Fast Arabic Glyph Recognizer based on Haar Cascade Classifiers

Ashraf AbdelRaouf[1], Colin A. Higgins[2], Tony Pridmore[2] and Mahmoud I. Khalil[3]
[1]*Cloudypedia, Cloud brokerage company, Dubai, UAE*
[2]*School of Computer Science, The University of Nottingham, Nottingham, U.K.*
[3]*Faculty of Engineering, Ain Shams University, Cairo, Egypt*

Keywords: Arabic Character Recognition, Document Analysis & Understanding, Haar-like Features, Cascade Classifiers.

Abstract: Optical Character Recognition (OCR) is an important technology. The Arabic language lacks both the variety of OCR systems and the depth of research relative to Roman scripts. A machine learning, Haar-Cascade classifier (HCC) approach was introduced by Viola and Jones (Viola and Jones 2001) to achieve rapid object detection based on a boosted cascade Haar-like features. Here, that approach is modified for the first time to suit Arabic glyph recognition. The HCC approach eliminates problematic steps in the pre-processing and recognition phases and, most importantly, the character segmentation stage. A recognizer was produced for each of the 61 Arabic glyphs that exist after the removal of diacritical marks. These recognizers 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 proposed method is fast, with an average document recognition time of 14.7 seconds compared with 15.8 seconds for commercial software.

1 INTRODUCTION

The HCC approach was initially presented in 2001. (Viola and Jones, 2001) introduced a rapid object detection algorithm using a boosted cascade of simple features applied to face detection. Integral images were introduced as a new image representation allowing very quick computation of features. The Haar-like features were extended by adding rotated features, and an empirical analysis of different boosting algorithms with improved detection performance and computational complexity was presented in (Lienhart, Kuranov et al., 2002).

Experimental work on the novel application of the HCC approach to the recognition of Arabic glyphs is presented. First, we justify the application of the Viola and Jones approach to Arabic character recognition, then section 2 briefly reviews the theoretical basis of the HCC method. An experiment is then described in which all the Arabic naked glyphs (Arabic glyphs after the removal of diacritical marks) are recognised, with the results presented in section 3. The paper concludes with a discussion of the usefulness of the HCC approach in Arabic character recognition in section 4.

1.1 The Challenge of Arabic Character Recognition

Research in OCR faces common difficulties regardless of approach. A method is needed to distinguish ink from non-ink, skew detection and correction algorithms are needed to correct rotational scanning error and a normalization algorithm is also required to scale the document so input and model glyphs are the same size (Al-Marakeby, Kimura et al., 2013, Alginahi, 2013).

The Arabic language causes additional difficulties. It is cursive, so a sophisticated character segmentation algorithm is needed if the word is to be segmented into its consituent glyphs. Character segmentation is one of the bottlenecks of current Arabic character recognition systems (Abdelazim, 2006, Naz, Hayat et al., 2013). We suggest that skipping the character segmentation process could improve Arabic character recognition success rates.

This section provides an overview of Arabic script and discusses the problems facing the developer of an OCR application (AbdelRaouf, Higgins et al., 2008, AbdelRaouf, Higgins et al., 2010).

1.1.1 Key Features of Written Arabic

Arabic script is rich and complex. Most notably:

- It consists of 28 letters (Consortium, 2003) written from right to left. It is cursive even when printed and letters are connected by the baseline of the word. No distinction is made between capital and lower-case letters.
- Dots are used to differentiate between letters. There are 19 “joining groups” (Consortium, 2013), each of which contains multiple similar letters which differ in the number and place of dots. For example (خ ح ج) have the same joining group (ح) but different dots. The root character is referred to as a glyph.
- Arabic script incorporates ligatures such as Lam Alef (لا) which actually consists of two letters (ا ل) but when connected produce another glyph (Alginahi, 2013).
- Arabic words consist of one or more sub-words, called PAWs (Pieces of Arabic Word) (Slimane, Kanoun et al., 2013, Lorigo and Govindaraju May, 2006), PAWS without dots are called naked PAWS (AbdelRaouf, Higgins et al. 2010)).
- Arabic letters have four different shapes according to their location in the word (Lorigo and Govindaraju May, 2006); start, middle, end and isolated.
- Arabic font files exist which are similar to the old form of Arabic writing. For example a statement with an Arabic Transparent font like (احمد يلعب في الحديقة) when written in Andalus old shape font becomes (احمد يلعب في الحديقة).

