

# Structural Features Of Cursive Arabic Script

Mohammad S Khorsheed  
Mohammad.Khorsheed@cl.cam.ac.uk

William F Clocksin  
William.Clocksin@cl.cam.ac.uk

Computer Laboratory, University of Cambridge, New Museum Site  
Cambridge CB2 3QG, United Kingdom

## Abstract

We present a technique for extracting structural features from cursive Arabic script. After preprocessing, the skeleton of the binary word image is decomposed into a number of segments in a certain order. Each segment is transformed into a feature vector. The target features are the curvature of the segment, its length relative to other segment lengths of the same word, the position of the segment relative to the centroid of the skeleton, and detailed description of curved segments. The result of this method is used to train the Hidden Markov Model to perform the recognition.

## 1 Introduction

The Arabic alphabet, as show in Table 1, is commonly used for writing many widespread languages (e.g. Arabic, Urdu, Persian), yet there is much less research in progress for recognising Arabic text than there is for Roman and Chinese text. One factor accounting for this is the characteristics of the Arabic alphabet which oblige researchers to examine some difficulties only recently being addressed by researchers on other languages [1]. Among these characteristics is the cursiveness of the script even in the machine-printed form.

To measure the performance of any Arabic text recognition system, we need to assess how successful the system is in overcoming the obstacles of cursiveness and context-sensitivity. The conventional approach is to segment the words into either characters [3, 4, 7] or symbols [2, 9, 6]. The first type of segmentation is the cause of recognition errors, and hence a low recognition rate. The second type is where a sub-word, or a primitive, is segmented into symbols where each symbol may represent a character, a ligature, or possibly a fraction of a character. The advantage of the second type over the first is that it is easier to find a set of potential connection points, than to find the actual connection points directly.

In this paper, we present another approach where the word is recognised as a single unit. This depends highly on a predefined lexicon which acts as a look-up dictionary. The procedure is to extract all segments from the skeleton of the word

## BMVC99

Char	Iso	Ini	Mid	End	Char	Iso	Ini	Mid	End
<i>a</i>	أ	أ	ا	ا	<i>b</i>	ب	ب	ب	ب
<i>t</i>	ت	ت	ت	ت	<i>t</i>	ث	ث	ث	ث
<i>ǧ</i>	ج	ج	ج	ج	<i>h</i>	ح	ح	ح	ح
<i>h</i>	خ	خ	خ	خ	<i>d</i>	د	د	د	د
<i>d</i>	ذ	ذ	ذ	ذ	<i>r</i>	ر	ر	ر	ر
<i>z</i>	ز	ز	ز	ز	<i>s</i>	س	س	س	س
<i>š</i>	ش	ش	ش	ش	<i>s</i>	ص	ص	ص	ص
<i>d</i>	ض	ض	ض	ض	<i>t</i>	ط	ط	ط	ط
<i>z</i>	ظ	ظ	ظ	ظ	ع	ع	ع	ع	ع
<i>ǧ</i>	غ	غ	غ	غ	<i>f</i>	ف	ف	ف	ف
<i>q</i>	ق	ق	ق	ق	<i>k</i>	ك	ك	ك	ك
<i>l</i>	ل	ل	ل	ل	<i>m</i>	م	م	م	م
<i>n</i>	ن	ن	ن	ن	<i>h</i>	ه	ه	ه	ه
<i>w</i>	و	و	و	و	<i>y</i>	ي	ي	ي	ي

Table 1: Arabic alphabet.

to be recognised, then transform each segment into a feature vector. Using Vector Quantisation (VQ) [8], each feature vector is mapped to the closest symbol in the codebook. This results a sequence of observations that is presented to a Hidden Markov Model (HMM) [11]. A number of reasons motivate the proposed technique. First, we wish to dispense with segmenting words into characters or other primitives. Next, extracting segments from the skeleton graph is more reliable than finding the actual connection points in the word. Finally, the extracted features are shape descriptors of the skeleton graph, so they provide a compromise between powerful discrimination and efficient extraction.

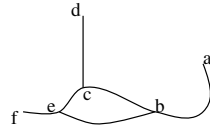
## 2 Preprocessing

The image of the word to be recognised is introduced to the system as a matrix of black pixels (foreground) and white pixels (background). Two preprocessing steps are then performed in sequence: thinning and centroid calculation.

The thinning method is based on Stentiford's algorithm [10]. The aim is to remove boundary pixels of the character that neither are essential for preserving the connectivity of the pattern nor do they represent any significant geometrical feature of the pattern. The process converges when the connected skeleton does not change or vanish even if the iteration process continues.

