Arabic character recognition using a Haar-Cascade Classifier approach (HCC)

Article *in* Formal Pattern Analysis & Applications · April 2015

DOI: 10.1007/s10044-015-0466-2

Arabic character recognition using a Haar-Cascade Classifier approach (HCC)

Ashraf AbdelRaouf Colin A. Higgins Tony Pridmore Mahmoud I. Khalil

Abstract Optical Character Recognition (OCR) shows great potential for rapid data entry, but has limited success when applied to the Arabic language. Traditional OCR problems are compounded by the nature of Arabic language and because the script is heavily connected. A machine learning, Haar-Cascade classifier (HCC) approach was introduced by Viola and Jones [1] to achieve rapid object detection based on a boosted cascade of simple Haar-like features. Here, that approach is applied for the first time to suit Arabic glyph recognition. HCC approach eliminates problematic steps in the pre-processing and recognition phases and, most importantly, character segmentation stage. A classifier was produced for each of the 61 Arabic glyphs that exist after the removal of diacritical marks (dots). These classifiers were trained and tested on some 2,000 images each. The system was tested with real text images and produces a recognition rate for Arabic glyphs of 87%. The technique gives good results relative to those achieved using a commercial Arabic OCR application and existing state of the art research application.

A. AbdelRaouf
Faculty of Computer Science, Misr International University, Cairo, Egypt
e-mail: ashraf.raouf@miuegypt.edu.eg

C. Higgins (✉). T. Pridmore
School of Computer Science, The University of Nottingham, Nottingham, UK
e-mail: colin.higgins@nottingham.ac.uk

e-mail: tony.pridmore@nottingham.ac.uk

M. Khalil
Computer and Systems Engineering Department
Faculty of Engineering, Ain Shams University, Cairo, Egypt
e-mail: mahmoud.khalil@eng.asu.edu.eg

Keywords: Arabic character recognition, Haar-like features, Cascade classifiers, Haar cascade classifiers

1. Originality and contribution

The main contribution of this research is to identify an Arabic OCR application without character segmentation and with reduced pre-processing and recognition steps.

Other studies have attempted to use character segmentation-free recognition, but these attempts failed either to present a concrete application or to achieve reasonable results [2]. A successful, visual object detection approach has been adopted here to achieve true character segmentation-free recognition depending on Haar-like features. This approach successfully achieved a glyph recognition rate of 87%.

The Haar-Cascade Classifier (HCC) approach is applied to character recognition, and specifically Arabic character recognition, for the first time. Though the approach was originally developed for face detection, it has also been used to detect eyes, pedestrians and even bowls [3].

2. Introduction

The HCC approach was initially presented in 2001. Viola and Jones [1] introduced a rapid object detection algorithm using a boosted cascade of simple features which they applied to face detection. The integral image is introduced as a new image representation which allows the features to be computed very quickly. Lienhart et al. [4] extended the Haar-like features by adding rotated

features, and Lienhart *et al.* [5] presented an empirical analysis of different boosting algorithms with improved detection performance and computational complexity.

The present paper describes experimental work on the novel application of the HCC approach to the recognition of Arabic glyphs. Related work with the features of Arabic language and the HCC approach are presented in section 2. Section 3 shows a review of existing Arabic OCR approaches. Then, an experiment is introduced in which the entire set of Arabic dot-less (naked) glyphs (Arabic glyphs that exist after the removal of diacritical marks) are recognised. Results are presented and the paper concludes with a discussion of the value of the HCC approach to Arabic character recognition.

2.1 Related work

An Arabic OCR application was presented by Kanoun *el al.* [6] for the identification of Arabic and Latin texts. The approach comprised of two different analyses. The first is morphological analysis and is performed on the text block level. The second is geometrical and concerns the text line level and the connected components level. When tested with a KNN classifier and without any optimization, it achieved an identification rate of 88.5%. The system was developed by performing an Arabic text analysis method using the affixal approach [7]. The system developed through [8-12] using different approaches to enhance its recognition rate by adding morphological enhancement. An identification method for ultra-low resolution Arabic word images using a stochastic approach was presented in [13]. It achieved 23% better word recognition using the APTI database [10]. This state of the art application recognition results was used as a comparison with the proposed approach.

2.2 Key Features of Written Arabic

The Arabic language consists of 28 letters [14, 15] and is written from right to left. Arabic script is cursive even when printed and Arabic letters are connected by the baseline of the word. The Arabic language makes no distinction between capital and lower-case letters. The widths of letters in Arabic are variable [16].

Arabic alphabets depend on dots to differentiate between letters. There are 19 "joining groups" [17]. Each joining group contains more than one similar letter which differs in the number and place of the dots, as for example (ج ح خ) which have the same joining group (ح) but with differences in the placement of dots. Table 1 shows the joining groups with the letters shown in their location in the word. A character in any of its different locations is called a glyph [18].

Arabic letters have four different shapes according to their location in the word [19]; Start, Middle, End and Isolated. For the six letters (ا د ذ ر ز و) there is no Start or Middle location shape (table 1). An Arabic word may consist of one or more sub-words. These disconnected sub-words are termed PAWs (Pieces of Arabic Word) [19-21]. For example (رسول) is an Arabic word with three different PAWs (ل) (سو) (ر). Removing the dots to leave dot-less words and dot-less PAWs reduces the number of letters that needs to be considered from 28 to 19 [22].

Arabic font files exist which define font shapes with historical form which are totally different from modern fonts. The Arabic language incorporates ligatures such as (Lam Alef لا) which actually consist of two letters (ل ا) but when connected produce another glyph [23]. The Arabic language contains a number of similar letters like Alef (ا) and the number 1 (١), and also the full stop (.) and the Arabic number 0 (٠) [16, 24].

Arabic OCR is more complex than that of European languages and suffers from the complexity of the language itself. The connectivity of the letters and the four different shapes of each letter represent the most complex issues (table 1).

Table 1: Arabic joining groups and group letters, defined with letters location

Characters	Isolated	Start	Middle	End
ا	ا			ـا
ب - ت – ث	ب ت ث	بـ تـ ثـ	ـبـ ـتـ ـثـ	ـب ـت ـث
ج - ح – خ	ج ح خ	جـ حـ خـ	ـجـ ـحـ ـخـ	ـج ـح ـخ
د – ذ	د ذ			ـد ـذ
ر – ز	ر ز			ـر ـز
س – ش	س ش	سـ شـ	ـسـ ـشـ	ـس ـش
ص – ض	ص ض	صـ ضـ	ـصـ ـضـ	ـص ـض
ط – ظ	ط ظ	طـ ظـ	ـطـ ـظـ	ـط ـظ
ع – غ	ع غ	عـ غـ	ـعـ ـغـ	ـع ـغ
ف	ف	فـ قـ	ـفـ	ـف
ق	ق			ـق
ك	ك	كـ	ـكـ	ـك
ل	ل	لـ	ـلـ	ـل
م	م	مـ	ـمـ	ـم
ن	ن			ـن
ه	ه ة	هـ	ـهـ	ـه ـة
و – ؤ	و			ـو ـؤ
ى - ي – ئ	ى			ـى ـي ـئ
ء	ء			
لا	لا			ـلا

2.3 Haar Cascade Classifier (HCC) Approach

The HCC is a machine learning approach that combines three basic components [1, 25]. These are the Integral image, Haar-like feature extraction and boosting of a classifier cascade.

