Isolated Arabic Characters Recognition, Using a Robust Method Against Translation, Noise and Scaling, Based on the «Hough Transform»

Mohammed Kadi
Laboratory MATSI, EST.
Mohamed I University.
Oujda, Morocco.
kadi.m926@gmail.com

M'barek Nasri
Laboratory MATSI, EST.
Mohamed I University.
Oujda, Morocco.
nasrihome@gmail.com

***Abstract*—This work proposes a structural approach to extract the features of isolated Arabic characters, based on the «Hough Transform». This approach supports multiple fonts and resists noise, scaling, and translation. It is consists first in detecting loops and dots in the structure of the entered Arabic character and classifying it accordingly. Then, the «Hough Transform» is used to detect the longest lines in the character structure, in different directions, and according to a well-chosen threshold. Afterward, we improved this method, by introducing another threshold to consider also the shortest lines in the character structure. A remarkable recognition rate has been achieved for fonts already learned or not in the learning phase. We verified by many tests that the method is invariant by translation and that it is not too much affected by the noise and scaling.**

***Index Terms*—Image processing, Arabic characters recognition, feature extraction, Hough Transform.**

I. Introduction

OPTICAL Characters Recognition (OCR) is a computer technology that allows the software to recognize and transform printed or handwritten text in an image into a treatable text file or document. This technology is urgently needed for many governments and commercial departments.
Optical character recognition for the Arabic Language is one of the most important research topics related to the development of the use of the Arabic language. Here are some points about the importance of an OCR of Arabic:

- Department and institutions within the Arab world still use paper documents in their dealings with customers (certificates, contracts, buy and sale agreements, marriage, birth and death certificates, Bank checks, letters, etc.). It is, therefore, necessary to scan, recognize and archive the contents of these documents. Firstly, to preserve them from natural factors (fires, floods, etc.). Secondly, to explore and search their contents easily with a click of a button. And thirdly, to reduce storage space, for example, instead of storing 1,000 image files, we store 1,000 text files.
- There are ancient Arabic manuscripts in various fields. The digital recognition of the contents of these manuscripts and their archiving will help the process of fast searching, comparison, solving the mysterious things and revision. In 2018, at a scientific conference, D. Fassi Fihri spoke about: the digitization of ancient Arabic manuscripts challenges and opportunities [1].

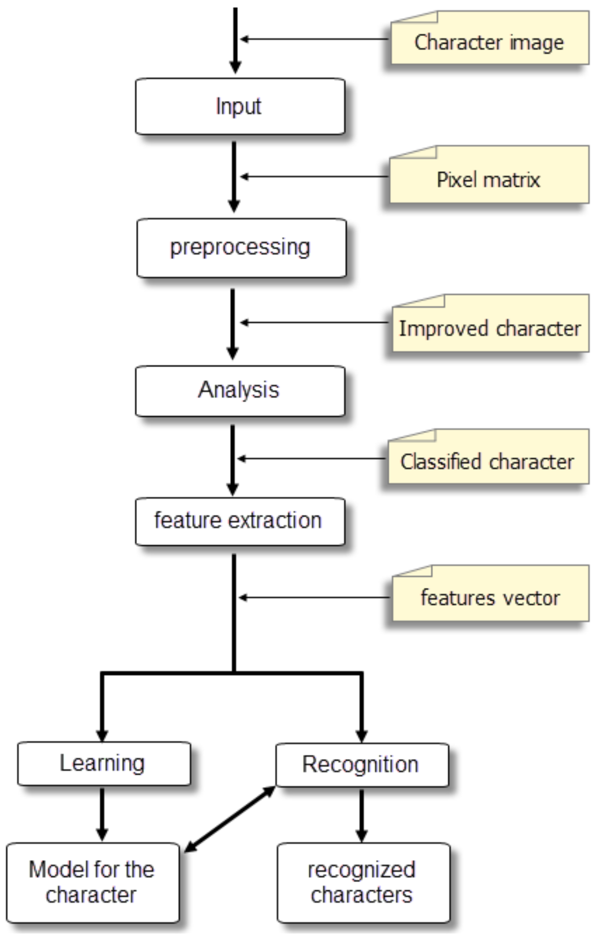


Fig. 1. Our proposed system.

• There are many written exams and competitions organized in different Arab countries to obtain a certificate, to go from one level of education to another, or to find a job. Thus, the development of a digital system which allows recognizing the paper answers of these exams and competitions, and corrects them with precision accordingly, will reduce the cost and save time.

• Improving the performance of smart devices and robots to recognize Arabic writing (e.g., recognition of traffic-sign and plaques in Arabic, for automatically help or alert the driver), will be in line with the current technology, and preparation for the future technologies.

• To help blind people to read: e.g., the blind can use a device with a built-in OCR system that can scan the open page of a book, extract the text, and send it to an integrated text-to-speech system (TTS) which in turn transforms this text into a voice that the blind finally recognizes by hearing.

There is much progress in Latin and Chinese character recognition techniques, while Arabic character recognition systems still have not produced satisfactory results yet. This is due to various problems posed in the case of Arabic characters:

• Arabic script is cursive, i.e. in a word, the characters can be connected without spaces between them, as in the case of the word «كتب».

• Most Arabic characters are associated with complementary parts above «ق», below «ب» or inside the character «ك». Some of these parts, like the dots, are used to differentiate similar Arabic characters as in the case of «ج» and «خ».

• Most Arabic characters have different shapes depending on their position in the word, as is shown in Tables I. In addition to other shapes, when there is a union of more than one character, such as: {ﻻ ﻼ ﻷ ﻸ ﻹ ﻺ ﻵ آ ـآ}.

• There are numerous Arabic fonts with wide variability.

• Some forms of Arabic characters are common to many fonts.