1.2 Implementing the Haar-Cascade Classifier (HCC) Approach

The HCC approach (Viola and Jones, 2001) is implemented in the Open Computer Vision library (OpenCV). OpenCV is an open source library of Computer Vision functions. It is aimed at real time computer vision applications using C/C++ and Python. (Bradski, 2000; Bradski and Kaehler, 2008).

1.2.1 Faces and Glyphs

The HCC approach was originally intended for face detection. There are, however, important similarities between faces and Arabic glyphs:

- Like faces, most Arabic glyphs have clear and distinguishing visual features.
- Arabic characters are connected, and recognition requires individual glyphs to be picked out from a document image.
- Characters can have many font sizes and may also be rotated, similar to size and orientation differences in face images.
- Facial images may vary considerably, reflecting gender, age and race. The use of different fonts introduces similar variations into images of Arabic glyphs.

Each glyph can be considered a different object to be detected and having a distinct classifier. That glyph classifier will detect its glyph, ignoring others, and so becomes a glyph recogniser.

1.2.2 Training and Testing

Two sets of training images are needed; a positive set contains images which include at least one target and a negative set containing images which without any target objects. A further positive set is required for testing. Each positive set includes a list file containing the image name(s) and the position(s) of the object(s) inside each image.

As each Arabic letter can appear in four locations (hence four glyphs) a total of 100 datasets and classifiers are needed.

1.2.3 Advantages

Combining the feature extraction with the classification stages in HCC facilitates the process of training and testing the many glyphs that must be recognised. The HCC approach is scale invariant and so removes the need for explicit normalization. Using extended, rotated features also removes the need for skew detection and correction. HCC is applied directly to grey-scale images, removing the need for a binarization algorithm. This leads to a segmentationless process.

2 THEORETICAL BACKGROUND

The HCC is a machine learning approach that successfully combines three basic ideas. The first is an image representation that allows the features to be computed very quickly (integral image). Secondly, an extensive set of features that can be computed in a short and constant time. Finally, a cascade of gradually more complex classifiers results in fast and efficient detection (Viola and Jones 2001, Kasinski and Schmidt 2010, Wang, Deng et al., 2013).

2.1 Integral Image

Integral images were first used in feature extraction by (Viola and Jones, 2001). (Lienhart and Maydt, 2002) developed the algorithm by adding the rotated integral image (Summed Area Table - SAT). This 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. This is defined by the pixel of interest and the upper left corner of the image (Crow, 1984).

The integral image at any location (x, y) is equal to the sum of the pixels' intensities (grey-scale) from the $(0, 0)$ to (x, y) as shown in Figure 1 (a) and is known as $\mathrm{SAT}(x, y)$.

$$\mathrm{SAT}(x, y) = \sum_{x' \le x, y' \le y} I(x', y') \qquad (1)$$

The $\mathrm{RecSum}(r)$ for the upright rectangle $r = (x, y, w, h, 0°)$ as shown in Figure 1 (b) is:

$$\begin{aligned}\mathrm{RecSum}(r) = {} & \mathrm{SAT}(x-1, y-1) \\ & + \mathrm{SAT}(x+w-1, y+h \\ & - 1) \qquad (2) \\ & - \mathrm{SAT}(x+w-1, y-1) \\ & - \mathrm{SAT}(x-1, y+h-1)\end{aligned}$$

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 as shown in Figure 1 (c) and is known as $\mathrm{RSAT}(x, y)$.

$$\begin{aligned}& \mathrm{RSAT}(x, y) \\ & = \sum\nolimits_{(y^{\wedge\prime} \le y, y^{\wedge\prime} \le y - |x - x^{\wedge\prime}|\,)} I(x^{\wedge\prime}, y^{\wedge\prime}\,)\end{aligned} \qquad (3)$$

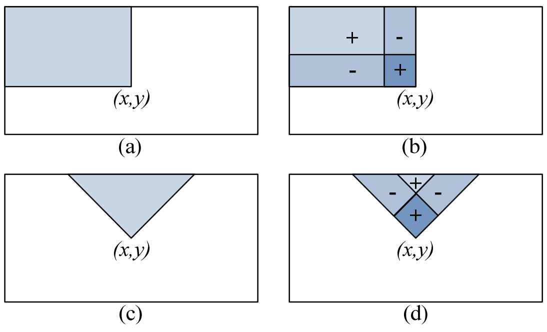


Figure 1: Image illustration of SAT and RSAT.