The purpose of centroid calculation is to find a reference point relative to which all segment locations are defined.

## BMVC99



$a \rightarrow b$	
$*b \rightarrow c$	$a \rightarrow b$
$*b \rightarrow e$	$*b \rightarrow e$
$c \rightarrow d$	$c \rightarrow d$
$*c \rightarrow e$	$e \rightarrow f$
$e \rightarrow f$	

Figure 1: segment and loop extraction. The segments:  $b \rightarrow c$ ,  $b \rightarrow e$  and  $c \rightarrow e$  are merged to form the complex loop  $b \rightarrow e$ .

## 3 Feature Extraction

This stage transforms the word to be recognised into a sequence of feature vectors. This is done in three consecutive steps: segment extraction, loop extraction and segment transformation.

### 3.1 segment Extraction

The skeleton graph of a word image consists of a number of segments where each segment starts and ends with a feature point. A feature point is a black pixel in the image which has a number of transitions from black to white pixels in the surrounding  $3 \times 3$  window equals to: one (*end point*), three (*branch point*), or four (*cross point*).

For the machine-printed fonts used in this paper, cross points appear only in the skeleton of a word if this word has the letter (*w, و*) in the middle (*يعود*) or at the end (*دعو*). However, these actually consist of three branches connected to the feature point: a loop and two other branches. This defines the cross point as a branch point and consequently leads us to say that: a segment may be incident between two end points, an end point and a branch point, a branch point and an end point, or two branch points.

An important requirement for segment extraction is to ensure that segments are assigned a canonical order, so that the observation sequence for the HMM is well defined. The rule is: all segments are listed in descending order relative to the horizontal value of the starting feature point of that segment.

### 3.2 Loop Extraction

During segment extraction, the skeleton is checked for loops, as seen in Fig.1. A number of Arabic letters have a loop-like shape. These loops can be divided into three categories: a *simple loop*, e.g. *ه ق ف*, consists of a single segment that starts from a feature point and returns to the same point again. A *complex loop*, e.g. *ص ه ط*, either consists of two segments, both having the same starting and finishing feature points, or three segments. A *double loop*, e.g. *ه ه*, consists of two connected simple and/or complex loops.

### 3.3 segment Transformation

Having extracted segments and loops from the skeleton graph of the word image, each segment is now transformed into an 8-dimensional feature vector. Each feature has the following description:

1. Normalised length feature ( $f_1$ ): This feature gives the length of a segment relative to other segment lengths in the same word. It is calculated as follows:

$$f_1 = \frac{ActualLength - EL_{min}}{EL_{max} - EL_{min}} \quad (1)$$

where  $EL_{min}$  and  $EL_{max}$  are the minimum and the maximum segment lengths for that word, respectively. This feature tolerate font size and rotation.

2. Curvature feature ( $f_2$ ): This feature measures the curvature of a segment. Simply divide the Euclidean distance between the two feature points of that segment by its actual length. This feature equals zero when the segment is a loop, and 1 when the segment is a straight line. Although this feature does not measure the curvature precisely ( $\subset$  and  $\supset$  have the same value), it is useful when combined with other features.
3. Endpoint feature ( $f_3$ ): This feature defines the two endpoints of the segment. It has one of the following values:

Value	Description
0	<i>end point → end point</i>
1	<i>end point → branch point</i>
2	<i>branch point → end point</i>
3	<i>branch point → branch point</i>

This feature can distinguish between similar segments belonging to different characters, e.g. the curve in  $\dot{\cup}$  and the last part of  $\dot{\cup}$  they are almost identical except that  $f_3$  equals zero and two, respectively.

4. Relative location feature ( $f_4$ ): This binary feature indicates whether the starting feature point of a segment falls above/below the centroid of the skeleton and shows this by one/zero. This feature helps deciding whether a dot is above or below the character which is considered a crucial decision, e.g.  $\dot{\cdot}$  and  $\dot{\cdot}$ .
5. Curved features ( $f_5 - f_8$ ): These features calculate the percentage of pixels above the top feature point, below the bottom feature point, left of the left-most feature point and right of the right-most feature point of that segment. The importance of these features are noticed when considering character such as  $\dot{\cup}$  and  $\dot{\cup}$ .