In our view, the most difficult stage faced by Arabic character recognition systems is the process of segmentation of connected characters in the word (or part of the word) due to the complex cursive nature of Arabic writing as we have explained before. If the segmentation process is not done correctly, it will greatly affect the accuracy of recognition. Therefore, the segmentation of Arabic characters is another field of study that cannot be covered in this paper. This work focuses only on the recognition of isolated Arabic characters, i.e. with the assumption that the process of segmentation was done well.

First, and as part of preprocessing and analysis of the image, we present two techniques: One for the image enhancement of the Arabic character, and another for determining some information on its structure. This will allow the first classification of the Arabic character.

Next, in order to generate the feature vector, we use the «Hough Transform» to obtain the long lines most representative of the structure of the Arabic character according to a threshold that has been well chosen. And after

TABLE I
SHAPES OF ARABIC CHARACTERS

Name	Isolated	Shape 1	Shape 2	Shape 3	Shape 4
Hamza	ء	أ / إ / ـأ / ـإ	ؤ / ـؤ	ئـ / ـئـ / ـئ	ء
Alif	ا	ا	ـا	ـى	
Ba	ب	ـب	ـبـ	بـ	
Ta	ت	ـت	ـتـ	تـ	ة
Tha	ث	ـث	ـثـ	ثـ	
Jim	ج	ـج	ـجـ	جـ	
Ha	ح	ـح	ـحـ	حـ	
Kha	خ	ـخ	ـخـ	خـ	
Dal	د	د	ـد		
Thal	ذ	ذ	ـذ		
Ra	ر	ر	ـر		
Zai	ز	ز	ـز		
Sin	س	ـس	ـسـ	سـ	
Shin	ش	ـش	ـشـ	شـ	
Sad	ص	ـص	ـصـ	صـ	
Dhad	ض	ـض	ـضـ	ضـ	
Taa	ط	ـط	ـطـ	طـ	
Dha	ظ	ـظ	ـظـ	ظـ	
Ayn	ع	ـع	ـعـ	عـ	
Ghayn	غ	ـغ	ـغـ	غـ	
Fa	ف	ـف	ـفـ	فـ	
Qaf	ق	ـق	ـقـ	قـ	
Kaf	ك	ـك	ـكـ	كـ	
Lam	ل	ـل	ـلـ	لـ	
Mim	م	ـم	ـمـ	مـ	
Nun	ن	ـن	ـنـ	نـ	
He	ه	ـه	ـهـ	هـ	ه
Waw	و	و	ـو		
Ya	ي	ـي	ـيـ	يـ	

some tests, we made improvements to this method, by introducing another threshold.
Finally, we have tested the recognition rate and the sensitivity of the method we propose to noise, scaling, translation, and rotation.
The proposed system has three main steps, preprocessing, Analysis, and features extraction. The full system is shown in Fig. 1.

II. STATE OF THE ART

A state of the art of the main methods developed in the field of Arabic character recognition was done by A. Lawgali [5], and more recently by K. M. Jambi et al. [6] and by F. M. A. Nashwan et al. [7].
Among the recent interesting works, a work done by H. Althobaiti et al. [8] where they describe a novel approach to recognize the handwritten isolated Arabic characters, by using the Freeman chain code.
With regard to our proposed method, Fakir, et al. [9] proposes to use the Hough transform for printed Arabic character recognition. A recognition rate of 95% was achieved with 300 characters obtained from the segmentation process. They concluded that the Hough transform is computationally simpler compared to the Fourier transform.
S. Touj et al. [10] proposes to use a Generalized Hough transform for printed Arabic character recognition. With a set of 234868 characters in their isolated form, and in three different fonts: Arabic Transparent, Andalus and Traditional Arabic; the recognition rate obtained was around 97%. But, they noticed that the enlarging of the dictionary by adding characters from other fonts, will increase computational costs, therefore, a preliminary detection of fonts proves to be necessary.
N. B. Amor et al. [11] have used Hough transform with Hidden Markov Models and Artificial Neural Networks. Their system was implemented with a set of 85000 samples in five different fonts and the overall recognition rate was 97.36%. They noted that the use of a hybrid approach at the classification level will increase the recognition rate compared to the approach based only on hidden Markov models.
The Persian characters share many features with the Arabic characters. In 2014, N. A. Khuzani et al. [12] presented a method for detecting the Persian handwritten characters by using the Hough transform. With a set of 32 characters belonged to different scripts, the accuracy of 70% was yielded in the detection of characters. They concluded that in addition to the methods used as preprocessing, the spurring techniques lead to better results.

III. OPTICAL CHARACTER RECOGNITION

The Optical character recognition has emerged from the field of pattern recognition in general. It is a complex technology that reads text within images and converts it into a specific format with an editable text. And in more detail, an OCR system starts from the digital image made from an optical scan of a page (printed document, typed sheet, etc.), or taken from a digital camera or even screenshot. The text that the OCR system aims to recognize may be originally handwritten on a paper, on a device (tablet, digitizer, etc.), or written with a machine (computer, typewriter, etc.) and then printed. Finally, the OCR outputs an editable text file in various formats (DOC, TXT, XML, etc.).
Some OCR systems try to preserve the style of the text (its type of font, its color, is it italic, bold, underlined, etc.) as well as the layout, or even rebuild the tables and extract the images. While other OCR systems try to recognize only what is written without paying attention to the style of the text and return the result in a unified simple form. And our work is of this type.
An OCR system can be “offline” (see Fig. 2), that is, characters are recognized after processing text documents, or can be “online”, that is, characters are recognized immediately when they are written for example on a digitizer (pen tablet) or on a personal digital assistant (PDA). In the online systems, the points of writing are stored as a function of time, and therefore, in case of overwriting the characters, it is possible to distinguish the characters from each other [2], and this is difficult in offline systems.
The first commercial OCR system came out in the early 1950s, but the idea of an OCR and the studies on it has been appeared or used much earlier before that. Since the mid-1940s, researchers have published much research on character recognition. Most of these studies concerned the recognition of Latin characters, and later, Japanese and Chinese characters since 1960. As for Arabic character recognition, maybe the first published work on this is due to 1975 [3]. Among the leading commercial systems of Arabic character recognition is Sakhr [4], and a few other programs that support Arabic, such as ReadIris, OmniPage and Verus. Some of these programs claim that their accuracy in Arabic character recognition exceeds 99% [4].

