B)Moments of Hu.

group, as the unknown letter (have the same Discrete Features). The average intersection of features within a group is a bit lower, because within a group there are no classes with the same main body.

As you see in Table 2, the simple Chaincode Histogram is unexpectedly the group of features with the best recognition rate. Therefore, Chaincode Histogram together with the KF based features complement each other to increase accuracy. We implemented the best 6 features which are originally used in [3] to recognize Farsi letters (with 85.59% accuracy). We tested them on our database and confirm the result (85.92%). Already with our Chaincode Histogramm features, we get a higher accuracy, although they are

[1]We get 91.65% by testing them on the words of the first writer (we delete the best letter candidate, the one which has exactly the same chain code as the letter), so we found that the features are only slightly writer dependant.

fewer and less complex features. If we also use the normalized distance of the highest point of the letters main body and the baseline as well as the KF features, we get an accuracy of 88.01%.

Fig. 7 (A) shows the results for classifying all words in the test database using all features of the Chaincode Histogram based on the pre-classification. Although the moments of Hu (Fig. 7) (B) are more complex features, none of the 7 Hu moments is as suitable as those of the chaincode histogram. Most significant for the tested Arabic handwritten letters are the feature "←" and "↓".

Summarizing we can say that our statistical features allow us to compare segmented letters relatively quickly and reliably with a huge amount of prototypes. Otherwise our structural features (especially the Active Contours) are a little disappointing, for they are less useful in spite of their high complexity. At least, we have shown that Active Contours are basically suitable for the recognition of Arabic letters, thus we are going to optimize them.

We tested a k-NN classifier with $k = 5$, because for some seldom classes we do not have more than 5 prototypes. In Fig. 8 you see that a higher value for k can be helpful using poor features, because the distance between the very best prototype sample and the letter could be large in feature space, even if their shapes are very close.

To show how sensible are the results of the classifier we conducted experiments to classify each of the 28 letters in isolated form and presented the results in a confusion matrix as shown in Fig. 9.

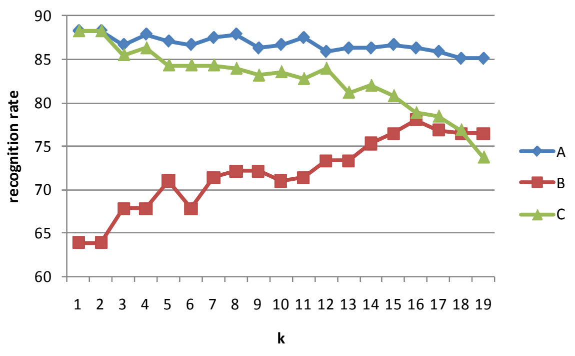


Figure 8. The Diagram shows the influence of the parameter k if the k-NN classifier is used. In experiments A and B the features CH, B and KF are used, but only up to 20 instead of 40 prototypes per letter class are allowed in B. In experiment C we allow again 40 prototypes but the baseline is the only feature.

Our proposed active contour (AC) based feature proves that it is useful to utilize the contours of Arabic letters to recognize deformed handwritten Arabic letters. We uses approximated polygons that obtain small structures like the right part of Sin (س) to increase the accuracy. But our experiments show that such increase in accuracy is minimal and not necessary. Because many Arabic letters exhibits dots, the pre -classification is so important. Our automatic pre-classification has poorer results than the manual pre-classification. This is because of the inconclusive notation especially regarding single dots or groups of two dots, in which both are shown as a stroke. Pre-Classification results with manually generated information (e.g. number of dots) for prototypes show a slight improvement in accuracy as compared to the automatic pre-classification. The last four rows in Table 4 show the accuracy of post-classification of different combinations of features. To show the effect of assigning weights to features on the accuracy of classification, we measured the accuracy of classification after assigning the weights. This is important, if the used features have a very different significance as the features of the Chaincode Histogram. Table 3 contains some well written words our system can segment and recognize successfully (except the last one that is an example for a problematic case).

6 Conclusion

We proposed a system that divides a handwritten Arabic Word into its PAWs, segments them into single letters and extract features from them. Then, it reduces the possible classes the candidate letters may