## BMVC99

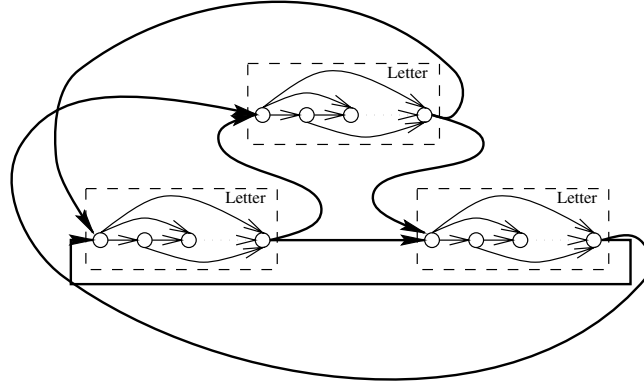


Figure 2: HMM topology .

## 4 Hidden Markov Models

### 4.1 Definition

The proposed method is based on a Hidden Markov Model [11]. An HMM may be represented by the parameter  $\lambda(\pi, A, B)$ , where :

$\pi = \{\pi_i = P(q_i \text{ at } t = 1)\}$ , initial state probability.

$A = \{a_{ij} = P(q_j \text{ at } t + 1 | q_i \text{ at } t)\}$ , state transition probability.

$B = \{b_i(k) = P(v_k \text{ at } t | q_i \text{ at } t)\}$ , observation symbol probability in state  $i$ .

$T$  = length of observation sequence.

$N$  = number of states in the model.

$M$  = number of observation symbols.

$Q = \{q_i\} \quad 1 \leq i \leq N$ , states.

$V = \{v_i\} \quad 1 \leq i \leq M$ , discrete set of possible symbol observations.

### 4.2 HMM Implementation

In this paper, we build only one model for all the words in the lexicon and use different paths, state sequences, to distinguish one pattern from the others. A pattern is classified to the word which has the maximum path probability over all possible paths. This approach is called *path discriminant* HMM [5]. A state may signify only one segment, and this segment represents a complete character (s), a fraction of a complete character, or touching characters (k̂). Accordingly, we did not use any optimisation criterion, such as the maximum likelihood (ML) [11]. The reason is that optimisation criteria produce a better model but do not preserve the correspondence of the states to individual characters which yields a lower recognition rate.

The HMM is formed from ergodic, fully connected, elementary units, as shown in Fig.2 . Each elementary unit represents a letter and is structured as a left-to-right HMM. As previously mentioned, a letter is decomposed into a number of

## BMVC99

segments. This number determines the number of states in an elementary unit.

An important step to put the HMM in practice is to estimate the model's parameters, but first we directly derive the dictionary statistics from the lexicon

$$Initial_\alpha = \frac{\text{No. of words starting with letter } \alpha}{\text{Total number of words in the lexicon}} \quad (2)$$

$$Trans_{\alpha \rightarrow \beta} = \frac{\text{No. of transitions from letter } \alpha \text{ to letter } \beta}{\text{Total number of transitions from letter } \alpha} \quad (3)$$

The initial state probability is computed as

$$\pi_i = \begin{cases} Initial_\alpha & \text{If } i \text{ is the } 1^{st} \text{ state in the elementary unit } \alpha \\ 0 & \text{Otherwise} \end{cases}$$

The state transition probability is calculated as

$$a_{ij} = \begin{cases} Trans_{\alpha \rightarrow \beta} & i \text{ is the last state in letter } \alpha \\ & \& j \text{ is the } 1^{st} \text{ state in letter } \beta \\ 0 & i \& j \text{ are middle states in different letters} \\ 0 & i \& j \text{ are in the same letter, } i \geq j \\ P(q_j \text{ at } t+1 | q_i \text{ at } t) & i \& j \text{ are in the same letter, } i < j \end{cases}$$

In our implementation, we use VQ to generate the symbol probabilities. VQ partitions the training samples into several classes in the Euclidean space using the K-means clustering algorithm. The symbol probability can be calculated as

$$b_i(k) = \frac{\text{No. of times in state } i \text{ and observing symbol } v_k}{\text{Total number of times in state } i} \quad (4)$$

Each word is represented by at least one path through the HMM. There are several possible ways to find the optimal state sequence associated with the given observation sequence. We use a modified form of the Viterbi algorithm [1] for finding the optimal path and some near-optimal paths. This procedure is stated below:

1. Initialisation  $1 \leq i \leq N$

$$\delta_1(i) = \pi_i b_i(o_1) \quad (5)$$

$$\psi_1(i) = 0 \quad (6)$$

2. Recursion  $1 \leq i \leq N, 2 \leq t \leq T$

$$\delta_t(i) = \max_{1 \leq j \leq N} [\delta_{t-1}(j) a_{ji}] b_i(o_t) \quad (7)$$

$$\psi_t(i) = \arg \max_{1 \leq j \leq N} [\delta_{t-1}(j) a_{ji}] \quad (8)$$

3. Termination

$$P^* = \max_{1 \leq j \leq N} [\delta_T(j)] \quad (9)$$

$$q_T^* = \arg \max_{1 \leq j \leq N} [\delta_T(j)] \quad (10)$$

## BMVC99

Font type	Percentage when recognised as the 1st choice	Percentage when recognised among the best 5 choices
<i>Simplified Arabic</i>	74%	97%
<i>Traditional Arabic</i>	68%	94%
<i>Arabic Transparent</i>	72%	95%

