Pattern Recognition 48 (2015) 849–861

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

An Arabic handwriting synthesis system

Yousef Elarian [a], Irfan Ahmad [a], Sameh Awaida [b], Wasfi G. Al-Khatib [a], Abdelmalek Zidouri [a,*]

[a] King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia
[b] Qassim University, Qassim 51452, Saudi Arabia

ARTICLE INFO

Article history:
Received 3 May 2014
Received in revised form
7 August 2014
Accepted 15 September 2014
Available online 28 September 2014

Keywords:
Arabic handwriting
Handwriting synthesis
Training data
Offline text recognition
Statistical model

ABSTRACT

In this paper, we present an Arabic handwriting synthesis system. Two concatenation models to synthesize Arabic words from segmented characters are adopted: Extended-Glyphs connection and Synthetic-Extensions connection. We use our system to synthesize handwriting from a collected dataset and inject it into an expanded dataset. We experiment by training a state-of-the-art Arabic handwriting recognition system on the collected dataset, as well as on the expanded dataset, and test it on the IFN/ENIT Arabic benchmark dataset. We show significant improvement in recognition performance due to the data that was synthesized by our system.

1. Introduction

Handwriting recognition is an active area where researchers are trying various approaches to increase recognition rates [1]. Researchers agree that expanding the training set of a text recognition system is generally beneficial to recognition rates. However, conventional ways of collecting datasets can be time-consuming and may incur a lot of effort, especially for ground-truthing. Hence, researchers proposed the use of synthesized data in expanding training sets of recognition systems [2–6].

Handwriting synthesis refers to the computer generation of online and offline data that resemble human handwriting. It is a reverse process for handwriting recognition as it transforms input text into image samples, whereas recognition maps handwritten samples into digital text.

Handwriting synthesis has become a topic of rapidly increasing interest because of its applications such as the improvement of text recognition systems (in terms of overall performance [2,7], stability [8,9], and speed [10,11]), personalized fonts [12,13], and forgery detection [14,15]. Depending on the application, synthesis methods and their corresponding evaluation methods vary. Personalized fonts, for example, aim at capturing the style of a particular writer and tend to be evaluated subjectively. Whereas synthesized data for text recognition may aim at maximizing style variability within natural limits [3,16,17] and its evaluation is mainly tied to recognition rates.

Handwriting synthesis encompasses *generation* and *concatenation* operations [18,19]. Handwriting generation operations alter samples of handwriting to increase their shape-variability within some closed-vocabulary. Concatenation operations, in contrast, aim at the compilation of new units of vocabulary, such as words, from a smaller pool of basic samples, such as characters. Handwriting generation can be seen as the inverse operation of preprocessing in a text recognition system whereas handwriting concatenation can be regarded as the inverse operation of segmentation.

Synthesized data can improve systems that have deficiencies in their text segmentation accuracy, their recognition features and classifiers, or in the variability of their training data. One advantage of this approach is that it functions on the data level which is system-independent.

Arabic is a widely used language and the Arabic script is used in other languages as well like Urdu and Persian [20]. In Arabic, most characters must connect to their successor within a word. These characters take one of four character-shapes: Beginning (B), Middle (M), Ending (E), and Alone (A); the few characters that do not connect to their successors can only take the (E) or (A) character-shapes. These characters cause Arabic words to break into Pieces of Arabic Words (PAWs). From right to left, a multi-character PAW consists of one (B) character-shape followed by zero or more (M) character-shapes and is terminated by one (E) character-shape. A PAW that consists solely of one character always takes the (A) character-shape.

Characters connect in Arabic via a stroke called the Kashida [21]. Kashida are semi-horizontal strokes that often lie in the

* Corresponding author. Fax: +966 138603535.
E-mail addresses: yarian@kfupm.edu.sa (Y. Elarian), irfanics@kfupm.edu.sa (I. Ahmad), s.awaida@qu.edu.sa (S. Awaida), wasfi@kfupm.edu.sa (W.G. Al-Khatib), malek@kfupm.edu.sa (A. Zidouri).

http://dx.doi.org/10.1016/j.patcog.2014.09.013
0031-3203/© 2014 Elsevier Ltd. All rights reserved.

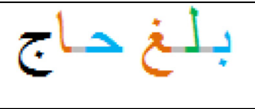
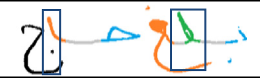
Printed (with explicit Kashida in gray)	A E K B E K M K B 	**Blue**: Beginning (B) **Green**: Middle (M) **Brown**: Ending (E) **Black**: Alone (A) **Gray**: Kashida Overlaps
Handwritten sample		

Fig. 1. Arabic printed and handwritten samples colored to distinguish their character-shapes and connection strokes called Kashida.

baseline zone of an Arabic text-line. Unlike printed text, vertical overlaps between PAWs and characters are common in Arabic handwriting. Fig. 1 presents some of the aforementioned features of Arabic writing.

