

# Arabic Handwriting Recognition Using Baseline Dependand Features and Hidden Markov Modeling

Ramy El-Hajj<sup>a,b</sup>, Laurence Likforman-Sulem<sup>c</sup>, Chafic Mokbel<sup>a</sup>

<sup>a</sup> University of Balamand, Faculty of Engineering, PoBox 100 Tripoli, LEBANON.

<sup>b</sup> Université de Reims, École Doctorale - Sciences Exactes, Moulin de la Housse, 51687 Reims, FRANCE

<sup>c</sup> GET-Ecole Nationale Supérieure des Télécommunications / TSI, 46, rue Barrault, 75013 Paris, FRANCE

## Abstract

*In this paper we describe a 1D HMM off-line handwriting recognition system employing an analytical approach. The system is supported by a set of robust language independent features extracted on binary images. Parameters such as lower and upper baselines are used to derive a subset of baseline dependent features. Thus, word variability due to lower and upper parts of words is better taken into account. In addition, the proposed system learns character models without character pre-segmentation. Experiments that have been conducted on the benchmark IFN/ENIT database of Tunisian handwritten country/village names, show the advantage of the proposed approach and of the baseline-dependant features.*

## 1. Introduction

The recognition of cursive handwriting is still an open problem due to the existence of many difficulties such as the variability of the handwritten styles and shapes, writing skew or slant and the size of the lexicon. Unconstrained Arabic handwriting is naturally cursive and challenging for off-line recognition systems. Letter shapes are context sensitive, inter and intra word spaces and character ligatures are of variable lengths, words include many dots and diacritical marks which change word meaning and indicate vowels. The writing is set on a baseline where character connections occur, and from where descending and ascending characters extend [1].

Arabic printed and handwriting recognition widely use HMM-based systems because of the connected nature of the Arabic script [3][5][6][7][10]. Even for analytical approaches, there is no need for character pre-segmentation for training and the recognition phase offers joint character segmentation and recognition. Moreover HMMs systems stochastically model sequences of variable length and cope with nonlinear distortions along one direction. HMM systems share a same theoretical framework but differ on the following

points: word and character graphical structures and parameters (ergodic, left-right / right-left structures and state skips allowed, number of states and mixtures, set of character models), by the way images are preprocessed (slant, skew, stroke width normalization) and on the choice of the feature set (statistical, structural, derivative). Feature sequences may be extracted from a sliding window or from grapheme segmentation.

In the present paper, we describe an off-line Arabic cursive handwritten word recognition system based on an analytical approach. Specific models are used for each character and word models are built by concatenating the appropriate character models. An overview of the proposed handwritten recognition system is shown in Figure 1. Language independent features are extracted from the word images after the baseline detection phase. This paper stresses the importance of the choice of the features, especially the introduction of the baselines in the performance of the system. From an original set of features, baseline dependent features have been added. These features are then passed to a general purpose HMM classifier, the HCM toolkit [8].

Section 2 presents the baseline detection phase. Then feature vectors are extracted using sliding window technique. A detailed description of the experimented features is given in Section 3. Section 4 details the HMM classifier as used in our system. The performance of the recognition system has been experimented on the benchmark database IFN/ENIT [9] which includes Tunisian city names. The obtained experimental results are shown and analyzed in Section 5. Finally, Section 6 draws some conclusions and perspectives.

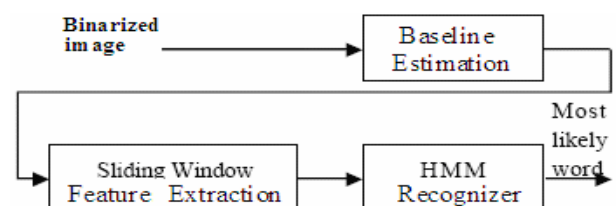


Figure 1. Block diagram of the recognition system

## 2. Baseline detection

Baseline extraction is generally used for skew normalization [10] or for word segmentation into characters [1][2]. Normalization may introduce image distortions. Therefore, we only use baseline position to extract baseline dependent features which emphasize the presence of descenders and ascenders. In Latin script, as well as in Arabic script, ascending and descending characters are salient characteristics for recognition. More precisely, we extract the lower and upper baselines of word images. These baselines divide the image into: a middle zone that doesn't contain ascenders and descenders, and 2 zones where ascenders and descenders can be found respectively. Figure 2 provides an example of extracted baselines.