The RecSum (r) for the upright rectangle $r = (x, y, w, h, 45°)$ as shown in Figure 1 (d) is:

$$\begin{aligned}RecSum(r) = {} & RSAT(x - h + w, y + w + h \\ & - 1) \\ & - RSAT(x - h, y + h - 1) \qquad (4) \\ & - RSAT(x + w, y + w - 1) \\ & + RSAT(x, y - 1)\end{aligned}$$

2.2 Haar-like Feature Extraction

Haar-like feature extraction captures basic visual features of objects. It uses grey-scale differences between rectangles in order to extract object features (Viola and Jones, 2001). 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 (Messom and Barczak, 2006). The Haar-like features are computed in short and constant time

Following (Lienhart, Kuranov et al., 2002) we assume that the Haar-like features for an object lie within a window of $W \times H$ of pixels, which can be defined in Equation 5:

$$\text{features} = \sum_{i \in I}^{N} \omega_i \cdot \mathrm{RecSum}(r_i) \qquad (5)$$

where ω_i is a weighting factor which has a default value of 0.995 (Lienhart, Kuranov et al., 2002). A rectangle is specified by five parameters $r = (x, y, w, h, \alpha)$ and its pixel sum is denoted by $\mathrm{RecSum}(r_i)$ as explained in Equations 2 and 4. Two examples of such rectangles are given in Figure 2.

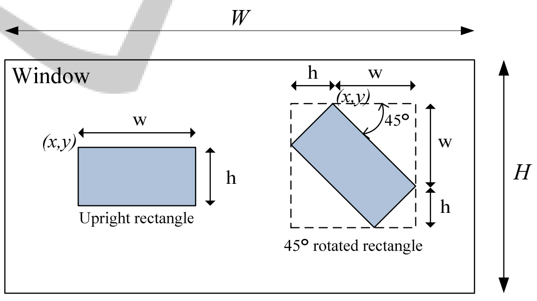


Figure 2: Upright and 45° detection windows.

Equation 5 generates an almost infinite feature set, which must be reduced in any practical application. The 15 feature prototypes are shown in Figure 3: (1) four edge features, (2) eight line features, (3) two centre-surround features, and (4) a special diagonal line feature. This set of features was scaled in the horizontal and vertical directions. Edge features (a) and (b), line features (a) and (c) and the special diagonal line feature were first used in (Papageorgiou, Oren et al., 1998, Mohan, Papageorgiou et al., 2001, Viola and Jones, 2001). They took as the value of a two-rectangle feature (edge features) the difference between the sum of the pixels in the two regions. A three-rectangle feature (line features) subtracts the sum of the two outside rectangles from the sum of the middle rectangle. A four-rectangle feature (special diagonal line feature) subtracts the sum of the two diagonal pairs of rectangles (as in Figure 3).

(Lienhart and Maydt, 2002) 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.

The number of features differs among prototypes. E.g., a 24×24 window gives 117,941 features (Lienhart, Kuranov et al. 2002).

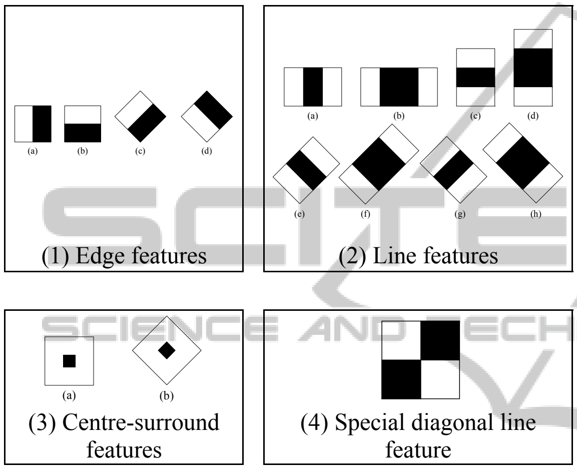


Figure 3: Haar-like feature prototypes.