Integral image: was first used in feature extraction by Viola and Jones [1]. The integral image at any location (x, y) is equal to the sum of the pixels' intensities (grey-scale) from $(0, 0)$ to (x, y) and is known as the Summed Area Table $SAT(x, y)$. SAT is an algorithm for calculating a single table in which pixel intensity is replaced by a value representing the sum of the intensities of all pixels contained in the rectangle [26]. Lienhart and Maydt [4] developed an algorithm to add the rotated

integral image. The integral image for the 45° rotated rectangle at any location (x, y) is equal to the sum of the pixels' intensities at a 45° rotated rectangle with the bottom most corner at (x, y) and extending upward till the boundary of the image and is known as $RSAT(x, y)$.

Haar-like feature extraction: captures basic visual features of objects. It uses grey-scale differences between rectangles to extract object features [1]. Haar-like features are calculated by subtracting the sum of a sub-window of the feature from the sum of the remaining window of the feature [27]. Lienhart and Maydt [4] added rotated features, significantly enhancing the learning system and improving the classifier performance. These rotated features were significant when applied to objects with diagonal shapes, and are particularly well suited to Arabic character recognition [28].

Boosting of a classifier cascade: is a decision tree which depends upon the rejection of non-object regions. Boosting algorithms use a large set of weak classifiers to generate a powerful classifier. The weak classifiers rely on a single feature only. Weak classifiers discriminate required objects from non-objects in a simple and efficient way. Weak classifiers use only one classifier at each stage and depend on a binary threshold decision or small Classification And Regression Tree (CART) for up to four features at a time [29].

3. Structure of the Arabic OCR Application

An OCR application normally consists of three main sections: pre-processing, recognition and post-processing [15, 19, 23, 30] as shown in figure 1. The pre-processing section is responsible for preparing the document image for recognition. The more successful the pre-processing algorithms are, the better the recognition rate of the application. The recognition section handles the document image after, for example, line finding. In turn, the recognition section outputs a machine-readable recognized form of a document image. Finally, the post-processing section receives the series of recognized letters/words and indicates which possible words are most likely in this situation, based on linguistic or statistical data.

3.1 Pre-processing

The pre-processing section cleans the image using image processing techniques, and often converts the image to a more concise representation. The pre-processing section normally includes the following steps [19, 31]:

1. Page layout analysis and decomposition is responsible for analysing the contents of the document image. It defines the blocks of data inside the image, whether text or non-text [32].
2. Binarization is responsible for receiving the scanned image and converting the document from a grey-scale image to a binary image. Noise removal eliminates small erroneous regions and pixels from the binarized image [33].
3. A thinning step is normally used with handwritten input and is sometimes used with printed input. This step removes the stroke width and leaves a skeleton stroke with a width of one pixel [24].
4. Slant and skew pre-processing detect the slant and skew in the document image and correct them. Ahmad, I. [34] proposes a new multi-step skew detection technique for printed Arabic documents.
5. Baseline finding identifies the baselines of the text lines in a paragraph of text. The baseline is the horizontal line that passes through the connected primitives of the line of text [30] or just under the non-descending letters for non-connected text [35].

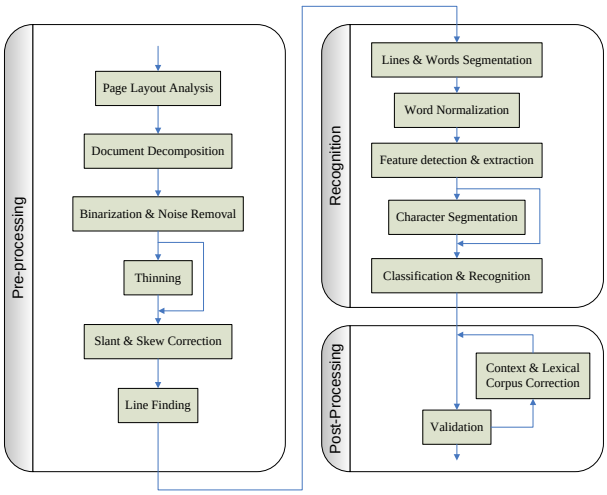


Fig. 1: A general framework for character recognition

3.2 Recognition

The recognition part normally includes three main phases in OCR research and applications. These phases are feature extraction, character segmentation and classification as shown in figure 1.

3.2.1 Feature extraction phase

The purpose of the feature extraction phase is to calculate the attributes of patterns that are most suitable for the classification phase. Feature extraction algorithms can be categorized into geometric, structural and space transformation features. Moments are an example of geometric features. Broumandnia *et al.* [36] presents an approach for Farsi character recognition based on fast Zernike wavelet moments and artificial neural networks.

Touj *et al.* [37] presented a generalized Hough transform technique which is known for its ability to

absorb the distortions in the document image and noise. The approach showed high efficiency in recognizing Arabic printed characters in their different shapes. Noor *et al.* [38] presented a simple and accurate method to recognize Arabic (Indian) numerals using Fourier descriptors. The test results indicate a recognition ratio of 98% when 4 Fourier descriptors are used.

Linear transforms include Principal Component Analysis (PCA) and Linear Discriminate Analysis (LDA) [39] are used a pre-processing step in Arabic OCR systems such as Zidouri [40] and Kurt et al. [41].

Haar-like feature is a simple method depending on the basic visual features of the objects. It uses grey-scale differences between rectangles to extract the object features [1]. The proposed approach used this method for feature extraction as shown in section 2.3.

3.2.2 Character segmentation phase

Character segmentation refers to the process of splitting a word into its characters, strokes or primitives [19]. Some Arabic OCR systems used isolated and pre-segmented characters which do not address the segmentation problem [2, 30]. Touj *et al.* [37] used Hough Transform to recognize isolated Arabic characters. Most Arabic OCR systems segment a word into primitives. Each symbol may be a character, a stroke, ligature or sub-word. Trenkle *et al.* [42] proposed a system that recognizes Arabic and Farsi text. The text segmentation module generates atomic image segments with a size smaller than a single character.

Character segmentation and recognition can be integrated. Yalniz *et al.* [43] presented a segmentation and recognition method for connected letters in Ottoman script. The method first extracts a set of segments from a connected script and determines the candidate letters. The segmentation stage can be avoided by recognizing the whole word or sub-word without trying to segment and recognize characters or primitives individually [30]. Sabbour and Shafait [44] presented a segmentation free generic OCR system for Arabic and Urdu script languages called Nabocr.

3.2.3 Classification phase

Classification involves assigning each point in the feature space to a class label score. This point was originally mapped from the input pattern onto points in the feature space during the feature extraction phase [32].

Abandah *et al.* [45] used a statistical classifier OCR system for Arabic handwritten scripts. Hidden Markov Models (HMMs) are widely used to build Arabic OCR systems. Alma'adeed *et al.* [46] presented a complete scheme for unconstrained Arabic handwritten word recognition based on an HMM. Structural classifiers are another class of classifiers which are usually used in online character recognition rather than in offline character recognition. Bushofa and Spann [47] presented an approach to segment Arabic words into characters.

Different types of Artificial Neural Networks (ANNs) have been used in Arabic OCR systems, such as Harty and Ghaddar [24] who built two neural networks to classify the segmented characters of handwritten Arabic text. Mehran *et al.* [48] presented a Support Vector Machine (SVM) based system to recognize Persian/Arabic cursive documents. It controls an adaptive layout analysis system with a combined MLP-SVM recognition process.

Combined classifier methods have proven to have the advantage of grouping over individual classifiers [32]. Rahman and Fairhurst [49] presented a review of the multi classifiers used for character recognition.

Cascade of boosting classifiers is a decision tree algorithm which depends upon the rejection of non-object regions. A classifier detects the objects of interest and rejects the non-object patterns [29]. The proposed approach used this method for classification as shown in section 2.3.