Our approach is based on the algorithm described in [4] with few alterations. It is based on the horizontal projection curve that is computed with respect to the horizontal pixel density. First, the peak lower baseline, which corresponds to the maximum of the projection profile curve, is searched. This is justified with the Arabic written language where most of the letters have a lot of pixels on the lower baseline. Then, the algorithm scans the image from top to bottom to find the upper baseline, which corresponds to the first line with a projection value higher or equal to the average row density.

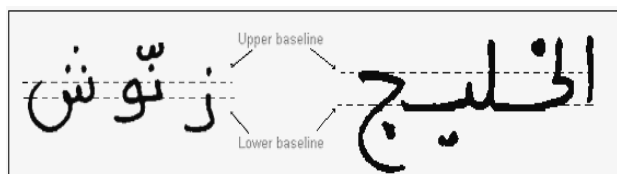


Figure 2. Upper and lower baselines on sample data

## 3. Feature Extraction

Classification performance largely depends on the quality and the relevance of the features extracted. The feature set has been chosen in order to capture the presence of ascenders, descenders and dots whatever their exact position in the word image. Concavity features are added to reflect local concavity and stroke directions. We also added a derivative feature, commonly used in speech recognition, as it partly overcomes conditional independence of observations.

In order to build the feature vector sequence, the image is divided into vertical overlapping windows (or frames). Window width is fixed and is a system parameter, while window height depends on each word image. The sliding window is shifted along the word image from right to left and a feature vector is calculated for each frame. The overlap is also a system

parameter which may vary from 0 pixel to *frame-width-1* pixels.

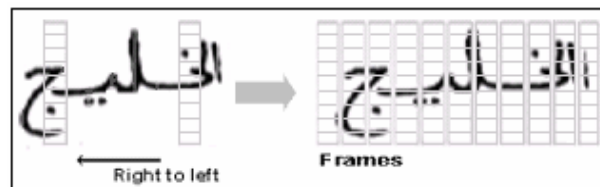


Figure 3. Word image divided into vertical frames (here without overlap)

The window height varies with every image. Then, each frame is divided into cells (Figure 3) where the cell height is fixed (in our experiments to 4 pixels). This yields to a variable number of cells in each frame according to the word image height.

We extract on each frame 24 features. There are two types of features: distribution features based on foreground (black) pixel densities, and concavity features. For each type of feature, a subset is baseline dependent.

### 3.1 Distribution features

Let  $H$  be the height of the frame in an image,  $h$  be the fixed height of a cell,  $w$  the width of a frame. The number of cells in a frame  $n_c$  is equal to:  $n_c = H/h$

Let  $r(j)$  be the number of foreground pixels in the  $j^{\text{th}}$  row of a frame,  $n(i)$  the number of foreground pixels in cell  $i$ , and  $b(i)$  the *density level* of cell  $i$ :

$$b(i)=0 \quad \text{if } n(i)=0 \text{ else } b(i)=1$$

Let  $L$  the position of the lower baseline,  $U$  the position of the upper baseline. For each frame  $t$ , the features are the following:

- density of foreground (black) pixels :

$$f_1 = \sum_{i=1}^{n_c} n(i)$$

- number of transitions between two consecutive cells of different density levels :

$$f_2 = \sum_{i=2}^{n_c} |b(i) - b(i-1)|$$

- $f_3$  is a derivative feature which represents the difference in  $y$  position of gravity centers of foreground pixels in the current frame  $t$  and in the previous one.

$$f_3 = g(t) - g(t-1)$$

Where position  $g$  is computed as (index  $t$  has been omitted):

$$g = \frac{\sum_{j=1}^H j \cdot r(j)}{\sum_{j=1}^H r(j)}$$

This derivative feature is inspired from  $\Delta$  coefficients used in speech recognition.