Table 2: System recognition rate.

$$\begin{aligned}
 &4. \text{ Path Backtracking} && T - 1 \geq t \geq 1 \\
 &&& q_t^* = \psi_{t+1}(q_{t+1}^*) \tag{11}
 \end{aligned}$$

For the sake of word recognition application, a considerable improvement can be made if more than just the first optimal path is recovered. The approach is to extend the  $\delta$  and  $\psi$  to another dimension which represents the choice  $W$ . Assume the model in state  $j$  at instance  $t$  then all the possible  $\delta_{t-1}(i, w)$  are considered and the  $W$  best paths are recorded in  $\psi_t(j, w)$  with their probabilities in  $\delta_t(j, w)$  where  $w = 1, 2, \dots, W$ .

## 5 Experimental Results

Images of the text were captured using a scanner with a resolution of 300 dpi. Each image passed the four-step processing sequence to be transformed into a sequence of observations. First, the thinning algorithm was applied to obtain the skeleton of the word, and the centroid of the skeleton was calculated. Secondly, segments were extracted from the skeleton graph in descending order relative to the horizontal value of the starting feature point. Then, each segment was transformed into an 8-dimensional feature vector. Thirdly, if two or more segments formed a loop then those segments were merged together in a single feature vector to be assigned a curvature value ( $f_2$ ) zero. Finally, VQ algorithm was used to form a codebook. This was done by partitioning the training samples into several classes. Each class was then represented by its class centroid. Each codebook symbol represented one class. The codebook included a total number of 76 symbols. Fig.3 shows an original image and the results of the first three steps of the processing sequence. Table 2 shows the recognition rate of the proposed system.

To assess the performance of the proposed method an HMM was trained using a 294-word lexicon. The samples were printed using three different fonts: Simplified Arabic, Traditional Arabic and Arabic Transparent. Samples used for training were not used during recognition. Table 3 shows the system output of three different samples representing the same word **أطلق**. The observation sequence for each sample differs from the others. Where the system output shows the same word more than once, this means the same word was recognised by a different path through the HMM. The HMM was not always able to list the correct word among the best five paths. An example of such a case may be seen in Table 4, in which a dot is missing owing to a problem with thinning. Sometimes, the HMM

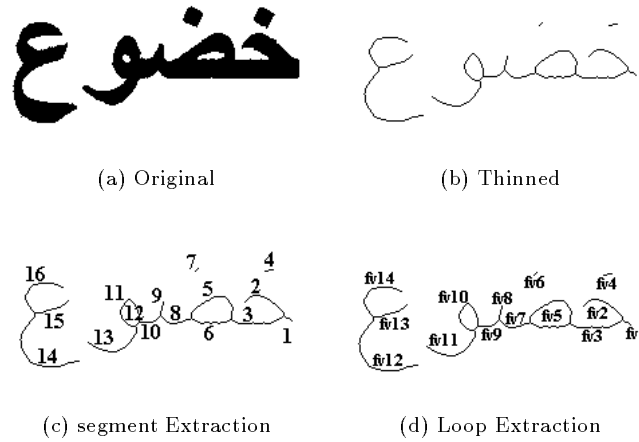


Figure 3: The processing sequence of an image word. The above image is transferred into the following observation sequence: 15, 24, 2, 4, 1, 4, 2, 10, 2, 1, 30, 47, 15, 31.

Font type	Obs. sequence	System Output	$P(\lambda O)$
Simplified Arabic	5, 41, 19, 22, 1, 36, 2 41, 22, 22, 17, 1, 66	إِطْلَاقٌ	$6.30 \times 10^{-13}$
		إِطْلَاةٌ	$1.67 \times 10^{-15}$
		إِطْلَاقٌ	$7.60 \times 10^{-19}$
		إِطْلَاقٌ	$2.10 \times 10^{-19}$
		إِطْلَاقٌ	$1.38 \times 10^{-19}$
Arabic Transparent	5, 42, 13, 22, 1, 36, 2 19, 22, 22, 21, 1, 66	إِطْلَاقٌ	$6.20 \times 10^{-12}$
		إِطْلَاةٌ	$4.12 \times 10^{-15}$
		إِطْلَاقٌ	$1.87 \times 10^{-18}$
		إِطْلَاقٌ	$8.44 \times 10^{-19}$
		إِطْلَاقٌ	$6.68 \times 10^{-19}$
Traditional Arabic	6, 36, 19, 22, 1, 19, 2 19, 2, 48, 22, 6, 1, 68	إِطْلَاقٌ	$8.00 \times 10^{-22}$
		إِطْلَاةٌ	$1.74 \times 10^{-22}$
		إِطْلَاقٌ	$7.30 \times 10^{-23}$
		إِطْلَاقٌ	$6.76 \times 10^{-23}$
		إِطْلَاقٌ	$1.94 \times 10^{-23}$