3.3 Post-processing

Post processing involves error correction or resolving ambiguity in OCR results using appropriate information. The output of OCR is compared to how the system's dictionary (corpus). According to the difference between the output of OCR and the output of corpus look-up, the numbers expressing belief in the classification are modified. The output sequence of suitable candidates is then ordered and the best candidate is selected. The post-processing part is beyond the scope of this research.

4. Experiment preparation

Three preparatory steps are necessary for the experiment. This approach is first justified theoretically and practically, then an experimental plan is presented and before describing how the data was prepared for the experiment.

4.1. Approach justification

The approach is justified theoretically and practically as follows.

4.1.1 Faces and Glyphs

The HCC approach was intended for face detection, which at first glance is different from printed Arabic character recognition. There are, however, important similarities between faces and Arabic glyphs:

1. As with faces, most Arabic glyphs have clear and distinguishing visual features.
2. Arabic characters are connected, and recognition requires individual glyphs to be picked out from a larger document image; like picking out individual faces from a crowd.
3. Characters can have many font sizes inside a document image and may also be rotated. This is also true of faces, which may appear at different sizes and orientations within a single image.
4. Facial images may vary considerably, reflecting gender, age and race. The use of different fonts introduces similar variations into images of Arabic glyphs.

4.1.2 Benefits of HCC approach

The HCC approach was originally introduce by Viola and Jones [1] and is implemented in the Open Computer Vision (OpenCV) library. OpenCV is an open source library of Computer Vision programming functions. It is aimed at real time computer vision applications using C/C++ and Python.

The advantages of the HCC approach make it particularly well-suited to Arabic character recognition. The Haar-like features technique captures the visual properties of an object; Arabic glyphs have clear visual properties. Combining feature extraction with the classification stages in HCC also facilitates the process of training and testing, which is valuable when so many glyphs must be recognised.

Methods based on the HCC approach capture a model of the target object from a training set. This technique can easily be adopted in Arabic glyph recognition, as many images of Arabic documents are readily available [16].

4.1.3 Pilot experiment

A pilot experiment was performed to justify and validate the HCC approach to Arabic printed character recognition. The aim was to run a basic experiment first, simply to discover whether the approach was applicable or not. If the pilot experiment was successful, a more advanced experiment would then be undertaken.

For the sake of simplicity in the pilot, a single letter, Ain (ع), was used in its isolated location as an object. The recognition rate for Ain was 88.6%. The results of the pilot experiment demonstrated that HCC is suitable for Arabic printed character recognition.

4.1.4 How to apply the HCC approach

The different Arabic glyphs represent different objects. Each glyph can be considered a different object to be detected, giving each glyph a distinct classifier. That glyph classifier will detect its glyph and ignore all others; in doing so it becomes a glyph recogniser.

The HCC approach relies on two sets of training images. A positive set containing images which include at least one target object and a negative set containing images which do not include any target objects (see Figure 2). A further positive set is required for testing.

Applying the HCC approach to Arabic character recognition requires the generation of training and testing datasets for each glyph. Each letter can appear in four different locations in the word [16], which leads to a different glyph. A total of 100 datasets and classifiers are thus required. Dealing with dot-less (naked) glyphs reduces the number of required glyphs datasets and classifiers to 61.

4.2 Planning the experiment

Planning the experiment requires applying this approach practically to achieve high recognition accuracy. Key features of the experiment are:

1. The binarization and noise removal step is skipped. The approach uses the original grey-scale images. For this reason grey-scale images were used in the positive and negative document image datasets.
2. The approach deals with the basic and rotated features of the glyphs so there is no need for the skew detection and correction step. For this reason an application was developed to artificially create rotated images from the negative and positive images.
3. The text lines detection step is skipped. Each glyph is detected along with its location in the document image, so extracting the text lines is possible later using the detected glyphs.
4. The normalization step is not needed because the HCC approach is scale invariant. For that reason, different font sizes are used.
5. The character segmentation phase of the connected Arabic words or PAWs can be omitted when using the HCC approach. Eliminating the character segmentation stage leads to improved recognition accuracy [2, 28, 30]. For this reason the system was trained and tested using real Arabic document images.

The experiment was planned to generate all the classifiers needed for Arabic printed character recognition. The HCC approach produces a classifier for each single glyph. The following aspects were addressed in the experiment:

1. Dot-less (naked) glyphs were used in the experiment to reduce the number of classifiers. Reducing the number of classifiers minimizes the recognition duration. Also, finding the place and number of dots above or below the recognized glyph is an easy process [2].
2. The total number of dot-less (naked) Arabic letters, as indicated in Unicode are 18 [17]. AbdelRaouf *et al* [16] showed that adding Hamza (ء) and Lam Alef (ﻻ)

is essential, Table 1 shows the Arabic glyphs in their different locations that were used in the experiment.

4.3 Training and testing data preparation

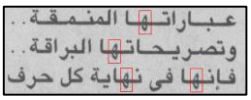
The datasets used are real scanned datasets (i.e. images of Arabic documents) that act as both negative and positive images. The paragraph image datasets of AbdelRaouf *et al* [16] were used to generate the negative and positive images for each glyph.

The MMAC corpus [16] provided data for this experiment. The part of the corpus used in this experiment was the paragraph image dataset. This part originated from 15 different Arabic document images. Five of these documents were from real scanned documents (Real data), five documents were computer generated (Computer data), and the remaining five were computer generated with artificial noise documents (Noise data). The Computer and Noise data used different Arabic fonts, sizes, bold and italic. The variety in the Computer and Noise data helped in generating robust classifiers.

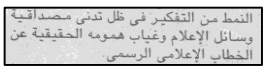
4.3.1 Creating positive and negative images

The data required for each of the 61 glyphs in this part of the experiment are:

1. *Positive images:* This set is separated into training and testing sets. Three quarters of the positive images were used during training and the rest during testing [3]. Figure 2 (a) shows a sample positive image of the Heh middle glyph (ـهـ) with four glyphs visible.
2. *Negative images:* These are used to train the classifiers. Figure 2 (b) shows a sample negative image of Heh middle glyph (ـهـ).



(a)
A positive image of Heh middle (ـهـ)



(b)
A negative image of Heh middle(ـهـ)

Fig. 2: Example positive and negative images for the Heh middle glyph (ـهـ)

A program was developed to separate the positive and negative images of each glyph. The program reads the ground truth text file of each document image and determines whether or not the glyph exists in this document. If the glyph is missing from the document, the image is considered a negative example; otherwise it is considered a positive example. The program also generates a spreadsheet to report the positive and negative images of all the glyphs and the number of occurrences of the glyph in the positive images.

The OpenCV *Objectmaker* utility [50] was used to manually define the position of the glyph in each positive document image. The utility shows each image separately to allow the user to manually define the bounding rectangle(s) of the glyph. It then moves to the next image, until all have been analysed. The utility generates a text file containing a list of each image name, number of occurrences of the glyph, and containing rectangle(s). This process was completed manually by two different people; one to make the selection of the glyphs and the other to validate the selection. This process was a very laborious process and was very time consuming, but produced an excellent research resource.