2.3 Cascade of Boosting Classifiers

A classifier cascade is a decision tree that depends upon the rejection of non-object regions (Figure 4). Boosting algorithms use a large set of weak classifiers in order to generate a powerful classifier. Weak classifiers discriminate required objects from non-objects simply and quickly. Only one weak classifier is used at each stage and each depends on a binary threshold decision or small Classification And Regression Tree (CART) for up to four features at a time (Schapire 2002).

3 EXPERIMENTS

A successful pilot experiment made to investigate the HCC approach for Arabic glyphs and test the methods applicability. A single Arabic letter, Ain (ع), was used in its isolated form. 88.6% recognition was achieved which showed that the HCC approach is suitable for printed Arabic character recognition.

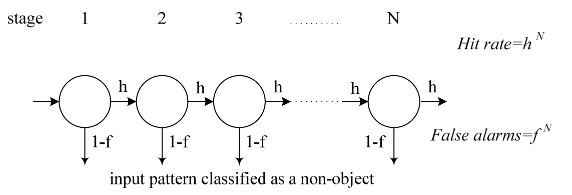


Figure 4: Cascade of classifiers with N stages.

3.1 Planning the Experiment

The hypothesis to be tested is that HCC allows the pre-processing, binarization, skew correction, normalisation and segmentation phases typically associated with OCR to be skipped. The experimental steps are:

- The binarization and noise removal step is skipped and the original grey-scale images used.
- 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 designed to generate rotated images for testing).
- The text lines detection step is skipped. Each glyph is detected along with its location in the document image implying the location of the lines.
- The normalization step is not needed because the HCC approach is scale invariant. For that reason, different font sizes are used and tested.
- The character segmentation phase can be omitted when using the HCC approach. Thus the system was trained and tested using real Arabic document images.

The following aspects were addressed in the experiment:

- Naked glyphs were used in the experiment to reduce the number of classifiers generated for classification which also reduces the recognition duration. It is easy to later locate and count the dots in the recognized glyph (Abdelazim, 2006).
- There are 18 naked Arabic letters (Unicode 1991-2006). (AbdelRaouf, Higgins et al. 2010) showed that adding Hamza (ء) and Lam Alef (لا) is essential; Table 1 shows all the naked Arabic glyphs used in the experiment.

3.2 Data Preparation

The datasets used in the experiment are images of real and computer generate Arabic documents which act as both negative and positive images. Negative

Table 1: Arabic naked glyphs as used in the experiment.

Letter English name	Arabic Letter	Isolated glyphs	Start glyphs	Middle glyphs	End glyphs
ALEF	ا	ا			ـا
BEH	ب ت ث	ب ت ث	بـ تـ ثـ نـ يـ ئـ	ـبـ ـتـ ـثـ ـنـ ـيـ ـئـ	ـب ـت ـث
HAH	ج ح خ	ج ح خ	جـ حـ خـ	ـجـ ـحـ ـخـ	ـج ـح ـخ
DAL	د ذ	د ذ			ـد ـذ
REH	ر ز	ر ز			ـر ـز
SEEN	س ش	س ش	سـ شـ	ـسـ ـشـ	ـس ـش
SAD	ص ض	ص ض	صـ ضـ	ـصـ ـضـ	ـص ـض
TAH	ط ظ	ط ظ	طـ ظـ	ـطـ ـظـ	ـط ـظ
AIN	ع غ	ع غ	عـ غـ	ـعـ ـغـ	ـع ـغ
FEH	ف	ف	فـ قـ	ـفـ ـقـ	ـف
QAF	ق	ق			ـق
KAF	ك	ك	كـ	ـكـ	ـك
LAM	ل	ل	لـ	ـلـ	ـل
MEEM	م	م	مـ	ـمـ	ـم
NOON	ن	ن			ـن
HEH	ه	ه ة	هـ	ـهـ	ـه ـة
WAW	و ؤ	و			ـو ـؤ
YEH	ى ي ئ	ى			ـى ـي ـئ
HAMZA	ء	ء			
LAM ALEF	لا	لا			ـلا

and positive images were generated for each glyph.