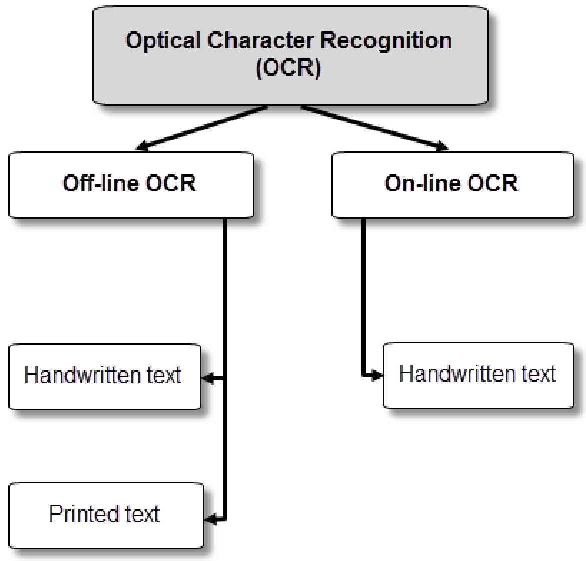


Fig. 2. The both types offline and online of an OCR system, with the writing nature that each type usually receives.

The main stages that may be within an OCR system are as follows (see Fig. 3):

A. *Preprocessing of the Image*

The goal of this stage is to improve the quality of the image. This may include the recovery of inclined or distorted images, noise removal, contrast corrections, passing to two-color mode (binary image), contour or skeleton detection, etc.

B. *Segmentation or Page Analysis*

At this stage, the detected text is segmented into lines of text, then, into words (or sub-words), and finally into characters. This phase can also detect the text style: underlined expressions, tables, images, etc.

C. *Classification and Recognition*

After the normalization process (of scale, translation, and rotation), a character to be recognized is compared to a set of known characters in a database, and the closest form (or the ***N*** closest forms) to the character is retained for the next step. Recognition techniques can be classified into two main types: the first is a naive classification, which consists of directly comparing the character to be recognized with the other stored in the database, pixel by pixel. It is also called "pattern matching". It is a technique used in early character recognition systems. The second is the classification by Features: which consists of giving to a character a unique description, on which the recognition is based. This description is an abstract mathematical representation of the character, and more precisely, it is a finite set of ***n*** measures that characterize the form of the character (such as the number of closed loops, lines, intersections of lines, directions of lines, etc.) and which are considered a point $X = (x_1,x_2,...,x_n)$ in an ***n***-dimensional space. This point can be considered as a vector called "feature vector", so that we can, for classification, use the properties of Euclidean space, such as the measure of Euclidean distance between two vectors. And many OCR systems use various types of artificial neural networks (ANN) trained on a wide database of possible forms of characters. And in the field of handwriting recognition, probabilistic and statistical methods such as Markov chains are frequently used.

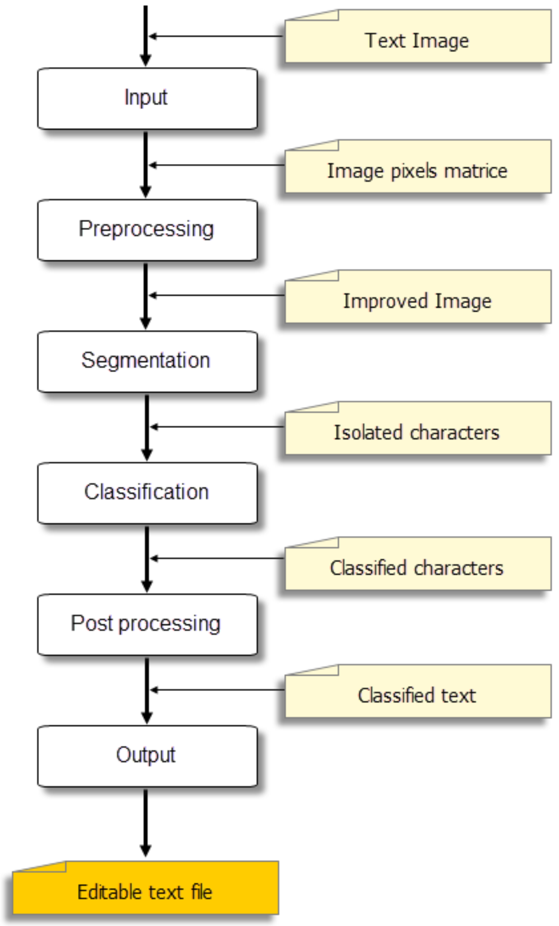


Fig. 3. The main stages that may be within an OCR system.

D. *Post Processing*

The goal of this stage is to reduce the number of errors when the recognition process generates a list of possible characters or words, by making orthographic corrections automatically by using a dictionary and successive constraints: lexical, syntactic, semantic, or other constraints.

E. *Generation of the Output Format*

At this stage, the OCR system generates a text file of the desired format, containing the recognized and editable text, which may or may not retain the «layout» of the original page.

IV. PREPROCESSING

Image preprocessing is an important stage for Arabic character recognition. It consists of highlighting the useful information appear in the input image and reducing or eliminating unnecessary information.

A. *Separation of Dots*

After observing the various situations where the character is accompanied by two or three dots, such as the characters «ث» and «ي», A. Mazroui et al. [13] found that in many fonts, two or three dots have often some shared pixels (see Fig. 4).