Table 2: Number of negative and positive images of glyphs in four locations

Glyph Name	Isolated		Start		Middle		End	
	Pos.	Neg.	Pos.	Neg.	Pos.	Neg.	Pos.	Neg.
Alef	198	25					207	16
Beh	178	186	203	20	190	33	91	132
Hah	52	500	133	90	133	90	47	317
Dal	182	182					170	53
Reh	242	122					194	29
Seen	24	340	101	122	132	91	39	325
Sad	26	338	40	183	84	139	25	339
Tah	9	355	61	303	68	155	34	330
Ain	23	341	146	77	131	92	40	183
Feh	29	335	178	45	141	82	34	330
Qaf	34	330					19	345
Kaf	21	343	88	135	90	133	29	335
Lam	92	272	203	20	137	86	68	155
Meem	104	260	172	51	114	109	58	165
Noon	145	219					90	133
Heh	152	212	88	135	108	115	188	35
Waw	314	50					170	53
Yeh	101	263					187	36
Hamza	40	324						
Lam Alef	206	158					54	169

Table 2 shows the total number of original negative and positive images for each glyph in their four different locations. The total number of positive images is 6,657 while the total number of negative images is 10,941.

Study of the relationship between the total numbers of positive and negative images for each glyph shows three different categories of Arabic glyphs. These are:

1. Glyphs that exist in almost all of the images. These glyphs are Alef isolated (ا), Alef end (ـا) and Lam start (لـ). They have a very small number of negative images, and sometimes no negative images at all. This problem was solved by editing the images containing a small number of the glyphs removing them manually by placing a small section of background over them. These images were then moved from the positive to the negative dataset. Figure 3 (a) shows a positive image of Alef end (ـا), while Figure 3 (b) shows the same document after removing the Alef end glyph and converting it to a negative image.
2. Glyphs that rarely appear in the images and which have very small number of positive images. These glyphs were left without any image editing because any editing carried out to add a glyph would be very

artificial. However, good results are not expected from their classifiers. For example Tah isolated **(ط)** and Ain isolated (ع).

3. Most of the glyphs have a reasonable ratio between negative and positive images.

(a) A positive document image of Alef end

(b) Converting positive image to negative

Fig. 3: Converting a positive glyph to a negative

Some glyphs such as Lam start (لـ) sometimes have another ligature glyph when followed by certain letters such as Meem or Jeem, in some fonts like Traditional Arabic. For example (لمـ) becomes (لـ). These new ligatures were ignored in our experiment.

Selecting rectangles to represent the glyphs in the positive files can be challenging when dots are presented. Using the glyphs without the dots was tried first, as in Figure 4 (a), but this method gave poor results because it usually missed parts of the glyphs when trying to avoid the dots. Finally the dots were included, as in Figure 4 (b), which gave better results.

(a) Glyphs defined without dots

(b) Glyphs defined with dots

Fig. 4: Glyph definitions for the experiment

4.3.2 Creating numerous samples of positive and negative images

Running the experiment requires a very large number of negative and positive images that are not directly available from the test data. A program was developed to generate the required amount of positive and negative document images by rotating the available images. The program uses two algorithms. The nearest neighbour interpolation algorithm [51] rotates the images in a constrained way without generating pixels with no value. The Box-Muller transform algorithm [52] defines the rotation angle of the document by selecting from a normal distribution.

The program attempted to produce positive and negative images totalling more than 2,000. This number of images was recommended by [3, 50, 53] as necessary for the HCC approach to run properly. The idea of generating numerous positive or negative images from a limited number of real images by rotating samples of images was recommended by Lienhart *et al.* [5].

4.3.3 Creating final testing dataset

A small sample of Arabic paragraph documents representing the variety of document types and quality that occur in real life cases were selected for use in the final stage of application test. The sample includes 10 Arabic documents containing 568 words and 2,798 letters. The average number of words in each document is 56.8 as shown in table 3.

The MMAC corpus [16] provided data for this test. The dataset was not used before the experiment and was prepared only for the final testing process.

Table 3: The final testing dataset information

Document	Doc 01	Doc 02	Doc 03	Doc 04	Doc 05	Doc 06	Doc 07	Doc 08	Doc 09	Doc 10	Total	Average
# Words	65	88	52	101	64	39	49	17	35	58	568	56.8
# Char	299	422	251	526	323	212	213	97	173	282	2798	279.8

5. Training and testing the experiment

5.1 Training (generating) the classifiers

Preparing the data and training the classifiers used in the experiment was a very lengthy process. Defining the training parameters was a particularly important step. The parameters used were the width, height, number of splits, minimum hit rate and boosting type. The training parameters were selected empirically, following auxiliary experiments.

Training size of the glyph: This is a very important issue to address. Auxiliary experiments on width and height showed that the optimum size is between 35 and 50 pixels of the sum of width and height. The ratio between them varies depending upon their average ratio.

The number of splits: This defines which weak classifier will be used in the stage classifier. If the number of splits used is one (stump), this does not allow dependencies between features to be learnt. A Classification And Regression Tree (CART) is obtained if the number of splits is more than 1, with the number of nodes smaller than the number of splits by one. The default value of number of splits is one.

Minimum hit rate: In order for the training process to move to the next stage, a minimum hit rate (TP) needs to be obtained [1, 5]. As the minimum hit rate increases it slows down the training process. The default is 0.995. Lienhart *et al.* [5] reported a value of 0.999 in their experiment which proved remarkably slow, so the two values used in our experiment were 0.995 and 0.997.

Boosting type: Experiments showed the GAB boosting type to be the best algorithm for Arabic glyph recognition. GAB was also recommended for face

detection by Lienhart *et al.* [5]. It not only gives the best results but also takes less training time than other boosting methods.

Table 4: Training information of glyphs in isolated and start location

Glyph Name	Isolated location glyphs					Start location glyphs				
	W	H	Pos. Img	Neg. Img	Total Glyphs	W	H	Pos. Img	Neg. Img	Total Glyphs
Alef	10	23	1639	1775	9592					
Beh	23	22	2144	2976	3312	12	29	1683	1420	7678
Hah	22	28	1599	5500	1848	22	30	2100	2340	3507
Dal	14	20	2192	2912	2992					
Reh	16	23	2002	2562	4004					
Seen	31	25	1278	3740	1420	26	23	1596	2562	2163
Sad	34	27	1420	3718	1420	30	24	1680	2928	1960
Tah	26	30	497	3905	497	20	28	1656	3333	1800
Ain	17	24	1278	3751	1278	14	21	1760	2387	2624
Feh	26	26	1562	3685	1633	14	22	2144	2070	3968
Qaf	27	40	1586	3630	1769					
Kaf	23	27	1136	3773	1278	17	25	1716	2160	2002
Lam	17	30	1794	2992	2028	13	30	1683	1420	5907
Meem	15	27	1638	2860	1848	14	22	2064	2091	3952
Noon	18	26	1744	2409	2240					
Heh	16	24	1824	2332	2368	15	17	1716	2160	2392
Waw	14	21	2596	2300	6193					
Yeh	22	22	1596	2893	1785					
Hamza	14	16	1680	3564	1904					
Lam Alef	16	23	1705	2528	2662					

Table 5: Training information of glyphs in middle and end location

Glyph Name	Middle location glyphs					End location glyphs				
	W	H	Pos. Img	Neg. Img	Total Glyphs	W	H	Pos. Img	Neg. Img	Total Glyphs
Alef						10	22	1716	1136	7194
Beh	12	33	2288	2178	12512	25	30	1794	2772	2080
Hah	25	20	2100	2340	3129	20	32	1656	3487	1794
Dal						14	20	2048	2173	3792
Reh						16	29	2336	2059	5632
Seen	26	24	2079	2366	2961	27	26	1680	3575	1904
Sad	23	20	1638	2224	2288	31	24	1349	3729	1491
Tah	18	27	1581	2480	1984	20	24	1586	3630	1647
Ain	14	25	2079	2392	3045	18	31	1680	2928	1906
Feh	14	25	1696	2132	2752	20	15	1586	3630	1647
Qaf						20	28	1065	3795	1065
Kaf	15	25	1768	2793	2418	20	20	1562	3685	1633
Lam	10	26	1648	2236	2544	17	25	1581	2480	1922
Meem	15	20	1806	2289	2520	15	26	1584	2640	1872
Noon						18	22	1768	2793	2522
Heh	18	33	1701	2415	2352	17	25	2256	2135	5184
Waw						17	28	2048	2173	4064
Yeh						22	17	2256	2196	4192
Hamza										
Lam Alef						18	25	1681	2704	1804

Tables 4 and 5 show the glyphs in their four different locations, the trained width and height, the total number of negative images, total number of positive images and the total number of glyphs in the positive images.

