# Recognition of multifont Isolated Arabic characters by Bézier Curves

A. Mazroui
Professor
Department of Mathematics and Computer Science
Faculty of Sciences, Oujda, Morroco

A. Kerkour Elmiad
PhDs
Department of Mathematics and Computer Science
Faculty of Sciences, Oujda, Morroco

## ABSTRACT

The recognition of Arabic characters is still a major challenge to overcome. In this paper, we propose a new approach in the field of recognition of multifont isolated Arabic characters. It is based on the semi-cursive nature of Arabic characters and consists in assimilating them to a small number of checkpoints equipped with their derivatives. The choice of this approach is motivated by the interesting properties of Bézier curves that allow drawing parametric curves from a limited number of dots equipped with their tangents. The results achieved are very interesting and relate to character recognition derived from a large number of fonts (23 fonts).

## 1. INTRODUCTION

The Arabic language is used by over 250 million people. The recognition of Arabic script, as in other writings, is an important tool in the development of new means of communication. It presents a crucial point for the development of human-machine communication [12]. The printed Arabic characters belong to the semi-cursive family. The large number of fonts for printed Arabic characters (over 450 fonts) and the different sizes generate a large number of morphological varieties for these characters. So, the recognition of printed Arabic script remains a significant challenge. Most of the works on the recognition of Arabic characters are inspired by methods used for Latin characters. Generally, these methods consist of detecting a small amount of information, called primitives, to describe the characters [12]. Indeed, in [6], the authors chose for primitives some data extracted from the outline of the character (Fourier descriptors and contour methods). Statistical methods are also used [2][8][9][14][17]. In other works, the authors used hybrid methods [5][11][18].
Our approach consists in choosing for primitives the dots where the shape of the character presents a significant variation (change of direction, inflection dots and cusps). These dots with their derivatives represent the primitives that will characterize the character. The choice of this approach is motivated by the possibility to approximate the shape of the character using only these primitives. Indeed, the theory of Bézier curves allows from data on a few dots (coordinates and derivatives) to draw shapes that conserve the properties of data [7].
The paper is organized as follows. In section 2 we survey some related work of Arabic character recognition systems. In section 3 we describe the preprocessing steps. Then, we present in section 4 a brief review of the Bézier curve theory related to our approach and we explain the procedure of feature extraction. Sections 5 and 6 are devoted respectively to the training and the recognition phases. Finally, we present in section 7 the experimental results and discussion, and we end this paper with a conclusion and a brief description of future works.

## 2. RELATED WORK

In order to achieve a multifont Arabic character recognition system, Ben Amor et al. [5] have used Hough transform for features and hybrid classifier based on hidden Markov chains and artificial neural networks. They noticed that the use of a hybrid approach at the classification level gives a better recognition rate compared with the approach only based on hidden Markov models. In 2008, H. Izakian et al.[10] have developed a recognition system for multifont Farsi/Arabic isolated character. They have used the chain codes in the classification phase and the one-nearest neighbour as classifier and obtained encouraging recognition rate. S.F. Bahgat et al. [4] examined the performance of four classes of statistical features: Moments Invariants, Gray Level Co-occurrence Matrix, Run Length Matrix and Statistical Properties of Intensity Histogram. Then, they studied the effect of merging two or more of these features on the recognition rate. They noted that the fusion of the moment features with those of Gray Level Co-occurrence Matrix provides a good recognition rate even for noisy images. M. Oujaoura et al. [15] have used the Zernike moments to extract features in order to recognize isolated Arabic characters. They conducted a comparison between the recognition rates relating to the use of Walsh transform and invariant moments as features. The obtained results with the Zernike moments are better than those concerning Walsh transform and invariant moments. Recognizing Urdu characters which derived from Arabic was the subject of a study by I. K. Pathan et al. [16]. They used the invariant moments in the feature extraction phase and SVM (Support Vector Machine) as a classifier. The results achieved are interesting.
In Section 7, we give a comparison between the recognition rates related to all these approaches.
A state of the art of the main methods developed in the field of OCR for Arabic was made by L. M. Lorigo et al. [12], and more recently by A. M. AL-Shatnawi et al. [1] and by A. Mesleh et al. [13].

## 3. PREPROCESSING

Image preprocessing is a significant part in Arabic character recognition. It consists to highlight the useful information contained in the input image and reduce or eliminate unnecessary information. Thereafter, the obtained image will be treated in order to extract the main features.