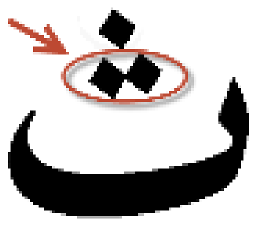


Fig. 4. Two dots often have one or two shared pixels.

To remove these shared pixels, we used the following masks:

$$\begin{array}{|c|c|c|}\hline 1&0&1\\\hline 1&1&1\\\hline 1&0&1\\\hline\end{array}\quad\begin{array}{|c|c|c|}\hline 0&0&1\\\hline 1&1&1\\\hline 1&0&1\\\hline\end{array}\quad\begin{array}{|c|c|c|}\hline 1&0&1\\\hline 1&1&1\\\hline 0&0&1\\\hline\end{array}\quad\begin{array}{|c|c|c|}\hline 1&0&0\\\hline 1&1&1\\\hline 1&0&1\\\hline\end{array}\quad\begin{array}{|c|c|c|}\hline 1&0&1\\\hline 1&1&1\\\hline 1&0&0\\\hline\end{array}$$

$$\begin{array}{|c|c|c|c|}\hline 1&0&0&1\\\hline 1&1&1&1\\\hline 1&0&0&1\\\hline\end{array}\quad\begin{array}{|c|c|c|c|}\hline 0&0&0&1\\\hline 1&1&1&1\\\hline 1&0&0&1\\\hline\end{array}\quad\begin{array}{|c|c|c|c|}\hline 1&0&0&1\\\hline 1&1&1&1\\\hline 0&0&0&1\\\hline\end{array}\quad\begin{array}{|c|c|c|c|}\hline 1&0&0&1\\\hline 1&1&1&1\\\hline 1&0&0&0\\\hline\end{array}$$

$$\begin{array}{|c|c|c|c|}\hline 1&0&0&0\\\hline 1&1&1&1\\\hline 1&0&0&1\\\hline\end{array}$$

B. Classes of Characters

According to [13], we can classify the 29 basic forms of Arabic characters into two classes:

- The class of characters having a loop:

LC = {ف ق ة ظ ط ص ض و م ه}

- The class of characters with no loops:

NLC = {ش س ك غ ع خ ج ح ز ر ذ د ل ي ن ث ت ب ا}

- Taking into account the number of dots in the character, the first class ***LC*** will be divided into three classes:

LC0 = {ه م و ص ط}
LC1 = {ض ظ ف}
LC2 = {ة ق}

- And the second class ***NLC*** will be divided into four classes:

NLC0 = {س ك ع ح ر د ل ا}
NLC1 = {ذ غ خ ج ز ن ب}
NLC2 = {ت ي}
NLC3 = {ش ث}

A given character belongs to the ***LC*** class, only if, after the skeletonization process, we encounter a pixel twice when we traverse the character curve in a given direction [13].

In this work, we have added other classes, according to the following remarks:

- There is only one Arabic character that can contain two loops, it is the «ه» character in its two forms: «هـ» at the beginning of the word, and «ـهـ» in the middle of the word. In some fonts, this character is written with two loops even if it is isolated, as for example in the «Arabic typesetting» font.
- In all Arabic characters, if there is more than one dot, they are all under the character's body, or, they are all above.
- In all Arabic characters that have three dots, the three dots are always located above the body of the character. Therefore, for characters containing three dots, it is not necessary to create subclasses of the ***NLC3*** class to classify the characters depending on the location of dots (under or above the character's body).
- In all Arabic characters containing loops and dots, the dots are always located above the character regardless of their number. Therefore, for characters containing loops and dots, it is not necessary to create subclasses of the ***LC*** class to classify the characters depending on the location of dots.
- The complementary parts, of an Arabic character, are not always dots, as is the case of the Arabic character "ك" whose complementary part is a symbol like "ء" (called Hamza). Therefore, from the class of characters containing loops, and dots above the character body, we will classify these characters according to the nature of its complementary parts (Dots, Hamza, Maddah, etc.).

We will add the form «هـ» to the previous basic forms of Arabic characters, and create these classes:

- A class for the dual-loops characters:

DLC = {هـ}

- A class of characters that have no loops and have a single dot under their bodies:

NLC1-0 = {ب ج}

- A class of characters that have no loops and have a single dot above their bodies:

NLC1-1 = {ذ غ خ ز ن}

- A class of characters that have no loops and have two dots under their bodies:

NLC2-0 = {ي}

- A class of characters that have no loops and have two dots above their bodies:

NLC2-1 = {ت}

- A class of characters that have no loops and have complementary parts that are not dots:

NLC0-1 = {ك}

A dot is considered under the body of a character if, at least in one column among the columns of the matrix that represents the image of the character, a pixel from the dot exists under a pixel from the body of the character. Otherwise, if the dot exists, it is considered above the character's body by default.

To differentiate the dot from another symbol, such as the Hamza symbol «ء», we can use the formula (1).

$$Circularity = \frac{4\pi A}{p^2} \qquad (1)$$

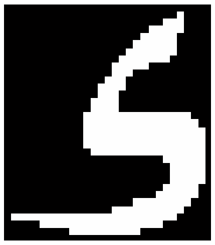
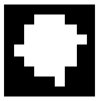
The object	Circularity
	0.14
	0.82

Fig. 5. The circularity of the symbol «ء» is close to ***0***, while the circularity of the dot is close to ***1***.

Where ***A*** is the area of the object in question and ***P*** is their perimeter. The returned result is between ***0*** and ***1***, if the object is almost a disk, the result tends to ***1***, and otherwise the result tends to ***0***. Generally, in our work, the object is considered a dot if its circularity is greater than or equal to ***0.5***, otherwise it is not a dot (see Fig. 5).