The total number of positive images in all glyphs is 106,324 while the total number of negative images is 168,241. The total number of glyphs in all positive images is 181,874. The average number of glyphs in the positive image is 1.71. The average width of glyphs is 18.9 while the average height is 24.9. The average number of positive images is 1,743 and for negative 2,758.

5.2 Testing the classifiers

The testing process was separated into two distinct parts. The first used the *performance* utility available in OpenCV to test the HCC glyphs classifiers. The second used the Levenshtein distance algorithm to test the two Arabic OCR applications. The first one is a state of the art application that generated by Kanoun *el al.* [7-10] using affixal approach which is guided by the structural properties of Arabic. The other one is the Readiris Pro version 10 commercial application [54].

The main concerns during the testing process were the numbers of True Positive (TP), False Negative (FN) and False Positive (FP) responses [55].

5.2.1 Testing HCC using the OpenCV performance utility

The process of testing the classifiers uses the *performance* utility offered by OpenCV. During testing this experiment tried to investigate the influence of the testing parameters over the classifier detection accuracy. There are two parameters which have a large effect on the detection accuracy; these are the scale factor and the minimum neighbours [5, 25, 53].

The scale factor affects the detection. The detection phase starts with a sliding window with the original size of the width and height and enlarges the window size depending on the scale factor. A scale factor of 1.1 means that the window is enlarged by 10% each time. Increasing the scale factor reduces detection time as the number of sliding windows decreases [5]. This experiment showed that a suitable scale factor for Arabic glyph detection is either 1.01 or 1.02. The minimum neighbour is the number of neighbour regions that are merged together during detection to form a single detected object [5]. Increasing the minimum neighbour reduces the number of False Positives (FP). The experiment showed that a suitable minimum neighbour value usually lies between 1 and 3 inclusive.

Table 6 shows the total number of positive testing images and the total number of glyphs in the positive images for each of the glyphs in every location.

The total number of positive images for all the glyphs is 34,543. The total number of all the glyphs in all the positive images is 59,498. The average number of glyphs in each positive image is 1.72; this value is almost the same as in the training phase.

The accuracy of the HCC approach was calculated to detect the glyphs using the 61 generated classifiers. A manual check was applied to calculate the accuracy of recognition of the classifiers.

Figure 5 shows a document image after application of ten classifiers of the proposed approach. Each classifier's rectangle is shown using a different colour.

Table 6: The testing information for all the glyphs in all locations

Glyph Name	Isolated		Start		Middle		End	
	Pos. Img	Total Glyphs	Pos. Img	Total Glyphs	Pos. Img	Total Glyphs	Pos. Img	Total Glyphs
Alef	539	3168					561	2211
Beh	704	992	550	2376	752	4192	572	780
Hah	533	615	693	1239	693	966	506	284
Dal	720	1120					672	1328
Reh	660	1331					768	1648
Seen	426	497	525	987	693	1155	504	504
Sad	426	426	560	560	546	624	426	426
Tah	142	142	540	648	527	589	488	549
Ain	355	426	576	944	672	861	560	560
Feh	497	497	704	1248	560	896	488	549
Qaf	488	488					284	284
Kaf	355	355	572	676	572	650	497	497
Lam	598	754	550	1980	544	784	527	744
Meem	546	546	688	1520	588	882	504	720
Noon	576	720					572	754
Heh	608	832	572	806	567	903	752	1632
Waw	858	2068					672	1280
Yeh	525	588					736	1536
Hamza	560	784						
Lam Alef	561	803					533	574

5.2.2 Testing the two Arabic OCR applications

The objective here is to compare the HCC approach against state of the art methods, as exemplified by a research application using a morphological with a geometrical analysis approach and a commercial Arabic OCR application.

Kanoun *el al.* [6] uses the morphological analysis and the geometrical analysis to identify the Arabic words. Further developments for the application by the authors in [7-13] using affixal approach (AABATAS) which is guided by the structural properties of Arabic language. A well-known commercial software also was used to perform this test; Readiris Pro 10 [56] which was used by AbdelRaouf et al. in [16].

The Levenshtein distance algorithm was used to calculate the accuracy of the two OCR applications [57].

6. Results

Experimental results showed that the HCC approach is highly successful in recognizing Arabic glyphs. High accuracy was indicated by the OpenCV testing *performance* utility as well as when compared with the two Arabic OCR applications.

6.1 Results of HCC using the OpenCV performance utility

Use of the *performance* utility in the experiment allows comparison of performance of glyph recognition in each of the four different locations in the Arabic language. The four types give different accuracy for each glyph. Tables 7 and 8 show the TP, FN and FP values for each glyph in its different locations. The last row shows the average accuracy for all the Arabic glyphs in the defined location.

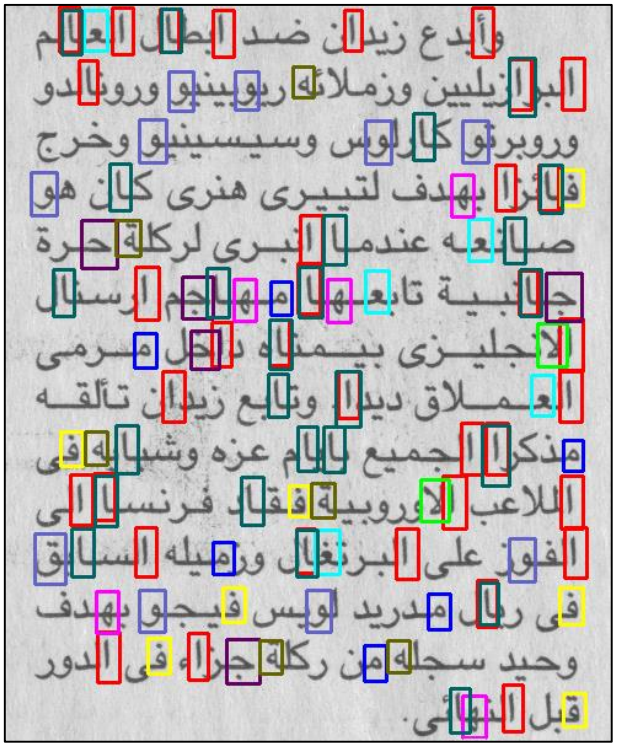


Fig. 5: Analysis of a test document image using ten HCC classifiers

Table 7: Testing results as a percentage for all glyphs in Isolated and Start locations

Glyph Name	Isolated location glyphs			Start location glyphs		
	TP	FN	FP	TP	FN	FP
Alef	94%	6%	11%			
Beh	84%	16%	14%	81%	19%	17%
Hah	76%	24%	0%	94%	6%	2%
Dal	85%	15%	5%			
Reh	73%	27%	8%			
Seen	85%	15%	1%	92%	8%	4%
Sad	83%	17%	28%	97%	3%	7%
Tah	95%	5%	4%	82%	18%	8%
Ain	100%	0%	2%	92%	8%	0%
Feh	92%	8%	29%	94%	6%	9%
Qaf	75%	25%	0%			
Kaf	97%	3%	4%	95%	5%	3%
Lam	94%	6%	2%	85%	15%	9%
Meem	92%	8%	5%	91%	9%	13%
Noon	89%	11%	9%			
Heh	86%	14%	13%	94%	6%	1%
Waw	83%	17%	16%			
Yeh	97%	3%	5%			
Hamza	78%	22%	4%			
Lam Alef	98%	2%	3%			
Average	88%	12%	8%	91%	9%	7%

6.2 Results of testing the two Arabic OCR applications

The two Arabic OCR applications achieved almost the same accuracy when tested with the final test datasets. The Readiris Pro10 commercial software achieved 85% recognition while the AABATAS research application achieved 84%. Table 9 shows the TP, FN and sensitivity values in each of the testing dataset documents when applying the two Arabic OCR applications and the HCC approach.