### 3.1 Separation of dots

As the number of pixels forming the dots is much smaller than those of the characters, we consider as dots any shape consisting of less than 100 pixels. By visualizing the different situations where the character is ac-companied by two or three dots (example: the character (Sheen)), we found that regardless of fonts, two dots have often one or two shared pixels (see Fig.1(a)). To remove these pixels, we identify the shared pixels by using the following masks:

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 1 | 1 | 1 |
| 1 | 0 | 1 |

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 0 | 1 |

Once these pixels are identified, we remove them (see Fig. 1(b)).

(a) Before separation (b) After separation

Fig. 1. (a)-(b): The character ش (Sheen) before and after separation dots.

After this step, the information that we keep are the number of dots and the character without dots.

### 3.2 Noise Removal

Given the nature of some Arabic script calligraphy, we noticed after the skeletonization phase (see paragraph 3.3 below) the emergence in the character skeleton of branches. These branches affect negatively the analysis, because they can be detected as primitives in the extraction phase of features. To avoid the appearance of these branches, we perform a filtering before skeletonization phase (see Fig. 2).

(c) Before filtering (d) After filtering.

Fig. 2. (a)-(b): The character ل (Lam) before and after filtering

### 3.3 Skeletonization

Among the fundamental problems of pattern recognition, we can mention the synthetic representation of these shapes. Indeed, the analysis and processing of the raw form of the characters are in many cases very expensive. The skeletonization consists in reducing the shape of the character to a curve centred on the original shape and called skeleton (see Fig. 3(b)). So, it is more beneficial in terms of time and efficiency to analyze a skeleton instead of the original shape.
From the algorithmic approaches to determine the medial axis of a binary form, we quote the topological slimming methods that are fast and give thin skeletons. That is why they are preferred in the character recognition. In terms of performance in computation time, the two fastest algorithms for the processing of Arabic characters are those of Deutsch and Zhang-Wang. The Zhang-Wang algorithm applied to Arabic is five times faster than that of Deutsch [3][19].

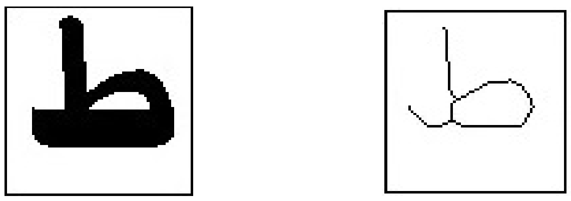

(a) (b)

Fig. 3. (a) The initial character ط (Thaa) ; (b) The character ط after skeletonization.

### 3.4 Straightening of shapes

Following the skeletonization step, we have noticed in some parts of the character skeleton the emergence of pixels shifted to a straight line (see Fig. 4(a)). So, we proceed with this step to a straightening of these pixels (see Fig. 4(b)).

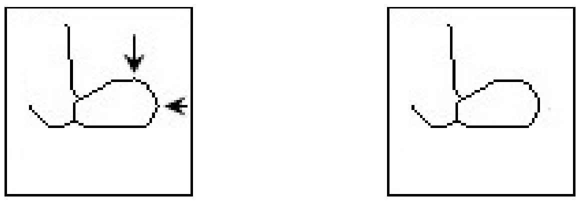

(a) (b)

Fig. 4. (a) The skeleton of ط before adjustment of shape ; (b) The skeleton of ط after adjustment of shape.

### 3.5 Adjustment and Resizing

To compare the features of character to recognize with those already learned, we must first unify the image size for all characters. We begin by framing the character (i.e. identify the smallest rectangle containing the character (see Fig. 5(a)). Then, we put this framed image in the center of a window $128 \times 128$ (see Fig. 5(b)).

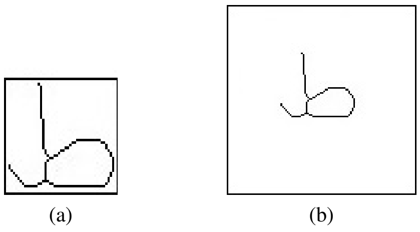


Fig. 5. (a) framed character ; (b) centered character in a window 128× 128.

## 4. BÉZIER FEATURES

### 4.1 Bézier Model

*4.1.1 Definition.* The Bézier curves have been introduced by Pierre Bézier. There exist many books that deal in detail these curves [7]. Bézier curves are parametric piecewise polynomial curves. They are easy to handle and have interesting properties for graphics [9]. Indeed,

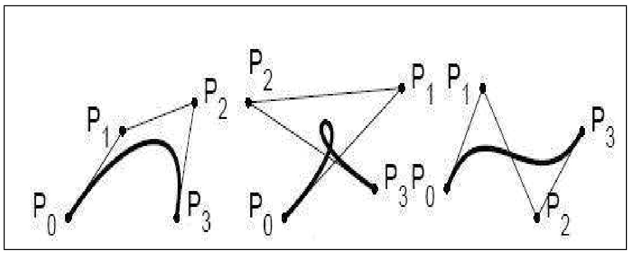


Fig. 6. According to positions of four dots $P_i$, we obtain various forms of *Bézier* curves.