V. THE HOUGH TRANSFORM

A. Principle

Our main reference about Hough Transform principle is [14]. Originally, the Hough Transform (HT) was considered a method for detecting lines in an image, so that each line ***(D)*** is defined by an angle $\boldsymbol{\theta}$ and a distance ***r*** as shown in (Fig. 6).

A point ***M(x,y)*** belongs to ***D(r, θ)*** if ***x*** and ***y*** satisfy this relation:

$$x \cos(\theta) + y \sin(\theta) = r \qquad (2)$$

Special cases are shown in (Fig. 7).

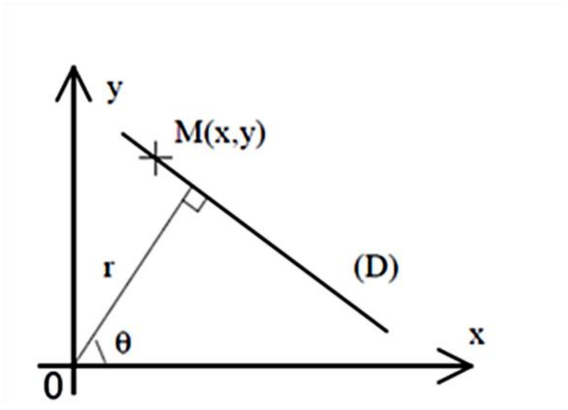


Fig. 6. Each line ***(D)*** in the plane is defined by an angle $\boldsymbol{\theta}$ and a distance ***r***.

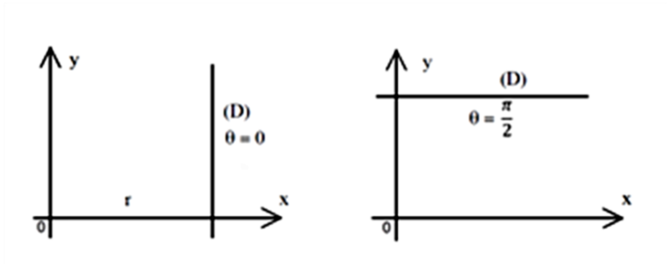


Fig. 7. Special cases.

Then, it is possible to make two transformations:

- The «Many to One» transformation, or in short «m to 1», which associates to each line of the image in the Cartesian space one and only one point in a space called «Hough Space», as shown in (Fig. 8).
- The «One to Many» transformation, or in short «1 to m», which associates to each point of the image in the Cartesian space the set of all lines passing through this point in the «Hough Space». This set of lines is represented in the Hough space by a sinusoid as shown in (Fig. 9).

In our application, we have chosen the «One to Many» transformation. The cost of performing this transformation is better than that of the transformation «Many to One» [14].

The principle of lines detection according to transformation «1 to m» is the following:

Since a curve in Hough space represents all lines passing through a certain point in the initial image, and since a line is defined only by two points, the intersection of two curves in Hough space represents a certain line in the initial image. For example, in (Fig. 10), the point ***I*** represents the line ***(D)*** passing through the two points ***M₁*** and ***M₂***.

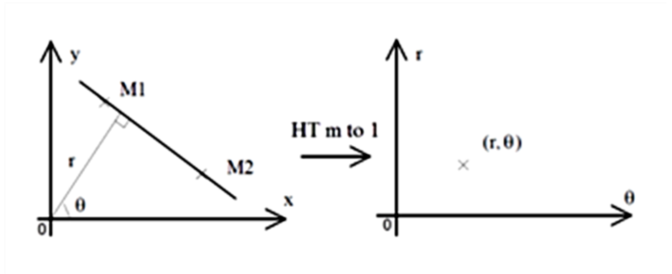


Fig. 8. The "Many to One" transformation.

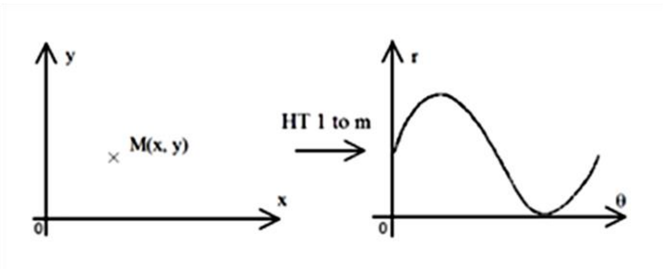


Fig. 9. The «One to Many" transformation.

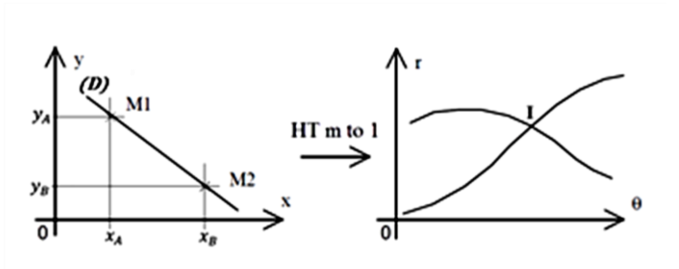


Fig. 10. The intersection of two curves in Hough space represents a certain line in the initial image.

B. *Application of HT to Arabic Characters Recognition*

The HT is suitable for Arabic characters that often vary significantly from one font to another. For example, there is a big difference between the structure of the «ث» character in the «Traditional Arabic» font and the structure of the same character in the «Andalus» font. However, the HT estimates that the two structures have the same horizontal linear extension, which is what encouraged us to rely on this method (see Fig. 11).