Table 8: Testing results as a percentage for all glyphs in Middle and End locations

Glyph Name	Middle location glyphs			End location glyphs		
	TP	FN	FP	TP	FN	FP
Alef				93%	7%	6%
Beh	74%	26%	4%	93%	7%	1%
Hah	86%	14%	18%	100%	0%	0%
Dal				93%	7%	2%
Reh				74%	26%	3%
Seen	93%	7%	6%	82%	18%	12%
Sad	89%	11%	13%	88%	12%	4%
Tah	93%	7%	24%	88%	12%	9%
Ain	92%	8%	2%	99%	1%	0%
Feh	83%	17%	9%	92%	8%	16%
Qaf				100%	0%	1%
Kaf	93%	7%	6%	93%	7%	8%
Lam	92%	8%	12%	92%	8%	0%
Meem	97%	3%	7%	90%	10%	4%
Noon				97%	3%	3%
Heh	92%	8%	0%	86%	14%	3%
Waw				98%	2%	4%
Yeh				98%	2%	3%
Hamza						
Lam Alef				96%	4%	1%
Average	89%	11%	9%	92%	8%	4%

Table 9: Testing results between the two Arabic OCR applications and HCC approach

	AABATAS results			Readiris 10 results			HCC results		
	TP	FN	Sensitivity	TP	FN	Sensitivity	TP	FN	Sensitivity
Doc01	96	203	32.1%	166	133	55.5%	269	30	90.0%
Doc02	412	10	97.6%	397	25	94.1%	409	13	96.9%
Doc03	176	14	92.6%	243	8	96.8%	220	31	87.6%
Doc04	501	25	95.2%	491	35	93.3%	457	69	86.9%
Doc05	294	29	91.0%	296	27	91.6%	285	38	88.2%
Doc06	194	18	91.5%	180	32	84.9%	165	47	77.8%
Doc07	147	66	69.0%	158	55	74.2%	190	23	89.2%
Doc08	82	15	84.5%	83	14	85.6%	84	13	86.6%
Doc09	149	47	76.0%	124	49	71.7%	131	42	75.7%
Doc10	266	16	94.3%	240	42	85.1%	224	58	79.4%
Total	2317	443	83.9%	2378	420	85.0%	2434	364	87.0%
Average	227.9	47.4	81.1%	237.6	42.0	83.1%	245.6	34.0	86.6%

6.3 Statistical analysis of the experiment results

In OCR there is a need for rigorous and correct statistical analysis approaches to compare the results of the development or modification of algorithms [58]. Different statistical techniques can decide whether the differences between the algorithms are real or random [58, 59]. In this section several statistical tests are examined for comparing our proposal and the other two Arabic OCR applications. Demsar [59] recommends using *averaging over dataset*, *paired t-test* and the *Wilcoxon signed ranks test* when comparing two algorithms using one dataset.

6.3.1 Averaging over dataset

This is the mean of errors across the dataset. It provides a gross indication of relative performance [60]. Table 9 shows that average across all the glyphs in the dataset is 87% for HCC, 85% for Readiris and 84% for AABATAS. The average across the documents in the dataset is 86.6% for HCC, 83% for Readiris and 81% for AABATAS.

Thus *averaging over dataset* shows slightly better results for the HCC approach, although it is not a significant improvement.

6.3.2 Paired t-test

Paired t-test measures whether the difference between two classifiers' results over a dataset is non-random. It checks whether the average difference in their performance over the dataset is significantly different from zero [59].

The *paired t-test* was applied between HCC AABATAS. The two-tail p-value for this t-test is p=0.69 (.69075) and t= -0.41, Using the Confidence Level (95%) value of 45.1435 in the table, the confidence interval is the mean plus or minus this value. Thus, a 95% confidence interval about Mean Difference is (-36.94, 53.34).

6.3.3 Wilcoxon signed-ranks test

The *Wilcoxon signed-ranks test* is a non-parametric alternative to the paired t-test, which ranks the differences in performances of two classifiers for each dataset, ignoring the signs, and compares the ranks for the positive and the negative differences [59]. The test was applied between HCC AABATAS.

The $R+ =21$, is the sum of ranks for the data sets on which the HCC algorithm outperformed the AABATAS, and $R- =34$, the sum of ranks for the opposite. Let T be the smaller of the sums, $T = min(R+,R-) = 21$ [59].

The critical value for T is given in the Wilcoxon Signed-Ranks Table. Here we use $\alpha = .05$ and $n = 10$. From the table we find that $T_{crit} = 8$ (two-tail test). Since $T_{crit} = 8 < 21 = T$, we can't reject the null hypothesis (i.e. $p \geq .05$), and conclude there is no significant difference between the two approaches [61].

6.4 General observations

The accuracy achieved by the HCC approach (87%) can be considered a high level of recognition at this stage and demonstrates the validity of the approach for Arabic character recognition. From the results of the experiment, the following can be observed:

- Each classifier has to detect only one glyph at a time. Some classifiers of a larger size can sometimes detect more than one glyph, for example: Tah start (ـطـ) sometimes detects Ain start and Lam middle (ـعلـ).
- Some classifiers built for different letters detect one another, such as: Seen start (ـسـ) and Sad start (ـصـ).
- The extension letter or Tatweel sometimes is detected by mistake.
- Classifiers sometimes, by mistake, detect a given glyph more than once. Additional work is needed to ensure that each glyph generates a unique response.
- Sometimes more than one classifier detects a given glyph; in most of these cases the classifiers concerned are seeking the same letter but in a different location,for example Alef end (ـا) and Alef isolated (ا).
- The space between words is clear in most cases and can be detected, although the space between PAWs is not that clear.
- Fitting text lines to the detected glyphs is clear and easy, as there are no overlaps between lines.
- Glyphs with a small number of positive or negative samples return unreasonable results, for instance most of the isolated location glyphs are not detected well.
- Glyphs with a balanced number of positive and negative samples usually return good results, for example Beh end (ـب) and Seen start (ـسـ).
- Letters with complicated visual features usually achieve better recognition accuracy, for instance Hah (ـحـ), and Lam Alef (لا).
- Using dot-less glyphs did not reduce the recognition accuracy but actually improved it.
- The glyphs Alef isolated (ا), Alef end (ـا) and Lam start (لـ) have almost the same shape and represent 23.7% of the glyphs in the language as indicated by AbdelRaouf *et al.* [16]. These glyphs require an extra post-processing algorithm to improve their recognition [16].
- There are no significant differences between recognition in the four different locations of the glyphs in the PAW.