(1) Given four dots $P_0$, $P_1$, $P_2$ and $P_3$, there exists a unique cubic Bézier curve which starts at $P_0$ and arrives at $P_3$, and has the vectors $P_0P_1$ and $P_2P_3$ as tangent vectors respectively at the dots $P_0$ and $P_3$. So, four dots are sufficient to define a cubic curve, whose shape is controlled by its envelope $P_0P_1$, $P_1P_2$ and $P_2P_3$ (see Fig. 6).
(2) By moving only one dot $P_i$, we obtain variations of the initial curve. This is used by typographers to refine the curves plotted.
(3) There is a simple and fast method for construction of these curves. It is a subdivision method based on the Theorem of De Casteljou.

These curves are easily connected together to form continuous splines of degree 3.

*4.1.2 Approximation of a character by Bézier curves.* To illustrate the method of approaching the shape of a character by spline functions, we will treat the case of the character ک (Kaf). As shown in Fig. 7, we can partition the shape of this character in several curves. The first one passes through $Q_1$ and arrives at $Q_2$, the second starts at $Q_2$ and arrives at $Q_3$, the third starts at $Q_3$ and arrives at $Q_4$ and the fourth starts at $Q_4$ and arrives at $Q_5$. By choosing the adapted tangents at dots $Q_i$ and using Bézier model to build a spline from these data, we obtain a shape close to that of the character.
Thus, a character can be characterized by a limited number of dots equipped with their tangents. The characteristic dots of a character

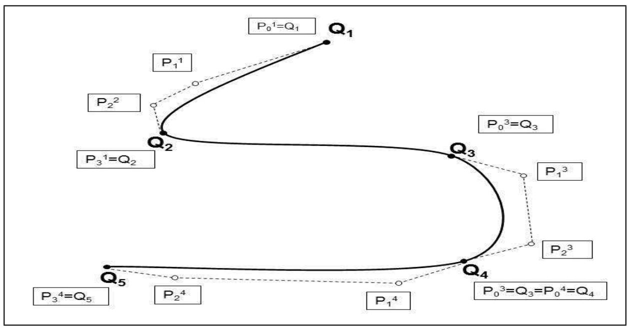


Fig. 7. The skeleton of the character ک.

are the extremities dots in addition to some dots where the shape of the character presents a variation (changes of direction, inflexion dots and cusps). All these dots with their tangents will represent for us the features which characterize the character.

### 4.2 Features extraction

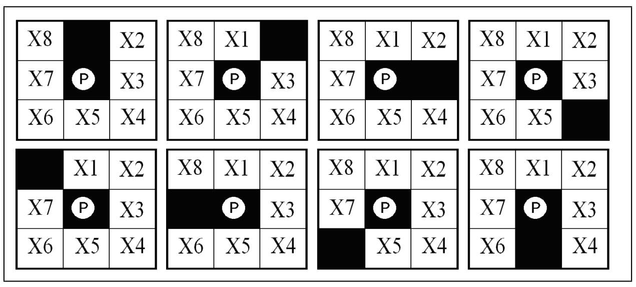
Once the steps of preprocessing are completed, we get a skeleton where the shape can be identified to a parametric continuous curve. To extract the features, we will base our analysis on the study of local variations of the pixels.

*4.2.1 Character Classes.* We distinguish two classes of the 30 Arabic characters:
- the class LC of characters that have a loop

{ه , م , و , ض , ص , ط , ظ , ة , ق , ف},

- and the class NLC of simple characters without a loop
{ا, ب, ت, ث, ن, ي, ل, د,ذ, ر, ز, ح, ج, خ, ع, غ, ک, ك, س, ش}.
The first class LC will be divided into three classes taking account the number of dots of the character:
$LC_0$ = {ط ,ص ,و ,م ,ه}, $LC_1$ = {ف ,ظ ,ض} and $LC_2$ = { ق ,ة}.
In the same, a second class NLC will be divided into four classes:
$NLC_0$ ={ا ,ل ,د ,ر ,ح ,ع ,ک ,ك , س}, $NLC_1$ = {ب,ن,ز,ج,خ,غ,ذ},
$NLC_2$={ ت ,ي} and $NLC_3$={ث ,ش }.
A given character $C$ belongs to the class LC only if we meet a pixel twice when we traverse its curve in a given direction.