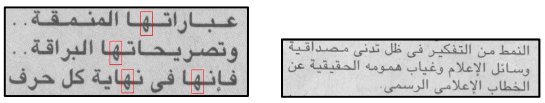
The MMAC corpus (AbdelRaouf, Higgins et al., 2010) provided data for this experiment. The data originated from 15 different Arabic document images. Five of these documents were from scanned documents (Real data), five documents were computer generated (Computer data), and the remaining five were computer generated with artificial noise added (Noise data). The Computer and Noise data used different Arabic fonts, sizes, bold and italic.

3.2.1 Creating Positive and Negative Images

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

- *Positive images:* This set of images is separated into two sub-sets; one for training and the other for testing. The testing sub-set accounted for 25% of the total positive images, while the training sub-set was 75% (Adolf 2003). Figure 5 (a) shows a sample of a positive image for the Heh middle glyph (ـهـ).
- *Negative images:* These are used for the training process of the classifiers. Figure 5 (b) shows a sample of a negative image for Heh middle (ـهـ).

A program separated the positive from the negative images for each glyph. The *Objectmaker* utility offered in OpenCV (OpenCV, 2002) was used to manually define the position of the glyph in each positive document image. This process was very labour intensive and time consuming, but produced an excellent research resource.



عبـاراتـهـا المنـمـقـة..
وتصـريـحـاتـهـا البراقـة..
فـإنـهـا فى نـهـايـة كل حرف

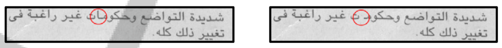
النمط من التفكير فى ظل تدنى مصداقية
وسائل الإعلام وغياب همومه الحقيقية عن
الخطاب الإعلامى الرسمى.

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

Figure 5: Sample of positive and negative image for Heh middle glyph (ـهـ).

For the 61 glyphs, the total number of positive images was 6,657 while the total number of negative images was 10,941. The relationship between the total numbers of positive and negative images for each glyph shows the following three different categories of Arabic glyphs:

- *Glyphs that exist in almost all of the images.* These glyphs are Alef isolated (ا), Alef end (ـا) and Lam start (لـ). These have very few negative images, so more were generated by manually masking out images that contained only one or two glyphs. Figure 6 (a) shows a positive image of the glyph Alef end. Figure 6 (b) shows the same document after converting it to a negative image.



شديدة التواضع وحكومات غير راغبة فى
تغيير ذلك كله.

شديدة التواضع وحكو ت غير راغبة فى
تغيير ذلك كله.

(a) A positive document image of Alef end (b) Converting positive image to negative

Figure 6: Sample of converting a positive image to negative.

- *Glyphs that rarely appear in the images* and so have a very small number of positive images. Any editing here would be too artificial, however, good results are not expected from their classifiers (e.g. Tah isolated (ط)).
- Most of the glyphs have a reasonable ratio of negative to positive images.

3.2.2 Creating Numerous Samples of Positive and Negative Images

The experiment requires a huge amount of images which are not immediately available. Software was developed to generate more positive and negative document images from the available test images, see (Lienhart, Kuranov et al., 2002). This uses two algorithms; the nearest neighbour interpolation algorithm (Sonka, Hlavac et al., 1998) to rotate the images; the Box-Muller transform algorithm (Box and Muller, 1958) to generate a normal distribution of random numbers from the computer generated uniform distribution. The document rotation angles use σ=5 and μ = 0. μ as the mean value and it is considered to be angle 0 as the scanning process may be skewed in any direction with the same value. Sigma (σ) is the standard deviation of the angles (i.e. σ=5 means that 68.2% falls within ±5°).

The program produced a total of over 2000 positive and negative images as recommended by (OpenCV, 2002; Adolf, 2003; Seo, 2008) which allowed the HCC approach to run properly.