6.5 Evaluation

The main contribution of this research is a demonstration that the Haar Cascade Classifier approach is applicable to Arabic printed character recognition. This approach eliminates steps in the traditional OCR pipeline, in particular the character segmentation phase. It can generate a full Arabic OCR application that is fast compared to commercial software, as the average document recognition time for HCC is 14.7 seconds compared with 15.8 in the Readiris commercial software. Experimental evaluation considered documents of varying size, quality and font..

The approach proposed here could be adopted when developing OCR methods for other languages using the Arabic alphabet, such as Pashto and Urdu. Those working on these languages can not only apply the approach, but can also benefit directly from this research, exploiting the glyph classifiers generated. The results obtained here suggest that the HCC approach could also be applied to cursive, handwritten characters. Although the training process of the HCC approach for handwritten characters is difficult, it will likely bring an improvement to the handwritten recognition accuracy.

The ideogram languages such as Chinese and Japanese, which are totally symbolic, could benefit from the HCC approach. In this case it might be the only approach, or a supplementary approach for first level of classification of the characters. The HCC approach could be applied to non-cursive languages such as English. The same methodology is applied by focusing on the features of a definite glyph and then detecting that glyph. The HCC approach is beneficial in this case, eliminating the need for some components of the traditional pre-processing and recognition process.

7. Future Work and Conclusion

Future enhancements for glyph recognition using Haar Cascade Classifiers can be made by keeping the glyph classifiers continually updated and enhanced by the use of more glyphs in the training dataset, which will lead to an increase in the recognition rate. The following will enhance glyph recognition:

- Modify the application of the HCC approach that was generated by OpenCV. Modifications would aim to make the algorithm more specific to character recognition or have a complete new version of the application for the character recognition. The modifications include, for example, accepting white background, dealing with a small number of positive images in the training set and rejecting an object detected more than once.
- Enhance the training of the glyphs that have a small number of positive or negative images by adding new document images to increase this small number. This enhancement of a selective number of glyphs will influence the overall recognition accuracy of the HCC approach disproportionately.
- Generate classifiers for both the Hindu and Arabic numerals for example (٣ ٢ ١) and (1 2 3) and for the Arabic special characters such as (’ ؛ ‘ ؟ ،). These glyphs are common in Arabic documents.

References

[1] P. Viola and M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features," in *IEEE Conference on Computer Vision and Pattern*

Recognition (CVPR01), Kauai, Hawaii, 2001, pp. 511-518.

[2] H. Y. Abdelazim, "Recent Trends in Arabic Character Recognition," in *The sixth Conference on Language Engineering*, Cairo - Egypt, 2006, pp. 212-249.

[3] F. Adolf. (2003, How-to build a cascade of boosted classifiers based on Haar-like features. Available: http://lab.cntl.kyutech.ac.jp/~kobalab/nishida/opencv/OpenCV_ObjectDetection_HowTo.pdf

[4] R. Lienhart and J. Maydt, "An Extended Set of Haar-like Features for Rapid Object Detection," in *IEEE International Conference of Image Processing (ICIP 2002)*, New York, USA, 2002, pp. 900-903.

[5] R. Lienhart, A. Kuranov, and V. Pisarevsky, "Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection.," in *25th Pattern Recognition Symposium (DAGM03)*, Madgeburg, Germany, 2002, pp. 297-304.

[6] S. KANOUN, I. MOALLA, A. ENNAJI, and A. M. ALIMI, "Script identification for Arabic and Latin, Printed and and written Documents " presented at the *4th IAPR-International Workshop on Document Analysis Systems: DAS*. 2000, Rio de Janeiro, Brazil, 2000.

[7] S. Kanoun, A. Ennaji, Y. Lecourtier, and A. M. Alimi, "Linguistic Integration Information in the AABATAS Arabic Text Analysis System " in *Eighth International Workshop on Frontiers in Handwriting Recognition (IWFHR'02)*, Ontario, Canada, 2002, pp. 389 - 394.

[8] S. Kanoun, A. M. Alimi, and Y. Lecourtier, "Affixal Approach for Arabic Decomposable Vocabulary Recognition: A Validation on Printed Word in Only One Font," in *The Eight International Conference on Document Analysis and Recognition (ICDAR'05)*, Seoul, Korea, 2005, pp. 1025-1029.

[9] S. Kanoun, F. Slimane, H. Guesmi, R. Ingold, A. M. Alimi, and J. Hennebert, "Affixal Approach versus Analytical Approach for Off-Line Arabic Decomposable Vocabulary Recognition," in *10th International Conference on Document Analysis and Recognition, ICDAR '09.*, Barcelona, Spain, 2007, pp. 661 - 665.

[10] F. Slimane, R. Ingold, S. Kanoun, A. M. Alimi, and J. Hennebert, "A New Arabic Printed Text Image Database and Evaluation Protocols," in *10th International Conference on Document Analysis and Recognition*, Barcelona, Spain, 2009, pp. 946-950.

[11] M. Benjelil, S. Kanoun, R. Mullot, and A. M. Alimi, "Complex documents images segmentation based on steerable pyramid features," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 13, pp. 209-228, 2010.

[12] S. B. Moussa, A. Zahour, A. Benabdelhafid, and A. M. Alimi, "New features using fractal multi-dimensions for generalized Arabic font recognition," *Pattern Recognition Letters*, vol. 31, pp. 361–371, 1 April 2010 2010.

[13] F. Slimane, S. Kanoun, J. Hennebert, A. M. Alimi, and R. Ingold, "A study on font-family and font-size recognition applied to Arabic word images at ultra-low resolution," *Pattern Recognition Letters*, vol. 34, pp. 209–218, 15 January 2013 2013.

[14] T. U. Consortium, "The Unicode Consortium. The Unicode Standard, Version 6.3," ed: Boston, MA, Addison-Wesley, 2013, pp. 195-206.

[15] F. K. Jaiem, S. Kanoun, M. Khemakhem, H. E. Abed, and J. Kardoun, "Database for Arabic Printed Text Recognition Research," pp. 251-259, 2014.

[16] A. AbdelRaouf, C. Higgins, T. Pridmore, and M. Khalil, "Building a Multi-Modal Arabic Corpus (MMAC)." *The International Journal of Document Analysis and Recognition (IJDAR)*, vol. 13, pp. 285-302, 2010.

[17] Unicode. (1991-2012, Arabic Shaping. *Unicode 6.3.0*. Available: http://unicode.org/Public/UNIDATA/ArabicShaping.txt

[18] I. Ahmed, S. A. Mahmoud, and M. T. Parvez, "Printed Arabic Text Recognition," pp. 147-168, 2014.

[19] L. M. Lorigo and V. Govindaraju, "Offline Arabic Handwriting Recognition: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, pp. 712-724, May, 2006.

[20] A. Amin, "Off line Arabic character recognition - a survey," in *the Fourth International Conference on Document Analysis and Recognition*, Ulm, Germany, 1997, pp. 596-599

[21] L. L. Muna Khayyat, Ching Y. Suen, "Learning-based word spotting system for Arabic handwritten documents," *Pattern Recognition*, vol. 47, pp. 1021–1030, March 2014 2014.

[22] A. AbdelRaouf, C. Higgins, and M. Khalil, "A Database for Arabic printed character recognition," in *The International Conference on Image Analysis and Recognition-ICIAR2008*, Póvoa de Varzim, Portugal, 2008, pp. 567-578.

[23] Y. M. Alginahi, "A survey on Arabic character segmentation," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 16, pp. 105-126, 2013.

[24] R. Harty and C. Ghaddar, "Arabic Text Recognition," *The International Arab Journal of Information Technology*, vol. 1, pp. 156-163, July 2004 2004.