*4.2.2 Detection of extremity dots.* There are no major problems in this step. Indeed, an extremity dot has a single neighbour pixel. Thus, the eight masks below can detect the extremity dots.



| | | |
|---|---|---|
| X8 | ■ | X2 |
| X7 | P | X3 |
| X6 | X5 | X4 |

| | | |
|---|---|---|
| X8 | X1 | ■ |
| X7 | P | X3 |
| X6 | X5 | X4 |

| | | |
|---|---|---|
| X8 | X1 | X2 |
| X7 | P | ■ |
| X6 | X5 | X4 |

| | | |
|---|---|---|
| X8 | X1 | X2 |
| X7 | P | X3 |
| X6 | X5 | ■ |

| | | |
|---|---|---|
| ■ | X1 | X2 |
| X7 | P | X3 |
| X6 | X5 | X4 |

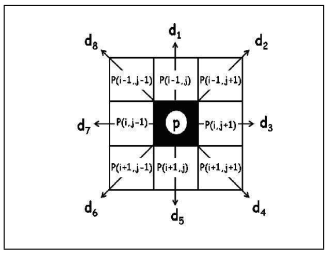
| | | |
|---|---|---|
| X8 | X1 | X2 |
| ■ | P | X3 |
| X6 | X5 | X4 |

| | | |
|---|---|---|
| X8 | X1 | X2 |
| X7 | P | X3 |
| ■ | X5 | X4 |

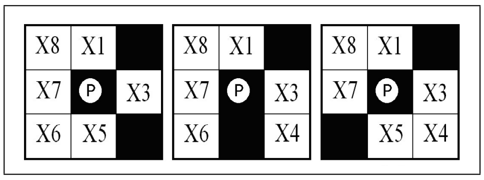
| | | |
|---|---|---|
| X8 | X1 | X2 |
| X7 | P | X3 |
| X6 | ■ | X4 |

### 4.3 Detection of singular dots

For each pixel, we consider the eight possible derivative directions $d_k$:



*Definition*: A pixel P is nonsingular if it is aligned with its two neighbor pixels, i.e. the two associate directions $d_i$ and $d_k$ of its derivatives are collinear, which is equivalent to $|i - k| = 4$ (the two first masks below are examples of masks used to detect singular dots and the last one is an example of masks used to detect nonsingular dots).



Our algorithm begins by removing the nonsingular pixels.

*4.3.1 Detection of characteristic dots.* The change in derivative directions of some singular dots is not necessarily due to a direction change in the shape of character, but a consequence of the digital image processing and the skeletonization algorithm. Thus, these dots do not necessary provide information on changes in the character shape. Hence we need a second filter to remove them. The idea is to eliminate any singular dot P that is almost aligned with its nearest neighbours $P_b$ and $P_a$, where $P_b$ (resp. $P_a$) is the singular dot detected just before (resp. just after) P. In the case where P is the first detected singular dot, the dot $P_b$ will be the first extremity dot and where P is the last detected singular dot, $P_a$ will be the last extremity dot. For a given character $C$ with a font f, we denote by $P_1$ and $P_m$ the two extremity dots and $(P_i)$, $2 \leq i \leq m-1$, the singular dots of $C$. Let $(u_i, v_i)$ the two derivative directions of the dot $P_i$, with the extension $u_1 = v_m = 0$.
**Lemma**
for each $2 \leq i \leq m-1$, the directions $v_{i-1}$ and $u_i$ are parallel and the directions $v_{i-1}$ and $v_i$ are not parallel.

Proof
Since $P_i$ is the first singular dot detected after the singular dot $P_{i-1}$, it is clear that the directions $v_{i-1}$ and $u_i$ are parallel. Since the directions $u_i$ and $v_i$ are not parallel (because $P_i$ is a singular dot), then the directions $v_{i-1}$ and $v_i$ are also not parallel.

Thus, for each singular dot $P_i$, if $P_{\hat{\imath}}$ is the singular dot judged as characteristic dot just before $P_i$, we have two cases:

(1) First case: the angle between the direction $v_i$ and one of the above directions $v_j$, $\hat{\imath} \leq j \leq i-1$, *is a right angle. In this case, the dot $P_i$ is a real cusp and will thereafter be considered as a characteristic dot of the character $C$.*

(2) *Second case: the angle between the direction $v_i$ and all previous directions $v_j$, $\hat{\imath} \leq j \leq i-1$, is a non-right angle (so it's equal to $135°$ or $180°$). In this case, the dot $P_i$ is a false cusp and should not be considered as a characteristic dot of the character $C$.*

The following algorithm shows how to detect the characteristic dots from the singular dots $P_i$:

**Algorithm to detect the characteristic dots from the singular dots**

```
1: The starting dot $P_1$ is considered as a characteristic dot
2:  i = 2
3:  î = 1
4:  while $i \leq m-1$
5:     for j=î : i-1
6:         if the angle between the direction $v_i$ and the
              direction $v_j$ is a right angle
7:             $P_i$ is a characteristic dot
8:             î=î+1
9:             i=i+1
10:        else
11:            i=i+1
12:        endif
13:    endfor
14:  endwhile
15: The last dot $P_m$ is considered as a characteristic dot
```

So, the features of the character $C$ with a font f are:

(1) the obtained characteristic dots $(Q_i)_{1\leq i\leq r}(r \leq m)$,
(2) the associated derivatives $(u_i, v_i)_{1\leq i\leq r}((u_i, v_i)$ are the two derivative directions of the dot $Q_i$),
(3) the associated information $(e_i)_{1\leq i\leq r}$ identifying the characteristic points that belong to loops ($e_i = 1$ if $Q_i$ belongs to a loop and $e_i = 0$ otherwise),
(4) the class affiliation $c$,
$(c \in LC_0, LC_1, NLC_0, NLC_1, NLC_2, NLC_3)$.

The features of the character $C$ with a font f are its class affiliation and the following characteristic matrix:

$$M = \begin{pmatrix} Q_1 & Q_2 & \dots & Q_r \\ u_1 & u_2 & \dots & u_r \\ v_1 & v_2 & \dots & v_r \\ e_1 & e_2 & \dots & e_r \end{pmatrix}$$

## 5. TRAINING

Let $f_1, \dots, f_n$ be n fonts of the same character $C$. We denote by $M_i$ and $c_i$, $1 \leq i \leq n$, respectively the characteristic matrix and the class affiliation of the character $C$ according to the font $f_i$. As the fonts are different, the obtained characteristic matrices $M_i$ have not necessarily the same size. So, a step for standardization of matrix sizes is needed.

### 5.1 Standardization of matrix sizes

Put $n^* = max\ n_i$ where $n_i, 1 \leq i \leq n$, is the number of columns of the matrix $M_i$ ($n_i$ = the number of characteristic dots of the character $C$ according to the font $f_i$). Let $M^*$ be one of the matrices $M_i$ such that its number of columns is equal to $n^*$. The standardization approach of sizes that we have adopted is to complete the matrix $M_i$ for which $n_i < n^*$ , by additional $(n^* - n_i)$ characteristic dots. For this, we denote by $N^*$ (resp. $N_i$) the sub matrix of $M^*$ (resp. $M_i$) formed by the first two lines of $M^*$ (resp. $M_i$) (it is obtained by keeping in $M^*$ (resp. in $M_i$) only the coordinates of characteristic dots). We begin by matching to each dot of the matrix $N_i$ the nearest dot of the matrix $N^*$ in the sense of Euclidean norm. Thus, it will remain $(n^* - n_i)$ dots in $N^*$ for which we have no counterpart in $N_i$. Then, for each of these $(n^* - n_i)$ dots in $N^*$, we seek the nearest dot of the character $C$ according to the font $f_i$. After, we add these dots with their associated derivatives and information relating to their belonging or not to loops to the matrix $M_i$, respecting the positions of corresponding dots in the matrix $M^*$ (see Fig. 5). So we get a new matrix $M_i^*$ of the same size as $M^*$.

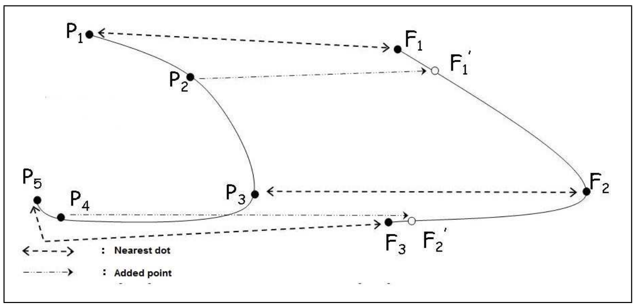


Fig. 8. Standardization of sizes applied to character د

### 5.2 features of the character

The characteristic matrices $(M_i^*)_i$ of the character $C$ according to the fonts $(f_i)_i$ are all of the same size $(5 \times n^*)$. Thus, the features of the character $C$ are:

(1) The characteristic matrix $M_C$ which is the mean of the matrices $(M_i^*)_i$:

$$M_C = \frac{1}{n}\sum_{i=1}^{n} M_i^*$$

(2) The class affiliation c of the character $C$ equal to the value $c_k$ which is the most common in the sequence $(c_i)_{1\leq i\leq n}$.