3.3 Training the Classifiers

Preparing the files and folders for the training process was a lengthy process. The training itself additionally took around a year. Each training process took two days on average. As each glyph has on average 3 different trials and there are 61 glyphs this gives around a year of continuous work on a single dedicated computer.

3.3.1 Defining Training Parameters

It was important to define the training parameters before the experiment. The parameters used were the width, height, number of splits, minimum hit rate and boosting type. These parameters were chosen following the results and conclusions of auxiliary experiments.

Training size of the glyph: This is a very important issue to address. The optimum width and height of the training size was empirically determined to achieve the best classification results. Experiment showed the optimal value for the sum of width and height is between 35 to 50 pixels.

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 learning dependencies between features. 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 (Barros, Basgalupp et al., 2011). 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 (Viola and Jones, 2001). As the minimum hit rate increases it slows down the training process. The default value is 0.995. (Lienhart, Kuranov et al., 2002) reported a value of 0.999 which proved remarkably slow so a range of 0.995 to 0.997 was used.

Boosting type: The experiment showed that the GAB boosting type is the best algorithm for Arabic glyph recognition and is similar to face detection (Lienhart, Kuranov et al., 2002).

3.3.2 Training Statistics

The total number of positive images for all glyphs is 106,324, the number of negative images is 168,241. The total number of all glyphs in all positive images is 181,874. The average number of glyphs in each positive image is 1.71. The average width of all glyphs is 18.9 while the average height is 24.9. There are 1743 positive and 2758 negative images.

Table 2 shows, for each location, the trained width and height, the total number of positive and negative images and total number of glyphs.

Table 2: Training information of glyphs in all locations.

Position	Isolated	Start	Middle	End
Average Width	20.05	17.90	17.27	19.21
Average Height	25.20	24.64	25.27	24.68
Average No. of Positive images	1,645.5	1,799.8	1,853.1	1,749.1
Average No. of Negative images	3,190.2	2,261.0	2,349.5	2,827.4
Average No. of glyphs	2,603.5	3,450.3	3,500.4	2,807.6
Total No. of Positive images	32,910	19,798	20,384	33,232
Total No. of Negative images	63,805	24,871	25,845	53,720
Total No. of glyphs	52,071	37,953	38,505	53,345

3.4 Testing the Classifiers

The testing process of the experiment was separated into two distinct parts. The first used the *performance* utility available in OpenCV. The second tested the HCC glyphs classifier against real commercial software.

The main concerns in the testing process were the values of True Positive (TP), False Negative (FN) and False Positive (FP) ratios in all tests (Kohavi and Provost, 1998). TP is the number or ratio of the glyphs that were detected correctly. FP is when glyphs are detected wrongly. FN is when glyphs are not detected at all even though they exist.

3.4.1 Testing using the OpenCV Performance Utility

This experiment tried to investigate the influence of the testing parameters on detection accuracy. Two parameters have a very big effect on detection accuracy; the scale factor and the minimum neighbours (Lienhart, Kuranov et al., 2002; Seo, 2008; Kasinski and Schmidt, 2010).

The detection phase starts with a sliding window using the original width and height and enlarges the window size depending on the scale factor (Lienhart, Kuranov et al., 2002). A suitable scale factor for Arabic glyph detection was found to be 1.01 or 1.02. The minimum neighbour is the number of neighbour regions that are merged together during detection in order to form one object (Lienhart, Kuranov et al., 2002). A suitable minimum neighbour value was found to lie between 1 and 3 inclusive.

Table 3 shows the minimum, average and maximum number of positive testing images and the total number of glyphs in the positive images for each location.

Table 3: The testing information of glyphs in all locations.

Position	Isolated	Start	Middle	End
Minimum No. of Positive images	142	525	527	284
Average No. of Positive images	533.9	593.6	610.4	559.1
Maximum No. of Positive images	858	704	752	768
Minimum No. of glyphs in images	142	560	589	284
Average No. of glyphs in images	857.6	1,180.4	1,136.5	887.4
Maximum No. of glyphs in images	3,168	2,376	4,192	2,211

3.4.2 Testing using a Commercial Arabic OCR Application

Here the HCC approach is compared to well known commercial Arabic OCR software. This gives realistic measures of the performance of the HCC approach. All the glyphs classifiers of the HCC approach are tested against Readiris Pro 10 (IRIS 2004) which was used in (AbdelRaouf, Higgins et al., 2010).