- 8 features (f4 to f11) that represent the densities of black (foreground) pixels for each vertical column of pixels in each frame (in our case the width of the frame is 8 pixels)

- vertical position of the center of gravity of the foreground pixels in the whole frame with respect to the lower baseline. The result is then normalized by the height H of the frame.

$$f_{12} = \frac{g - L}{H}$$

- 2 features that represent the density of foreground pixels over and under the lower baselines for each frame.

$$f_{13} = \frac{\sum_{j=L+1}^H r(j)}{H \cdot w}, \quad f_{14} = \frac{\sum_{j=1}^{L-1} r(j)}{H \cdot w}$$

- number of transitions between two consecutive cells of different density levels above the lower baseline

$$f_{15} = \sum_{i=k}^{n_c} |b(i) - b(i-1)|$$

Where k is the cell that contains the lower baseline.

- f16 represents the zone to which the gravity center of black pixels belongs with respect to the upper and lower baselines. Actually, the two baselines divide a frame into three zones: above upper baseline (f16=1), a middle zone (f16=2), and below lower baseline (f16=3).

### 3.2. Local Concavity features

Concavity features provide local concavity information and stroke direction in each frame. Each concavity feature f17 to f20 represents the (normalized) number of white pixels (background) that belong to 4 types of concavity configurations. They are explored by using a 3x3 window as shown in Figure 4.

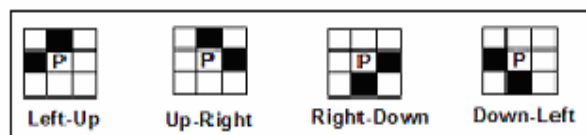


Figure 4. Four types of concavity configurations for a background pixel P

The concavity features are calculated as following. Let's,  $Nlu$ , (resp.  $Nur$ ,  $Nrd$ , and  $Ndl$  the number of background pixels that have neighbor black pixels in the two directions: left and up (resp. up-right, right-down, and down-left) in a frame. Thus, the four concavity features in each frame are defined as:

$$f_{17} = \frac{Nlu}{H} \quad f_{18} = \frac{Nur}{H}$$

$$f_{19} = \frac{Nrd}{H} \quad f_{20} = \frac{Ndl}{H}$$

By using the information coming from the detection of the two baselines of writing (upper and lower baselines), we have generated four new features describing the concavities in the core zone of a word, that is, the zone bounded by the two upper and lower baselines.

Let's d be the distance between the two baselines positions L and U:

$$d = U - L$$

Let's also,  $CNZ_{lu}$  (resp.  $CNZ_{ur}$ ,  $CNZ_{rd}$ , and  $CNZ_{dl}$ ) be the number of background pixels in the core zone that have neighbor black pixels in the configuration left-up (resp. up-right, right-down, and down-left). Thus, the four additional and baseline dependent concavity features related to the core zone are defined as:

$$f_{21} = \frac{CNZ_{lu}}{d} \quad f_{22} = \frac{CNZ_{ur}}{d}$$

$$f_{23} = \frac{CNZ_{rd}}{d} \quad f_{24} = \frac{CNZ_{dl}}{d}$$

This results in a total of 24-features vector per frame, 15 of them are baseline independent, whereas the 9 remaining ones are calculated with respect to baseline positions. It is to be noted that these features are convenient to any script that can be decomposed into three zones (body, ascending and descending zones) such as the Latin cursive script. This makes the feature extraction module language independent.

## 4. HMM-Based Classifier

Hidden Markov Models (HMMs) have been successfully applied to handwritten recognition systems. They offer several advantages, mainly the automatic training on non-segmented words (embedded training), and combined segmentation – recognition. Actually, with HMM modeling there is no need to apply any character-level pre-segmentation process. Pre-segmentation-free approach is important, especially in case of cursive handwritten, e.g. Arabic handwritten, where the characters are often connected.

The handwritten recognition system described in the present paper applies HMM modeling in the recognition (classification) module, where every character in the lexicon is represented with a right-left HMM. The HCM toolkit [8] has been used in our experiments.

## 4.1. Character and word modeling

Our approach is analytical and based on character modeling. Each character model has a left-right topology and parameters: number of hidden states, state transition probabilities and observation probabilities. Based on a cross validation methodology, we have found that a good choice for character HMM parameters is:

- 4 states for each character-model
- 3 transitions for each state: a self transition, a transition to the next state, and a transition that permits the skipping of a single state.
- a mixture of 3 Gaussian distributions for each state.