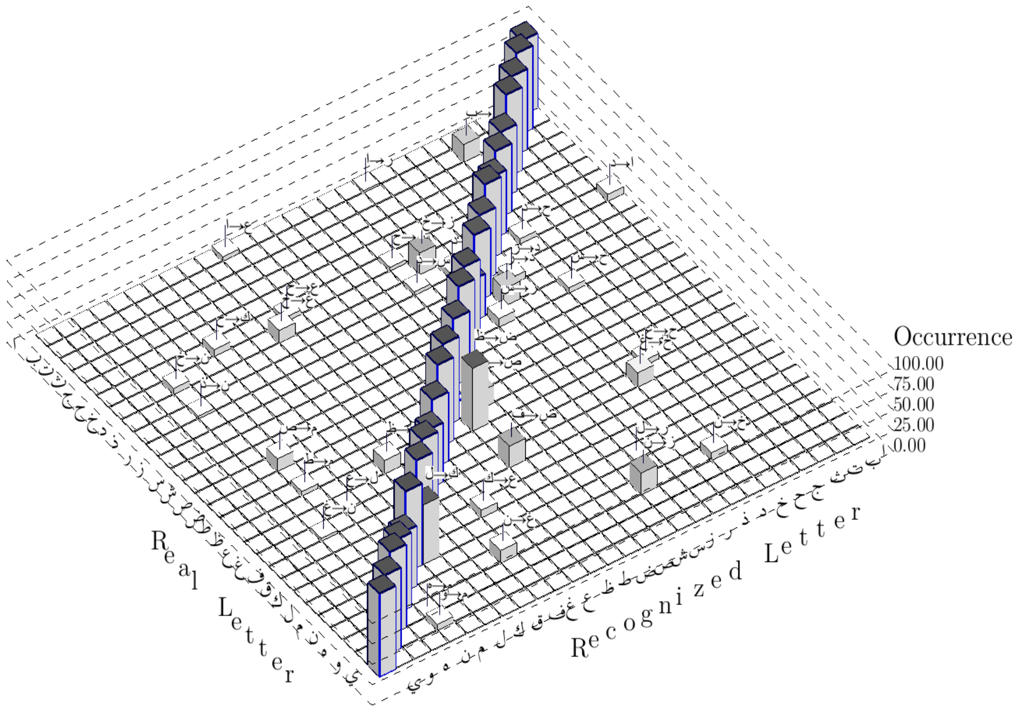


Figure 9. Confusion matrix of the 28 Arabic letters in isolated form for the features CH, KF and BL.

belong to, and classify them. Some novel approaches in this context are the Chaincode Histogram (CH) and the Active Contours (AC). We found, that the Chaincode Histogram is effective with a weighted NN-classifier. Although the shapes of Arabic letters are very complex, our extracted features are suitable for handwritten Arabic letters even for strongly deformed shapes. Besides recognition, methods to generate synthetic databases for the holistic approach [22] can be improved by Active Shape Models(ASMs) as we did in [23]. As a future work, we intend to use ASM also for classification. ASMs will extend AC with statistical information of a letter class that are extracted from many samples, whereby the high variance of handwritten letters should be compensated. Further we think it's necessary to compare the output of our system (Unicode) with a dictionary and compute the correlation with all correct, real words. Arica and Yarman-Vural did this for Latin characters [24] and could recognize 88.8% of all words with a dictionary of 40000 words, because most segmentation and letter-recognition faults could be corrected that way. If this kind of post processing is integrated in an application, a user would have the option to reduce the number of permissible words according to the current task (this causes not so much effort like making a database for the holistic approach). Furthermore the general probability that a word occurs in a text can be computed and

Table 3. Some results (including automatic segmentation and recognition of Arabic Words).

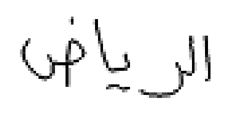
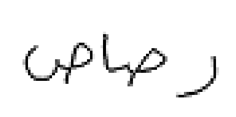
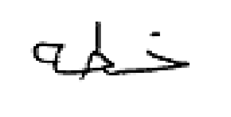
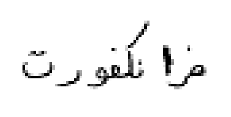
Image				
Rec. Word	الريَاض	رصَاص	خطة	ضَارنضمورت

Table 4. Some results (including automatic segmentation and recognition of Arabic Words).

Recognition Rate	Automatic pre classification.	Manual p.cl.	Weighted features	Post cl.
82.0041	x			
87.5256		x		
88.3436		x	x	
77.3082		x		x
92.229		x	x	x

used to reduce that number of permissible words automatically. The first international Arabic handwriting recognition competition [25], presented the results of systems proposed by five different research groups that are tested against the IFN-ENIT-database. A segmentation based approaches, achieved in the best case a recognition rate of 29.62% of the involved words. The best case achievement with the holistic approach was 74.69%. Important to note here, that even though the holistic approach as better results, it can only be applicable in closed domain application like bank, Post, and etc. To benefit from OCR systems in an unconstrained manner, it is crucial to solve the problem of segmentation based recognition.

References

[1] Moftah Elzobi, Ayoub Al-Hamadi, Laslo Dinges, and Bernd Michaelis. A structural feature based segmentation for off-line handwritten arabic text. In *5th International Symposium on Image/Video Communication over Fixed and Mobile Networks, ISIVC 2010 . - IEEE*, pages 1–4, (Rabat, Marocco),, 2010.