## 6. RECOGNITION

We first define the characteristic matrix $M_i$ of each character $C_i$ of the classes LC and NLC defined in Paragraph 5.1. The recognition of an unknown character $C$ will occur in four steps.

### 6.1 Class affiliation and characteristic matrix of the character $C$

By following the steps developed in Subparagraph 4.2, we get class affiliation and the characteristic matrix $M_C$ of the character $C$. Afterwards, the matrix $M_C$ will be compared to the characteristic matrices $M_i$ of the characters $C_i$ belonging to the same class as the character $C$.

### 6.2 Distance between $M_C$ and characteristic matrix $M_i$ of the character $C_i$

To measure the distance between $M_C$ and the characteristic matrix $M_i$ of a character $C_i$ belonging to the same class as $C$, we must first standardize their sizes. For this, we distinguish three cases on the number of columns $n_C$ and $n_i$ of matrices $M_C$ and $M_i$ respectively.

*6.2.1 First case: $n_C = n_i$.* We ordain the columns of the matrix $M_i$ so that the $j^{th}$ column of the obtained matrix is closest to the $j^{th}$ column of the matrix $M_C$.

*6.2.2 Second case: $n_C < n_i$.* We first ordain the columns of the matrix $M_i$ so that the $j^{th}$ column, for $j \leq n_C$, of the obtained matrix is closest to the $j^{th}$ column of the matrix $M_C$. Then, we complete the matrix $M_C$ by $(n_i - n_C)$ additional dots. To do it, we proceed as provided in Subparagraph 5.1. Thus, the obtained matrix is the same size as $M_i$.

*6.2.3 Third case: $n_i < n_C$.* As in Subparagraph 6.2.2, we ordain the columns of the matrix $M_i$ so that the $j^{th}$ column, for $j \leq n_i$, of the obtained matrix is closest to the $j^{th}$ column of the matrix $M_C$. As $M_i$ is the characteristic matrix of $C_i$ according to several fonts (this is the mean of characteristic matrices $(M_{ij})_{1\leq j\leq r}$ of the character $C_i$ according to the r fonts $(f_j)_{1\leq j\leq r}$ used in training phase), we first complete each matrix $M_{ij}$ by $(n_C - n_{ij})$ additional dots by following the steps of the Subparagraph 5.1 ($n_{ij}$ is the number of columns of $M_{ij}$). After, we substitute the matrix $M_i$ by the matrix $M_i^*$ mean of these matrices which is the same size as $M_C$.
Once the step of standardization of matrix sizes is complete, we denote $M_i^*$ and $M_C^*$ the obtained matrices, and we put $d_i = \|M_i^* - M_C^*\|$ where $\|\ \|$ is the Frobenius norm.

### 6.3 Identification of the character C

The character $C$ will be identified with the character $C_i^*$ satisfying the following minimization equation:

$$d_{i^*} = \min_i d_i.$$

The minimum is taken for all characters belonging to the same class as the character $C$.

## 7. EXPERIMENTAL RESULTS

The experimental results depend on the one hand, on the used fonts in the training phase and on the other hand on their number. In our study, we limited ourselves to the twenty-three most widely used fonts that are:

(1) *Naskhi*
(2) *ALAWI*
(3) *AL-jass*
(4) $AF\_Asee$
(5) *ACS Almass*
(6) $AF\_Jizan$
(7) $AF\_Buryidah$
(8) $AF\_Jeddah$
(9) *ACS Zomorrod*
(10) *Sultan rectangle*
(11) *MCS Alhada Out*
(12) $ArabicKufiSSK17$
(13) *ACS Almass Extra Bold*
(14) $AF\_Najed$ *Normal Traditional*
(15) *AF_Tabook Normal Traditional*
(16) *MCS Kofy5 S_U normal*
(17) $AF\_Jeddah$ *Normal Traditional*
(18) *MCS Alhada* $S\_Unormal$
(19) *MCS Hor 8* $S\_INormal2000$
(20) *MCS Hor 8* $S\_UNormal2000$
(21) *MCS Mozdalifa S_U norma*
(22) $AF\_AbhaNormalTraditional$
(23) $AF\_Ed$ *Dammam Normal Traditional*