In order to model all the Arabic characters, we built up to 159 character models. In fact, an Arabic character may have different shapes according to its position in the word (beginning, middle, end word position). Other models are specified for spaces and characters with additional marks such as shadda ( ّ ).

Then, each word of the lexicon is built by concatenating the appropriate character models. A right-left character model is shown in Figure 5.

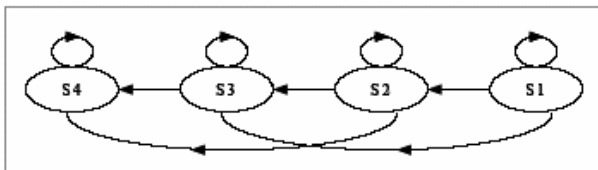


Figure 5. Right-left character HMM topology

## 4.2. Training and Recognition

The HMM-based classifier contains two modules:

- a training module.
- a recognition module.

The training module estimates character-HMM parameters on the training set of the database. A segmental Expectation Maximization (EM) algorithm is used for this purpose. The training is not performed on a character per character basis; rather whole words are used for this training while the character models are shared between the words' models. The training of HMM has no analytical solution since part of the observation is not measured. This is known as incomplete data problem. What is not measured is the state in the HMM when emitting a given character. To overcome the incomplete problem, the EM algorithm offers a suitable solution. It consists in iteratively estimating the model parameters. Each iteration is formed of two phases; the Estimate phase where the data are probabilistically completed using the parameters from the previous iteration, and the

Maximize phase where the parameters are re-estimated analytically because the data have been completed. In HCM a segmental version of the EM is implemented where in the Estimate phase the complete data are obtained in a hard association of the observation to the most probable states. This algorithm is also known as the Viterbi training. It is known that the EM algorithm offers a non-decrease of the likelihood with the iterations. A convergence towards a local optimum is therefore ensured. The quality of the convergence depends on the initial set of parameters. In HCM, initialization is obtained by linearly segmenting the observations to the HMM states and estimating the parameters based on this segmentation.

In the recognition phase, feature vectors extracted from an image are passed to a network of lexicon entries formed of character models. The character sequence providing the maximum likelihood identifies the recognized entry. To determine the most likely sequence of characters, the Viterbi algorithm is used.

## 5. Experiments and Results

### 5.1. IFN/ENIT Database.

To evaluate the performance of our recognition system, experiments are conducted on the benchmark database IFN/ENIT. The IFN/ENIT contains a total of 26459 handwritten words of 946 Tunisian town/villages names written by different writers and divided into four subsets a, b, c, and d [9].

We have noticed that the occurrence of examples in the IFN/ENIT database is different from one town/village name to another: they vary from 3 to 381. In these first experiments, we have considered a subset of the lexicon: the town/village names which occur more than 8 times in the whole database. So we have selected 21500 word images from the whole set, and the lexicon size was then reduced to 450 city names. We are presently conducting experiments for training the system on the whole database, in order to obtain complete results and participate to the ICDAR contest on the IFN/ENIT database.

### 5.2. Experiments and Results

In our experiments, we try to emphasize the importance of the features related to the estimated positions of the baselines. We have investigated two feature sets. Set F<sub>b</sub> includes the whole 24 features presented in Section 3, and set F<sub>w</sub> includes only 15 features: f1-f11, f17-f20. For every set of features four experiments are conducted by taking alternatively one subset of the database for testing and the remaining three ones for training. This leads to a non-parametric estimation of the error rate.

Table1 shows the experimental results of the performance evaluation of our recognition system using set Fw (only the 15 features) as a feature vector. This leads to an average recognition rate (resp. error rate) of 74.90% (resp. 25.10 %)

**Table1. Recognition results with feature set Fw (lexicon size 450)**

Test	Training data		Test data set		Rec. rate
	Data	Size	Data	Size	
1	b, c, d	16182	A	5358	<b>74.80%</b>
2	a, c, d	16019	B	5521	<b>75.33%</b>
3	a, b, d	16298	C	5242	<b>74.40%</b>
4	a, b, c	16121	D	5431	<b>75.41%</b>

In table 2 we provide the results after adding features relative to the detected baselines. The recognition rate (resp. error rate) increases (resp. drops) to 86.51 % (13.49 %). This shows that the features based on the baseline detection information bring a significant improvement (error rate reduced by 11.61%) to the recognition performance.