Table 3: System output of three different samples of the word إِطْلَاقٌ. Observation sequences are ordered from left-to-right.

## BMVC99

Word	System Output	$P(\lambda O)$
تَمَامًا	تَمَامًا	$1.99 \times 10^{-10}$
	تَمَامًا	$1.25 \times 10^{-10}$
	تَمَامًا	$3.12 \times 10^{-11}$
	تَمَامًا	$1.96 \times 10^{-11}$
	تَمَام	$4.34 \times 10^{-13}$

Table 4: An example of a word which was not recognised correctly.

Word	System Output	$P(\lambda O)$
تغذية	ثءدأتو	$3.58 \times 10^{-18}$
	تغذية	$1.67 \times 10^{-18}$
	ثءدأتبأ	$3.98 \times 10^{-19}$
	تغذية	$1.16 \times 10^{-19}$
	ثءدأآة	$9.61 \times 10^{-21}$

Table 5: System output of an example word.

throws up a sequence which was not included in the lexicon, as shown in Table 5. The first path ثءدأتو is not in the lexicon and it is not even an Arabic word. So, the second path تغذية which is in the lexicon was considered as the first option.

The last result concerns generalisation. Although it is difficult to predict in advance which untrained words the HMM will recognise, we have found a number of words recognised by the HMM, but were not among the training set nor the lexicon. An example word is shown in Table 6, and other words include إستعراض, عامل, بهاء, بن, بصر, إقتراح.

Word	System Output	$P(\lambda O)$
ثقبوب	ثقبوب	$9.34 \times 10^{-11}$
	ثقبوت	$1.56 \times 10^{-11}$
	ثقبوت	$1.07 \times 10^{-12}$
	ثقبوز	$1.26 \times 10^{-13}$
	ثقبوذ	$7.17 \times 10^{-14}$

Table 6: An example of a word which was recognised without a previous training.

## 6 Conclusion

We have proposed a technique for recognising Arabic words from digitised scans of script. The technique does not rely on segmentation into characters, but instead converts the skeletonised script into an observation sequence suitable for an HMM recogniser. A word model was trained from a 294-word lexicon acquired from a variety of script sources, and recognition rates of up to 97% were achieved. Future work will focus on increasing the number of fonts that can be recognised by the system.

## References

- [1] B. Al-Badr and S. Mahmoud. Survey and bibliography of arabic optical text recognition. *Signal Processing*, 41:49–77, 1995.
- [2] H. Al-Muallim and S Yamaguchi. A method of recognition of arabic cursive handwriting. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 9(5):715–722, 1987.
- [3] H. Al-Yousefi and S. Upda. Recognition of arabic characters. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 14(8):853–857, 1992.
- [4] A. Amin and J. Mari. Machine recognition and correction of printed arabic text. *IEEE Trans. on Systems, Man, and Cybernetics*, 19(5):1300–1306, 1989.
- [5] M. Chen, A. Kundu, and J. Zhou. Off-line handwritten word recognition using a hmm type stochastic network. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(5):481–496, 1994.
- [6] M. Fehri and M. Ben Ahmed. A new approach to arabic character recognition in multi-font document. In *The 4th International Conference and Exhibition on Multi-Lingual Computing*. Cambridge University Press, 1994.
- [7] H. Goraine and M. Usher. Printed arabic text recognition. In *The 4th International Conference and Exhibition on Multi-Lingual Computing*. Cambridge University Press, 1994.
- [8] R. M. Gray. Vector quantization. *IEEE ASSP Magazine*, 1:4–29, 1989.
- [9] K. Hassibi. Machine-printed arabic ocr using neural networks. In *The 4th International Conference and Exhibition on Multi-Lingual Computing*. Cambridge University Press, 1994.
- [10] J. R. Parker. *Algorithms For Image Processing and Computer Vision*. John Wiley and Sons, Inc., 1997.
- [11] L. Rabiner and B. Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, pages 4–16, January 1986.