C. *Put into Practice*

1) *Preliminary phase*

During the programming, the origin of the plane of the image is the usual origin (upper left corner), so the two coordinates ***x*** and ***y*** of each point ***M*** of this image are integers.
A points in the Hough space will have for coordinates (r,θ), where θ can vary between $-\frac{\pi}{2}$ and $\frac{\pi}{2}$, and ***r*** can vary between: **– *r*<i></i>**$_{max}$ and **+*r***$_{max}$ with :

$$r_{max} = \sqrt{L^2 + H^2} \quad (3)$$

Where ***L*** is the width of the image, and ***H*** is its height, so r_{max} is the diagonal of the image.
For the angles ***θ***, we take, in the beginning, ***180*** directions (almost) of the interval $[-\frac{\pi}{2};\frac{\pi}{2}]$, afterward, we reduce this number of directions to minimize the cost of execution.
For the distances ***r***, we take an amplitude equal to twice the diagonal of the image, i.e. $\boldsymbol{2\times r_{max}}$ because ***r*** can take positive and negative values.
We put ***n*** the value that accumulates each time when we meet another point that belongs to the same line of angle ***θ*** and distance ***r***.
During the programming, we created an «accumulator matrix» of dimension ***M* × *N***. The first dimension is for ***r***, the second is for *θ*, and each element of this matrix represents the value *n* associated with the pair ***(r,θ)***.

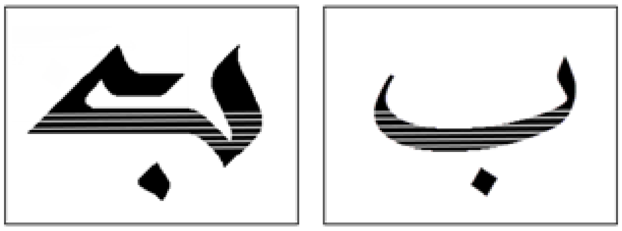


Fig. 11. With the HT, two different structures of the same character will have the same horizontal linear extension (the lines in white).

2) *Measures taken to generate the feature vector*

Since the image of an Arabic character can be considered as a line-drawing image, we have adopted the same parameters and measurements chosen by P. Fränti et al. in [15] to generate the feature vector, and which are the formulas from (4) to (8). At first, the accumulator matrix is thresholded in order to preserve the maximum values, then a feature vector is constructed by summing up the significant coefficients in the columns of each θ-value. In this way, only the angular information is used, therefore, the method by its nature is invariant under translation, and scaling [15].
Let A_{ij} be the accumulator matrix.

a) *To take only the maximum values of the accumulator matrix: We choose a threshold S, with:*

$$0 \le S \le \max_{i,j} A_{ij} \quad (4)$$

b) *Then we get another matrix:*

$$A'_{ij} = \begin{cases} A_{ij}\,, & if\ A_{ij} > S \\ 0\,, & if A_{ij} \le S \end{cases} \quad \forall i = 1..M, \quad j = 1..N \quad (5)$$

c) *Calculation of the preliminary feature vector:*

$$F_j^0 = \sum_{i=1}^{M} A'_{ij} \; \forall j = 1..N \quad (6)$$

d) *Calculation of the average:*

$$m = \frac{1}{N}\sum_{j=1}^{N} F_j^0 \quad (7)$$

e) *Normalization of the feature vector:*

$$F_j = \frac{F_j^0}{m} \forall j = 1..N \quad (8)$$

3) *Measures taken to compare two feature vectors*

Let $X = (x_{1,}x_{2,}, \dots, x_{p,})$ and $Y = (y_{1,}y_{2,}, \dots, y_{p,})$ be two feature vectors. To compare these two vectors, we have calculated the Euclidean distance of them, it is defined by:

$$d(X,Y) = \sqrt{\sum_{i=1}^{p}(x_i - y_i)^2} \quad (9)$$

An important advantage of the formula is its speed, and this is a desired property to search in a large database.
To obtain the percentage of similarity, we used the formula:

$$s_p(X,Y) = \frac{100}{1+d(X,Y)} \quad (10)$$

VI. IMPROVEMENTS

After reviewing different implementations of methods based on the Hough Transform and which are close to our work, the focus is always on the maximum values in the accumulator matrix, and only one threshold is taken into account. In this work, we have added another threshold to take also the minimum values in the accumulator matrix. This makes it possible to characterize an Arabic character more accurately, and not to neglect necessary components for Arabic characters such as the dots that are used to differentiate similar characters such as «ج» and «خ».
The addition of another threshold increased the size of the feature vector, and hence the cost of execution, but we improved the performance of the system by reducing the number of directions associated with angles $\boldsymbol{\theta}$, to the maximum, without affecting the results obtained in this work.