A preliminary error analysis shows different causes for wrong classifications according to the nature of the Arabic cursive handwritten words. Diacritical marks (dots) and extensions are often not at the exact position on top or under the main part of the letter. Besides, the descenders of a letter can be extended under one or more letters of the same word. Therefore, the vertical slicing approach will eventually split letters from their diacritical marks, and thus reduces the character recognition accuracy. In addition, some confusion errors are due to city names which have one word in common, as word images may include compound names (up to three ones). The lower baseline in such cases should be adapted to each word of the compounded name and not globally set.

**Table 2. Recognition results with feature set Fb (lexicon size 450)**

Test	Training data		Test data set		Rec. rate
	Data	Size	Data	Size	
1	B, c, d	16182	a	5358	<b>86.10%</b>
2	a, c, d	16019	b	5521	<b>86.88%</b>
3	a, b, d	16298	c	5242	<b>85.45%</b>
4	a, b, c	16121	d	5431	<b>87.20 %</b>

## 6. Conclusion and Perspectives

This paper presents a new HMM based system for offline handwritten recognition. Several features have been studied and compared. The importance of the detection of baseline position in the text image has been proved. The extracted features are based on the densities of foreground pixels, concavity and derivative features

in a sliding window. The extracted features are therefore script independent.

The developed system has been experimented and the results are provided on a subset of the benchmark database IFN/ENIT. These results show significant improvement of the recognition rate coming from using baseline dependent features. The accuracy of the baseline detection algorithm is quite satisfactory, and the recognition rates obtained so far are promising.

Many points are yet to be achieved such as taking into account the skew of the baseline in the feature extraction process, and adapting the lower baseline to each component of compound words. A major perspective consists in integrating of a post verification stage, in which re-ranking of the top hypotheses could be performed using the number of dots and connected components. Finally, we intend to apply our script independent features on a Latin handwritten database.

## References

- [1] Amin A. (1998), Off-line Arabic character recognition: the state of the art, *Pattern Recognition*, Vol. 31, 5, pp 517-530.
- [2] Arica N., Yarman-Vural F.T, (2002) Optical character recognition for cursive handwriting, *IEEE PAMI*, Vol. 24, 6, pp 801 – 813.
- [3] Ben Amara N., Belaïd A., Ellouze N. (2000), Utilisation des modèles markoviens en reconnaissance de l'écriture arabe état de l'art. In *Colloque International Francophone sur l'Écrit et le Document (CIFED'00)*, Lyon, France.
- [4] Blumenstein M., Cheng C. K., Liu X. Y. (2002), New Preprocessing techniques for Handwritten Word Recognition, *Proc. of the 2nd IASTED conference on visualization, Imaging and Image Processing*, pp. 480-484.
- [5] Khorsheed M.S. (2003), Recognizing Arabic manuscripts using a single hidden Markov model, *Pattern Recognition. Letters*, 24, pp. 2235-2242.
- [6] Makhoul J., Schwartz R., Lapre C., Bazzi I. (1998), A script independent methodology for optical character recognition, *Pattern Recognition*, vol. 31, no 9, pp. 1285-1294.
- [7] Miled H., Olivier C., Cheriet M., Lecourtier Y. (1997), Coupling observation/letter for a Markovian modelization applied to the recognition of Arabic handwriting, *ICDAR 97*, Ulm, pp 580 – 583.
- [8] Mokbel C., Abi Akl H. and Greige H. (2002), Automatic speech Recognition of arabic digits over the telephone network, *Proc. of Research Trends in Science and Technology RTST'02, Beyrouth*.
- [9] Pechwitz M., Maddouri S., Maegner V., Ellouze N. (2002), IFN/ENIT–DataBase for Handwritten Arabic words, *CIFED '02, Hammamet, Tunisia*, pp 129-136.
- [10] Pechwitz M., Maegner V. (2003), HMM Based approach for handwritten Arabic Word Recognition Using the IFN/ENIT– DataBase, *ICDAR '03, Edinburgh*, pp. 890-894.