A new small sample of Arabic paragraph documents that represents a variety of document types was used. The sample includes 37 Arabic documents with 568 words and 2,798 letters.

The Levenshtein distance algorithm used by (AbdelRaouf, Higgins et al., 2010) was used to calculate the commercial software accuracy. The accuracy of the HCC approach was calculated after running the *performance* tool offered by OpenCV to detect the glyphs using the 61 generated classifiers.

3.5 Experiment Results

Results showed that the HCC approach is highly successful. High accuracy was produced using the OpenCV testing utility as well as when compared with commercial software. Two sets of results were obtained, one for the OpenCV *performance* utility and the other for the commercial software.

3.5.1 Results using the OpenCV Performance Utility

The four location types give different accuracy for

Table 4: Testing information of glyphs in all locations.

Position	Isolated	Start	Middle	End
Minimum TP	73%	81%	74%	74%
Average TP	88%	91%	89%	92%
Maximum TP	100%	97%	97%	100%
Minimum FN	0%	3%	3%	0%
Average FN	12%	9%	11%	8%
Maximum FN	27%	19%	26%	26%
Minimum FP	0%	0%	0%	0%
Average FP	8%	7%	9%	4%
Maximum FP	29%	17%	24%	16%

each glyph. Table 4 shows the statistical results of TP, FN and FP values in the four different locations.

3.5.2 Results of Testing with the Commercial OCR Application

The HCC approach achieved marginally better accuracy (87%) than that of the commercial software (85%).

3.5.3 Experiment Results Comments

▪ The HCC accuracy achieved (87%) is high at this stage of the work and proves the validity of the approach.

- The glyphs with a small number of samples return unreasonable results, as expected (e.g. most isolated location glyphs).
- Glyphs with a balanced number of samples return good results, for example Beh end (ـب).
- The glyphs with complicated visual features get higher recognition accuracy, as expected e.g. Hah (ح), Sad (ص) and Lam Alef (لا).
- Using naked glyphs improved the recognition rate and reduced the number of classifiers.
- The higher the number of glyph samples the better the recognition accuracy.

4 CONCLUSIONS AND FURTHER WORK

The HCC approach is applicable to Arabic printed character recognition. This approach eliminates pre-processing and character segmentation. A complete, fast, Arabic OCR application is possible with this technique (average documentation time for HCC (14.7 seconds) is comparable with commercial software (15.8 seconds).

Enhancements can be obtained by keeping the glyph classifiers continually updated. The training of the glyphs that have a small number of positive or negative images can be improved by adding new document images. Hindu and Arabic numerals for example (3 2 1) and (1 2 3) and for the Arabic special characters such as (’ ؛ ‘ ؟ ،) could be added.

REFERENCES

Abdelazim, H. Y. (2006). *Recent Trends in Arabic Character Recognition*. The sixth Conference on Language Engineering, Cairo - Egypt, The Egyptian Society of Language Engineering.

AbdelRaouf, A., C. Higgins and M. Khalil (2008). *A Database for Arabic printed character recognition*. The International Conference on Image Analysis and Recognition-ICIAR2008, Póvoa de Varzim, Portugal, Springer Lecture Notes in Computer Science (LNCS) series.

AbdelRaouf, A., C. Higgins, T. Pridmore and M. Khalil (2010). "Building a Multi-Modal Arabic Corpus (MMAC)." *The International Journal of Document Analysis and Recognition (IJDAR)* 13(4): 285-302.

Adolf, F. (2003) "How-to build a cascade of boosted classifiers based on Haar-like features.".

Al-Marakeby, A., F. Kimura, M. Zaki and A. Rashid (2013). "Design of an Embedded Arabic Optical Character Recognition." *Journal of Signal Processing Systems* 70(3): 249-258.

Alginahi, Y. M. (2013). "A survey on Arabic character segmentation." *International Journal on Document Analysis and Recognition (IJDAR)* 16(2): 105-126.