VII. TESTS AND RESULTS

A. Noise, Scaling, Translation and Rotation Tests

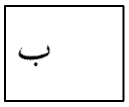
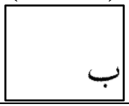
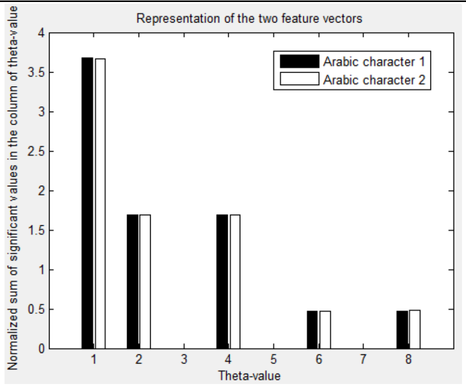
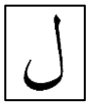
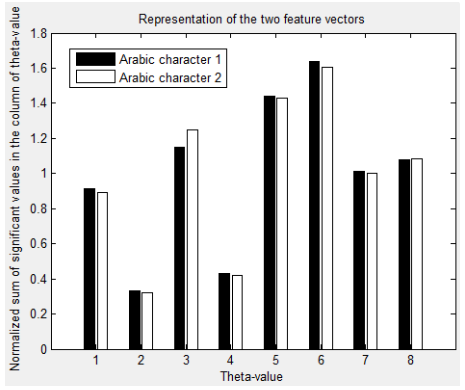
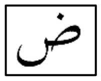
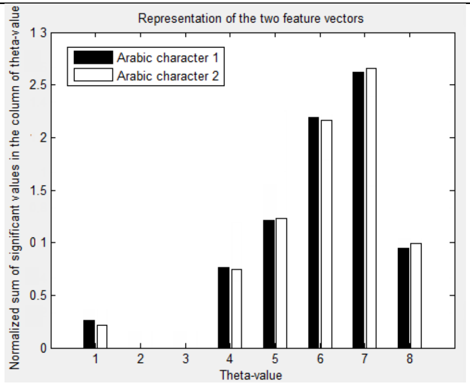
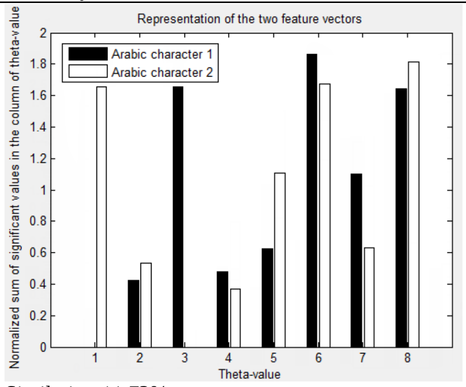
We have tested the sensitivity of the method we propose to noise, scaling, translation, and rotation.
Table II shows the results of some of these tests in the form of histograms. The black columns of each histogram represent the feature vector elements of the original Arabic character, while the white columns represent the feature vector elements of the same character after noise exposure, or translation or scaling, or rotation.
By the results of the tests, we made sure that the method is invariant by translation, is not too affected by the noise, and the change of scale, but is very sensitive to the rotation. According to [15] we can overcome the problem of sensitivity to the rotation, by permuting the elements of the feature vector of the query image, and calculating its distance from the feature vector of the target image, until we get the minimum distance. But this will be done at the cost of an increase in computing time. So we propose to solve this problem at a preliminary stage since we will be dealing with the character in the context of a word or text that can be preprocessed by applying the HT to determine the baseline of the Arabic writing, and then we return it to the horizontal direction if it was tilted [16].

B. Learning and Recognition Tests

For learning and recognition tests, we used a database containing 300 printed and isolated Arabic characters, and distributed over 10 fonts among those supported by some of the most widely used operating systems:

- Tradition Arabic
- Times New Roman
- Arial
- Simplified Arabic
- Simplified Arabic Fixed
- Arabic typesettig
- Dycotype Naskh
- Akhbar MT
- Microsoft Sans Serif
- Andalus

TABLE II
NOISE, SCALING, TRANSLATION AND ROTATION TESTS

Compared characters	Histogram
Arabic character 1  Arabic character 2 (translation) 	 Similarity: 100%
Arabic character 1  Arabic character 2 (noise)	 Similarity: 98,88%
Arabic character 1  Arabic character 2 (scaling)	 Similarity: 99,29%
Arabic character 1 Arabic character 2 (rotation)	 Similarity: 11,73%

Each of these ***10*** fonts will be noted, in this order, by $\boldsymbol{F_i}$, with $\boldsymbol{i}$ an integer, ***1≤i≤10***. We performed 1023 tests as follows: In each test, we take a new combination of fonts among the ten fonts for the learning phase, and then calculate the recognition rate (RR) for each font of the whole set. The total number of combinations - and therefore the number of tests - was$\sum_{k=1}^{10} C_{10}^{k}$. Then, for each value of $\boldsymbol{k}$, where $\boldsymbol{k}$ is an integer, ***1≤k≤10***, we choose the ***k***-combinations that gave the best result. Finally, we obtained the results presented in Table III.

TABLE III
THE RESULTS OBTAINED USING THE PROPOSED METHOD.

k	Combinations	RR (%)
1	*{F1}*	91.3
2	*{F2,F6}, {F3,F6}*	95.0
3	*{F2,F6,F10}, {F3,F6,F10}, {F4,F6,F10}*	97.7
4	*{F2,F6,F9,F10}, {F3,F6,F9,F10}, {F4,F6,F9,F10}, {F5,F6,F9,F10}*	99.0
5	*{F1,F2,F8,F9,F10}, {F1,F3,F8,F9,F10}, {F1,F4,F8,F9,F10}, {F1,F5,F8,F9,F10}, {F2,F3,F6,F9,F10}, {F2,F4,F6,F9,F10}, {F2,F5,F6,F9,F10}, {F3,F4,F6,F9,F10}, {F3,F5,F6,F9,F10}, {F4,F5,F6,F9,F10}*	99.0
6	*{F1,F2,F7,F8,F9,F10}, {F1,F3,F7,F8,F9,F10}, {F1,F4,F7,F8,F9,F10}, {F1,F5,F7,F8,F9,F10}*	99.7
7	*{F1,F2,F6,F7,F8,F9,F10}, {F1,F3,F6,F7,F8,F9,F10}, {F1,F4,F6,F7,F8,F9,F10}, {F1,F5,F6,F7,F8,F9,F10}*	100
8	*{F1,F2,F3,F6,F7,F8,F9,F10}, {F1,F2,F4,F6,F7,F8,F9,F10}, {F1,F2,F5,F6,F7,F8,F9,F10}, {F1,F3,F4,F6,F7,F8,F9,F10}, {F1,F3,F5,F6,F7,F8,F9,F10}, {F1,F4,F5,F6,F7,F8,F9,F10}*	100
9	*{F1,F2,F3,F4,F6,F7,F8,F9,F10}, {F1,F2,F3,F5,F6,F7,F8,F9,F10}, {F1,F2,F4,F5,F6,F7,F8,F9,F10}, {F1,F3,F4,F5,F6,F7,F8,F9,F10}*	100
10	*{F1,F2,F3,F4,F5,F6,F7,F8,F9,F10}*	100

The table shows that, in many cases, the recognition rate is high, despite the small number of fonts used during the learning phase. For example, when we only use *F1* («Traditional Arabic») for the learning phase, the rate is 91,3% and Table IV shows the results in detail. The lowest recognition rate is that of the font «Andalus», which can be attributed to the fact that the font «Andalus» is more different and complex than other fonts.