The first question we have studied is to identify the fonts that used in the training phase give the best recognition rate in the testing phase. To achieve this, we have tested as training fonts all combinations of k fonts, k ¡ 23. We noticed that the obtained results are not interesting when we have used in the training phase more than five fonts (i.e. k ¿5). So, we give the results only for $k \leq 5$.
We have displayed in Table 1 the results for the two combinations of k fonts that used in the training phase give the best rate. For each value $1 \leq k \leq 5$, the first column of Table 1 presents the k fonts used in training step. We give in the second column the training rate $Tr\_Rat$ (the recognition rate of $30k$ characters related to the k fonts used in the training phase). In the third column we give the testing rate $Te\_Rat$ (the recognition rate of $30(23 - k)$ characters related to the $(23 - k)$ fonts not used in the training phase). We give in the fourth column the recognition rate $R\_Rat$ relating to all the 23 fonts. We have displayed in Table 1 only the results for the k fonts that used in the training phase give the best rate.
The experiments allowed to highlight that the best performances in the test phase are obtained when the training process uses three fonts that are ACS Zomorroda, ACS Almass and AF Buryiada. The results show the interest of this approach. Indeed, the recognition rate achieved is very high (around 99%), and it concerns the recognition of characters written in several fonts (23 fonts). This multifonte recognition is one of the major advantages of this approach given that most of the previous works are limited to an unifonte recognition or a very limited number of fonts [5][10][15][16].
In Table 2, we report the recognition rate for each of the thirty characters of the Arabic language after using Zomorroda ACS, ACS and AF Almass Buryiada fonts in the training process. We recall that the testing was done on 23 samples for each of these characters (23 samples are obtained by writing the character according to the 23 fonts).
In Table 3, we present the recognition rate of some character recognition systems developed recently by several research teams. Since

Table 1. Recognition rate according to the number of fonts k used in the training phase

| Recognition accuracy for k= 1 | | | |
|---|---|---|---|
| Training fonts | Tr_Rat | Te_Rat | R_Rat |
| ACS Zomorroda | 100% | 82.50% | 83.26% |
| ACS Almass | 100% | 79.66% | 80.54% |
| AF Buryidah | 100% | 73.33% | 74.49% |
| Recognition accuracy for k= 2 | | | |
| Training fonts | Tr_Rat | Te_Rat | R_Rat |
| ACS Zomorroda and ACS Almass | 100% | 96.00% | 96.35% |
| ACS Zomorroda and AF Buryidah | 100% | 95.66% | 96.04% |
| ACS Almass and AF Buryidah | 100% | 95.33% | 95.74% |
| Recognition accuracy for k= 3 | | | |
| Training fonts | Tr_Rat | Te_Rat | R_Rat |
| ACS Zomorroda, ACS Almass and AF Buryidah | 100% | 98.83% | 98.98% |
| ACS Zomorroda, ACS Almass and Alawi | 100% | 98.33% | 98.55% |
| Recognition accuracy for k= 4 | | | |
| Training fonts | Tr_Rat | Te_Rat | R_Rat |
| ACS Zomorroda, ACS Almass, AF Buryiada and Alawi | 100% | 97.83% | 98.21% |
| ACS Zomorroda, ACS Almass, AF Buryiada and AL-jass | 100% | 97.36% | 97.82% |
| Recognition accuracy for k= 5 | | | |
| Training fonts | Tr_Rat | Te_Rat | R_Rat |
| ACS Zomorroda, ACS Almass, AF Buryiada, Alawi and AL-jass | 98,67% | 97.03% | 97.39% |
| ACS Zomorroda, ACS Almass, AF Buryiada, Alawi and Naskhi | 100% | 96.48% | 97.25% |

Table 2. Recognition rate per character

| Letters | Recognition rate | Letters | Recognition rate |
|---|---|---|---|
| ا | 100% | غ | 95.65% |
| ب | 100% | ک | 100% |
| ت | 100% | ك | 100% |
| ث | 100% | س | 100% |
| ن | 100% | ش | 100% |
| ي | 95.65% | ه | 100% |
| ل | 95.65% | م | 95.65% |
| د | 100% | و | 100% |
| ذ | 100% | ض | 95.65% |
| ر | 95.65% | ص | 95.65% |
| ز | 100% | ط | 100% |
| ح | 100% | ظ | 100% |
| ج | 100% | ة | 100% |
| خ | 100% | ق | 100% |
| پ | 100% | ف | 100% |
| Average | | | 98.98% |

Table 3. Recognition rate on different recognition systems

| System | Recognition rate |
|---|---|
| Ben Amor et al [5] | 97.36 |
| Izakian et al [10] | 97.40 |
| Bahgat et al. [4] | 99.40 |
| Oujaoura et al. [15] | 98.27 |
| Pathan et al. [16] | 93.59 |
| Our system | 98.98 |

the tests carried out on these systems were not on the same test database, it would be prudent to take the conclusions of comparison with a certain reserve.