Barros, R. C., M. a. P. Basgalupp, A. e. C. P. L. F. d. Carvalho and A. A. Freitas (2011). "A Survey of Evolutionary Algorithms for Decision-Tree Induction." *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* pp(99): 1-22.

Box, G. E. P. and M. E. Muller (1958). "A Note on the Generation of Random Normal Deviates." *The Annals of Mathematical Statistics* 29(2): 610-611.

Bradski, G. (2000). "The OpenCV Library." *Dr. Dobb's Journal of Software Tools*.

Bradski, G. and A. Kaehler (2008). *Learning OpenCV: Computer Vision with the OpenCV Library*, O'Reilly Media, Inc.

Consortium, T. U. (2003). The Unicode Consortium. The Unicode Standard, Version 4.1.0, Boston, MA, Addison-Wesley: 195-206.

Consortium, T. U. (2013). The Unicode Consortium. The Unicode Standard, Version 6.3, Boston, MA, Addison-Wesley: 195-206.

Crow, F. C. (1984). "Summed-Area Tables for Texture Mapping." *SIGGRAPH Computer Graphics* 18(3): 207-212.

IRIS (2004). Readiris Pro 10.

Kasinski, A. and A. Schmidt (2010). "The architecture and performance of the face and eyes detection system based on the Haar cascade classifiers." *Pattern Analysis and Applications* 13(2): 197-211.

Kohavi, R. and F. Provost (1998). "Glossary of Terms. Special Issue on Applications of Machine Learning and the Knowledge Discovery Process." *Machine Learning* 30: 271-274.

Lienhart, R., A. Kuranov and V. Pisarevsky (2002). *Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection.* 25th Pattern Recognition Symposium (DAGM03), Madgeburg, Germany.

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

Lorigo, L. M. and V. Govindaraju (May, 2006). "Offline Arabic Handwriting Recognition: A Survey." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28(5): 712-724.

Messom, C. and A. Barczak (2006). *Fast and Efficient Rotated Haar-like Features using Rotated Integral Images*. Australian Conference on Robotics and Automation (ACRA2006).

Mohan, b. A., C. Papageorgiou and T. Poggio (2001). "Example-Based Object Detection in Images by Components." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23(4): 349-361.

Naz, S., K. Hayat, M. I. Razzak, M. W. Anwar and H. Akbar (2013). *Arabic script based character segmentation: A review*. Computer and Information Technology (WCCIT), 2013 World Congress on, IEEE.

OpenCV (2002) "Rapid Object Detection With A Cascade of Boosted Classifiers Based on Haar-like Features." *OpenCV haartraining Tutorial*.

Papageorgiou, C. P., M. Oren and T. Poggio (1998). *A General Framework for Object Detection*. 6th International Conference on Computer Vision, Bombay, India: 555-562.

Schapire, R. E. (2002). *The Boosting Approach to Machine Learning, An Overview*. MSRI Workshop on Nonlinear Estimation and Classification, 2002, Berkeley, CA, USA.

Seo, N. (2008) "Tutorial: OpenCV haartraining (Rapid Object Detection With A Cascade of Boosted Classifiers Based on Haar-like Features).".

Slimane, F., S. Kanoun, J. Hennebert, R. Ingold and A. M. Alimi (2013). Benchmarking Strategy for Arabic Screen-Rendered Word Recognition. *Guide to OCR for Arabic Scripts*. H. E. A. Volker Märgner, Springer London: 423-450.

Sonka, M., V. Hlavac and R. Boyle (1998). *Image Processing: Analysis and Machine Vision*, Thomson Learning Vocational.

Unicode (1991-2006) "Arabic Shaping " *Unicode 5.0.0.*

Viola, P. and M. Jones (2001). *Rapid Object Detection using a Boosted Cascade of Simple Features*. IEEE Conference on Computer Vision and Pattern Recognition (CVPR01), Kauai, Hawaii.

Wang, G.-h., J.-c. Deng and D.-b. Zhou (2013). "Face Detection Technology Research Based on AdaBoost Algorithm and Haar Features." 1223-1231.