TABLE IV
USING ONE FONT FOR LEARNING

Learning fonts	Test Fonts	RR (%)	Average
Traditional Arabic	Traditional Arabic	100	91.3%
	Times New Roman	100	
	Arial	100	
	Simplified Arabic	100	
	Simplified Arabic Fixed	96,7	
	Arabic typesettig	90,0	
	Dycotype Naskh	80,0	
	Akhbar MT	96,7	
	Microsoft Sans Serif	86,7	
	Andalus	63,3	

VIII. CONCLUSION

The HT-based approach has worked well when it comes to optical recognition of Arabic characters written with fonts that are already learned or not at the learning phase, probably because we are generating a feature vector that contains only information on the directions of the lines that form the general structure of the character, which also makes this method invariant under translation and scaling, and not too affected by the noise.

Compared to works based on the Hough Transform [9] [10] [11] [12] or other works based on other methods [8] [13], although they were not based on a standard Arabic database, which makes difficult to give an objective comparison, however, the remarkable results obtained in this work confirm that the Hough Transform, by the way that we described, can be used to recognize Arabic characters and that it has some flexibility when these characters are written in different fonts.

Further research is still needed to adapt and effectively apply the proposed approach to a large set of more complex Arabic fonts. This is a starting point for our future research.

REFERENCES

[1] D. Fassi Fihri, "Digitalization of Arabic Ancient Manuscripts: Challenges and opportunities," Proceedings of the 5th IEEE Congress on Information Science and Technology, Marrakech, Morocco, October 21-27, 2018.
[2] K. Gaurav and P. K. Bhatia, "Analytical Review of Preprocessing Techniques for Offline Handwritten Character Recognition", International Journal of Advances in Engineering Sciences, Vol.3 No 3, 2013, pp. 14-22.
[3] B. Al-Badr and S.A. Mahmoud, "Survey and Bibliography of Arabic Optical Text Recognition", Elsevier Science, Signal Processing, Vol. 41, 1995, pp. 49-77.
[4] A. Beg, F. Ahmed and P. Campbell, "Hybrid OCR Techniques for Cursive Script Languages - A Review and Applications," 2010 2nd International Conference on Computational Intelligence, Communication Systems and Networks, Liverpool, 2010, pp. 101-105.
[5] A. Lawgali, "A Survey on Arabic Character Recognition," International Journal of Signal Processing, Image Processing and Pattern Recognition, Vol. 8, No. 2, 2015, pp. 401-426.
[6] L. S. Alhomed, K. M. Jambi, "A Survey on the Existing Arabic Optical Character Recognition and Future Trends," International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE), Vol. 7, No. 3, 2018, pp. 78-88.
[7] F. M. A Nashwan, M. A. A. Rashwan, H. M. Al-Barhamtoshy, S. M. Abdou, A. M. Moussa, "A Holistic Technique for an Arabic OCR System," Journal of Imaging, Vol. 4, No. 1, p. 6, 2018, pp. 1-11.
[8] H. Althobaiti, C. Lu, "A survey on Arabic Optical Character Recognition and an isolated handwritten Arabic Character Recognition algorithm using encoded freeman chain code," The 51st Annual Conference on Information Sciences and Systems (CISS), Baltimore, MD, 2017, pp. 1-6.
[9] M. fakir, M. M. Hassani, C. Sodeyama, "On the recognition of Arabic characters using Hough transform technique," Malaysian Journal of Computer Science, Vol. 13, No. 2, 2001, pp. 39-47.
[10] S. Touj, N. E. B. Amara, H. Amiri, "Generalized Hough Transform for Arabic Printed Optical Character Recognition," International Arab Journal of Information Technology, Vol. 2, No. 4, 2005, pp. 326-333.
[11] N. B. Amor, N. E. B. Amara, "Multifont Arabic Characters Recognition Using HoughTransform and HMM/ANN Classification," Journal of Multimedia, Vol. 1, No. 2, 2006, pp. 50-54.
[12] N. A. Khuzani , M. Mahmoodi, "Recognition Persian Handwritten Characters using Hough Transform," International Journal of Engineering Trends and Technology (IJETT), Vol. 16, No. 2, 2014, pp. 65-68.

[13] A. Mazroui, A. KerkourElmiad, "Recognition of Multifont Isolated Arabic Characters by Bézier Curves," International Journal of Computer Applications, Vol. 100, No. 6, 2014, pp. 1-7.
[14] D. Patrice, "Contribution à la Segmentation et à la Reconnaissance de l'Ecriture Manuscrite," thesis, The National Institute of Applied Sciences of Lyon, 1994. http://patrice.dargenton.free.fr/publications/these.html
[15] P. Fränti, A. Mednonogov, V. Kyrki, H. Kälviäinen, "Content-based matching of line-drawing images using the Hough transform," International Journal on Document Analysis and Recognition (IJDAR), Vol. 3, 2000, pp. 117-124.
[16] F. Stahlberg and S. Vogel, "Detecting dense foreground stripes in Arabic handwriting for accurate baseline positioning," 2015 13th International Conference on Document Analysis and Recognition (ICDAR), Tunis, 2015, pp. 361-365.

M. Kadi Professor of Mathematics, and a PhD student, works on image processing and pattern recognition in MATSI laboratory of the school of Technology, University of Oujda, Morocco.

M. Nasri Electromechanical Engineer of ENIM Rabat is Professor at the EST of Mohammed Premier University of Oujda (Morocco). He works on quality control by artificial vision and on the monitoring techniques of industrial equipment in operation. He is a permanent member of the MATSI laboratory of the school of Technology, University of Oujda, Morocco.