## 8. CONCLUSIONS AND PROSPECTS

We developed in this paper an approach based on the properties of Bézier curves for classification. The tests allowed to highlight that the best performances are obtained when the training phase uses three fonts that are ACS Zomorroda, ACS Almass and AF Buryiada. The tests were carried out on a very large sample of fonts (23 fonts), and the results are very interesting since the recognition rate is around 99%. This shows the interest and the consistence of this approach. Now, we are currently adapting this system and testing it on the Arabic handwriting characters and numerals. The first results are very encouraging. In future work, we plan to use the same concept for the recognition of printed Arabic words.

## 9. REFERENCES

[1] A.M. AL-Shatnawi, S. AL-Salaimeh, F. H. AL-Zawaideh, and O. Khairuddin. Offline arabic text recognition-an overview. *World of Computer Science and Information Technology Journal (WCSIT)*, 5(1):184–192, 2011.

[2] J. H. AlKhateeb, J. Ren, J. Jiang, and H. Al-Muhtaseb. Offline handwritten arabic cursive text recognition using hidden markov models and re-ranking. *Pattern Recognition Letters*, (32):1081–1088, 2011.

[3] D. Arrivault. *Apport des Graphes dans la Reconnaissance Non-Contrainte de Caractres Manuscrits Anciens*. PhD thesis, Universit de Poitiers, 2006.

[4] S. Bahgat, S. Ghomiemy, S. Aljahdali, and M. Alotaibi. A proposed hybrid technique for recognizing arabic characters. *International Journal of Advanced Research in Artificial Intelligence*, 4(1):35–43, 2012.

[5] N. Ben Amor and N. Essoukri. Multifont arabic character recognition using hough transform and hmm/ann. *Journal of Multimedia*, 2(1):50–54, 2012.

[6] T. S. Elsheich and R. M. Guindi. Automatic recognition of isolated arabic characters. *Signal Processing*, 14(2):177–184, 1980.

[7] G. Farin and P Massart. *Curves and Surfaces for Computer Aided Geometric Design*, volume 3368/2005. Academic Press - San Diego, 1983.

[8] A. Hassin, X. Tang, J. Liu, and W. Zhao. Printed arabic character recognition using hmm. *J. Comput. Sci. Technol*, 19(4):538–543, 1980.

[9] A.A.-M. Husni, A.M. Sabri, and S.Q. Rami. Recognition of off-line printed arabic text using hidden markov models. *Signal Processin*, 12(88):2902–2912, 2008.

[10] H. Izakian, S. A. Monadjemi, B. Tork Ladani, and K. Zamanifar. Multi-font farsi/arabic isolated character recognition using chain codes. *World Academy of Science, Engineering and Technology*, 43:67–70, 2008.

[11] A. Kacem, A. Belad, and M. Ben Ahmed. Automatic extraction of printed mathematical formulas using fuzzy logic and propagation of context. *International Journal on Document Analysis and Recognition*, 2(4):97–108, 2001.

[12] L.M. Lorigo and V. Govindaraju. Offline arabic handwriting recognition: A survey. *IEEE Trans. Pattern Anal. Machine Intell*, 28:712–724, 2006.

[13] A. Mesleh, A. Sharadqh, A. Al-Azzeh, J. Abu-Zaher, M. Al-Zabin, N. Jaber, Odeh, and T. Hasn. Offline arabic handwriting recognition: A survey. *Contemporary Engineering Sciences*, 11(5):521–529, 2012.

[14] N. Mezghani, A. Mitiche, and A. Cheriet. Bayes classification of online arabic characters by gibbs modeling of class conditional densities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(7):1121–1131, 2008.

[15] M. Oujaoura, R. El Ayachi, M. Fakir, Bouikhalene B., and B. Minaoui. Zernike moments and neural networks for recognition of isolated arabic characters. *International Journal of Computer Engineering Science*, 3(2):17–25, 2012.

[16] I. K. Pathan, A. A. Ali, and R. J. Ramteke. Recognition of offline handwritten isolated urdu character. *Advances in Computational Research- Intl. Journal*, 1(4):117–121, 2012.

[17] R. Prasad, A. Bhardwaj, K. Subramanian, H. Cao, and P. Natarajan. Stochastic segment model adaptation for offline handwriting recognition. In *In: Proc. Internet. Conf. Pattern Recognition*, pages 1993–1996, 2010.

[18] I. Zavorin, E. Borovikov, E. Davis, and A. Borovikov. Combining different classification approaches to improve off-line arabic handwritten word recognition. In *Proc. of SPIE 6815 (681504)*, 2008.

[19] Y.Y. Zhang and P.S.P. Wang. A new parallel thinning methodology. *International Journal of Pattern Recognition and Artificial Intelligence*, (8):999–1011, 1994.