[25] A. Kasinski and A. Schmidt, "The architecture and performance of the face and eyes detection system based on the Haar cascade classifiers," *Pattern Analysis and Applications*, vol. 13, pp. 197-211, 2010.

[26] F. C. Crow, "Summed-Area Tables for Texture Mapping," *SIGGRAPH Computer Graphics*, vol. 18, pp. 207-212, 1984.

[27] C. Messom and A. Barczak, "Fast and Efficient Rotated Haar-like Features using Rotated Integral Images," in *Australian Conference on Robotics and Automation (ACRA2006)*, 2006, pp. 1-6.

[28] A. AbdelRaouf, C. A. Higgins, T. Pridmore, and M. I. Khalil, "Fast Arabic Glyph Recognizer based on Haar Cascade Classifiers," presented at the International Conference on Pattern Recognition Applications and Methods (ICPRAM 2014), Angers, France, 2014.

[29] R. E. Schapire, "The Boosting Approach to Machine Learning, An Overview," in *MSRI Workshop on Nonlinear Estimation and Classification, 2002*, Berkeley, CA, USA, 2002, pp. 149-172.

[30] M. S. Khorsheed, "Off-Line Arabic Character Recognition – A Review," *Pattern Analysis & Applications*, vol. 5, pp. 31-45, 2002.

[31] A. Senior, "Off-line handwriting recognition: A review and experiments," Cambridge University, Engineering Department1992.

[32] M. Cheriet, N. Kharma, C.-L. Liu, and C. Suen, *Character Recognition Systems: A Guide for Students and Practitioners*: Wiley, 2007.

[33] A. Souza, M. Cheriet, S. Naoi, and C. Y. Suen, "Automatic Filter Selection Using Image Quality Assessment," in *the Seventh International Conference on Document Analysis and Recognition (ICDAR'03)*, Edinburgh, Scotland, 2003.

[34] I. Ahmad, "A Technique for Skew Detection of Printed Arabic Documents," in *Computer Graphics, Imaging and Visualization (CGIV), 2013 10th International Conference*, 2013, pp. 62-67.

[35] T. M. Breuel, "Robust Least Square Baseline Finding using a Branch and Bound Algorithm," in *Document Recognition and Retrieval VIII, SPIE*, ed, 2002.

[36] A. Broumandnia and J. Shanbehzadeh, "Fast Zernike wavelet moments for Farsi character recognition," *Image and Vision Computing*, vol. 25, pp. 717–726, 1 May 2007 2007.

[37] S. Touj, N. E. B. Amara, and H. Amiri, "Generalized Hough Transform for Arabic Optical Character Recognition," in *Seventh International Conference on Document Analysis and Recognition (ICDAR 2003)*, Edinburgh, Scotland, 2003, pp. 1242 - 1246.

[38] S. M. Noor, I. A. Mohammed, and L. E. George, "Handwritten Arabic (Indian) Numerals Recognition Using Fourier Descriptor and Structure Base Classifier," *Journal of Al-Nahrain University*, vol. 14, pp. 215-224, 2011.

[39] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 3rd ed.: Prentice Hall, 2007.

[40] A. Zidouri, "PCA-based Arabic Character feature extraction," in *9th International Symposium on Signal Processing and Its Applications (ISSPA 2007)*, Sharjah, United Arab Emirates, 2007, pp. 1-4.

[41] Z. Kurt, H. I. Turkmen, and M. E. Karsligil, "Linear Discriminant Analysis in Ottoman Alphabet Character Recognition," in *The European Computing Conference*, Tbilisi, Georgia, 2009, pp. 601-607.

[42] J. Trenkle, A. Gillies, E. Erlandson, S. Schlosser, and S. Cavin, "Advances In Arabic Text Recognition," in *Symposium on Document Image Understanding Technology*, Maryland, USA, 2001.

[43] I. Z. Yalniz, I. S. Altingovde, U. Güdükbay, and Ö. Ulusoy, "Integrated segmentation and recognition of connected Ottoman script," *Optical Engineering*, vol. 48, 2009/11/01 2009.

[44] N. Sabbour and F. Shafait, "A segmentation-free approach to Arabic and Urdu OCR," in *IS&T/SPIE Electronic Imaging*, 2013, pp. 86580N-86580N-12.

[45] G. A. Abandah, K. S. Younis, and M. Z. Khedher, "Handwritten Arabic character recognition using multiple classifiers based on letter form," in *the Fifth IASTED International Conference on Signal Processing, Pattern Recognition & Applications (SPPRA 2008)*, Innsbruck, Austria, 2008, pp. 128-133.

[46] S. Alma'adeed, C. Higgens, and D. Elliman, "Recognition of Off-Line Handwritten Arabic Words Using Hidden Markov Model Approach," in *the 16 th International Conference on Pattern Recognition (ICPR'02)*, Quebec, Canada, 2002, pp. 481-484.

[47] B. Bushofa and M. Spann, "Segmentation and recognition of Arabic characters by structural classification " *Image and Vision Computing*, vol. 15, pp. 167-179, 1997.

[48] R. Mehran, H. Pirsiavash, and F. Razzazi, "A Front-end OCR for Omni-font Persian/Arabic Cursive Printed Documents," in *Digital Image Computing: Techniques and Applications (DICTA'05)*, Cairns, Australia, 2005, pp. 56-64.

[49] A. F. R. Rahman and M. C. Fairhurst, "Multiple classifier decision combination strategies for character recognition: A review," *International Journal on Document Analysis and Recognition*, vol. 5, pp. 166–194, 2003.

[50] OpenCV. (2002, Rapid Object Detection With A Cascade of Boosted Classifiers Based on Haar-like Features. *OpenCV haartraining Tutorial*. Available: http://www.cognotics.com/opencv/docs/1.0/haartraining.htm

[51] M. Sonka, V. Hlavac, and R. Boyle, *Image Processing: Analysis and Machine Vision*, 2nd edition ed.: Thomson Learning Vocational, 1998.

[52] G. E. P. Box and M. E. Muller, "A Note on the Generation of Random Normal Deviates," *The Annals of Mathematical Statistics*, vol. 29, pp. 610-611, 1958.

[53] N. Seo. (2008, Tutorial: OpenCV haartraining (Rapid Object Detection With A Cascade of Boosted Classifiers Based on Haar-like Features). Available: http://note.sonots.com/SciSoftware/haartraining.html

[54] IRIS. (2011, 27/07.2011). *Readiris 12 Pro*. Available: http://www.irislink.com/c2-1684-225/Readiris-12-for-Windows.aspx

[55] R. Kohavi and F. Provost, "Glossary of Terms. Special Issue on Applications of Machine Learning and the Knowledge Discovery Process," *Machine Learning*, vol. 30, pp. 271-274, 1998.

[56] IRIS, "Readiris Pro 10," 10 ed, 2004.

[57] K. U. Schulz and S. Mihov, "Fast string correction with Levenshtein automata," *International Journal on Document Analysis and Recognition*, vol. 5, pp. 67-85, 2002.

[58] S. Garcıa and F. Herrera, "An Extension on "Statistical Comparisons of Classifiers over Multiple Data Sets" for all Pairwise Comparisons " *Journal of Machine Learning Research*, vol. 9, pp. 2677-2694, 2008.

[59] J. Demsar, "Statistical Comparisons of Classifiers over Multiple Data Sets," *Journal of Machine Learning Research*, vol. 7, pp. 1-30, 2006.

[60] G. I. Webb, "MultiBoosting: A Technique for Combining Boosting and Wagging," *Machine Learning*, vol. 40, pp. 159–196, 2000.

[61] C. Zaiontz. (2013-2015). *Real Statistics Using Excel*. Available: www.real-statistics.com