[2] Moftah Elzobi, Ayoub Al-Hamadi, and Zaher Al Aghbari. Off-line handwritten arabic words segmentation based on structural features and connected components analysis. In *In Gladimir Baranoski and Vaclav Skala, editors, WSCG ' 2011 Communication Papers Proceedings.*, 2011.

[3] Victor Lavrenko, Toni M. Rath, and R. Manmatha. Holistic word recognition for handwritten historical documents. In *Proceedings of the First International Workshop on Document Image Analysis for Libraries (DIAL'04)*, DIAL '04, pages 278–. IEEE Computer Society, 2004.

[4] Zaher Al Aghbari and Salama Brook. Hah manuscripts: A holistic paradigm for classifying and retrieving historical arabic handwritten documents. *Expert Syst. Appl.*, 36:10942–10951, 2009.

[5] Tal Steinherz, Ehud Rivlin, and Nathan Intrator. Off-line cursive script word recognition - a survey, 1999.

[6] Volker Maegner and Haikal El Abed. *Databases and competitions: strategies to improve Arabic recognition systems*, pages 82–103. Springer-Verlag, Collage Park, MD, USA,, 2008.

[7] Peter Burrow. Arabic handwriting recognition. Technical report, School of Informatics, University of Edinburgh,, 2004.

[8] Abdelmalek Zidouri. Oran system: A basis for an arabic ocr. *The Arabian Journal for Science and Engineering*, 31:91–100, 2005.

[9] Rashad Al-Jawfi. Handwriting arabic character recognition lenet using neural network. *Int. Arab J. Inf. Technol.*, 6(3):304–309, 2009.

[10] Xiaoqing Ding and Hailong Liu. Segmentation-driven offline handwritten chinese and arabic script recognition. In *Proceedings of the 2006 conference on Arabic and Chinese handwriting recognition*, SACH'06, pages 196–217, Berlin, Heidelberg, 2008. Springer-Verlag.

[11] Amjad Rehman, Dzulkifli Mohamed, and Ghazali Sulong. Implicit vs explicit based script segmentation and recognition: A performance comparison on benchmark database. *Int. J. Open Problems Compt. Math*, 2(3):352–364, 2009.

[12] N.A. Shaik, Z. Ahmed, and G. Ali. Segmentation of arabic text into characters for recognition. *IMTIC*, 20:11–18, 2008.

[13] Liana Lorigo and Venu Govindaraju. Segmentation and pre-recognition of arabic handwriting. In IEEE Computer Society, editor, *Eighth International Conference on Document Analysis and Recogni-*

tion (ICDAR'05), pages 605–609, Seoul, Korea,, 2005. Eighth International Conference on Document Analysis and Recognition (ICDAR'05).

[14] Francesco Camastra. A svm-based cursive character recognizer. *Pattern Recogn.*, 40:3721–3727, 2007.

[15] Jamshid Shanbehzadeh, Hamed Pezashki, and Abdolhossein Sarrafzadeh. Features extraction from farsi hand written letters. In *Proceedings of Image and Vision Computing*, pages 35–40, New Zealand,, 2007. Proceedings of Image and Vision Computing.

[16] L. Lam, S.W. Lee, and C.Y. Suen. Thinning methodologies-a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14:869–885, 1992.

[17] Majid Ziaratban and Karim Faez. A novel two-stage algorithm for baseline estimation and correction in farsi and arabic handwritten text line. In *ICPR*, pages 1–5, 2008.

[18] Saeed Al-Haj Ahmad. Recognition of on-line arabic handwritten characters using structural features. In *JOURNAL OF PATTERN RECOGNITION RESEARCH 1*, 2010.

[19] Anil K. Jain, Robert P. W., and Jianchang Mao. Statistical pattern recognition: A review. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 22(1):4–37, 2000.

[20] Jawad H. AlKhateeb, Jianmin Jiang, Jinchang Ren, Fouad Khelifi, and Stan Ipson. Multiclass classification of unconstrained handwritten arabic words. *The Open Signal Processing Journal*, 2:21–28, 2009.

[21] Richard J. Prokop and Anthony P. Reeves. A survey of moment-based techniques for unoccluded object representation and recognition. *CVGIP: Graph. Models Image Process.*, 54:438–460, 1992.

[22] Y.S. Elarian, H. A. Al-Muhsateb, and L. M. Ghouti. Arabic handwriting synthesis. In *First International Workshop on Frontiers in Arabic Handwriting Recognition (http://hdl.handle.net/2003/27562)*, 2011.

[23] L. Dinges, M. Elzobi, Z. Al Aghbari, and A. Al-Hamadi. Synthizing handwritten arabic text using active shape models, accepted by 3rd International Conference on Image Processing & Communications, 2011.

[24] Nafiz Arica and Fatos Yarman-Vural. Optical character recognition for cursive handwriting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(6):801–813, 2002.

[25] Liana M. Lorigo and Venu Govindaraju. Offline arabic handwriting recognition: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:712–724, 2006.