In this paper, we aim at the synthesis of offline Arabic handwriting for its use in improving handwritten text recognition systems. This paper is organized as follows. In Section 2, we discuss related work. In Section 3, we present our concatenation-based synthesis system. Our experiments and results are presented in Section 4. Finally, the conclusion is summarized in Section 5.

2. Related work

There are top-down and bottom-up approaches for handwriting synthesis. Top-down approaches, also referred to as movement-simulation, aim at modeling the neuromuscular actions of writing [22–29]. Bottom-up approaches, also known as shape-simulation, model the written samples themselves [30]. Movement-simulation usually requires the acquisition of online data on tablets; hence, shape-simulation approaches may be more practical for offline data [31].

Synthesis techniques may aim at the generation or concatenation of samples. Generation techniques produce new synthesized images at the same level of the input samples they receive; e.g. new character samples from input character samples. Concatenation techniques, in contrast, produce output images at higher levels than their inputs; e.g. word samples from character samples. Generation encompasses perturbation-based, fusion-based and model-based techniques [32]. Perturbation-based techniques alter a handwritten sample using image-processing tools. Several geometric perturbations are applied on handwritten text-lines to supplement training sets of recognition systems [3,16–18]. Similarly, affine transformations by Wakahara et al. [33] and local perturbations by Keysers et al. [34] were applied for the same goal. Fusion-based, or example-based, techniques merge few samples into new hybrid ones. Among the works adopting fusion-based techniques for the expansion of training sets are those of Zheng and Doermann [32] and Viswanath et al. [35,36]. Research efforts have been little here probably because it lacks established models. Model-based techniques rely on significant numbers of online [31,37,38] or offline [39,40] samples in capturing the writing styles. Except for perturbation-based techniques, the two other generation techniques require shape-matching [32,35]. Hence, perturbation-based generation techniques are easier to implement. However, their results may appear unnatural if parameters are not calibrated carefully [3,41,42].

Concatenation operations can be performed with or without connecting the aligned units. Alignment without connection sets in juxtaposition character groups to form words and lines, as in [43–46]. Direct-connection techniques connect character tails to their heads. Arabic [47,46] and Latin [2,5,48,49] cursive text-lines are synthesized by direct-connection. More sophisticated concatenation was achieved by connection-stroke interpolation which is based on polynomial-models [44,45] spline-models [48,50,51] or probabilistic-models [2,49].

For the Arabic script, shape-simulation was first presented in Elarian et al. [46] for offline handwritten text, where we introduced the idea of sample selection. Another work, but for online recognition, was presented in [6,47]. Dinges et al. [52,53] generate and concatenate Arabic character-shapes represented by Active Shape Models. They synthesize text and modify it by affine transformations and B-Spline interpolation to obtain artificial offline handwriting.

One major application for handwriting synthesis is to improve OCR systems. Some researchers, e.g. [3,7,16], inject synthesized data to improve the original results, whereas others, e.g. [2,6,47], experiment on the synthesized data only without the original data. Bayoudh et al. [7] experiment on online writer-dependent lower-case-character Radial Basis Function Network (RBFN) and Support Vector Machines (SVM) recognizers. They inject 300 synthesized versions of the 26 English characters to the training set. Their best improvement increases the character recognition rate (CRR) by approximately 13 percents. Helmers and Bunke [2] generate data that performs approximately as well as collected data on an offline HMM recognizer whereas Varga and Bunke [3,17] perturb handwritten text-lines to expand their training set and improve their recognition rate. Saabni and El-Sana intent to find Piece of Arabic Word (PAW) recognition rates (PAWRR) for alternative online and offline training sets [6,47]. They evaluate their work on a Dynamic Time Wrapping (DTW) online recognizer [58] and adaptation of it for offline recognition. Similarly, Miyao and Maruyama synthesize a virtual Hiragana dataset that performs comparably to their original dataset [4]. The robustness, speed and performance of the recognition of online gestures are addressed in [9]. We summarize the Related Work section in Table 1.

In our current work, we present shape-simulation synthesis by direct concatenation and statistically-modeled connections to synthesize handwriting samples that look natural to improve recognition system training.

3. Synthesis of Arabic handwriting

We present an approach for synthesis of Arabic handwriting by concatenation techniques. The approach is outlined in the block diagram of Fig. 2. The approach takes character-shape images classified as strictly segmented or extended characters as inputs and concatenates them into synthesized handwriting. Important properties of the dataset are detailed in Section 3.1.

The filled rectangles in the diagram show the four steps of the synthesis procedure while the rounded rectangles indicate the information needed in each step. The connection-point location block receives character-shapes and intends to locate their connection-points so that character-shapes can be aligned to them when concatenated. This step can benefit from contextual information about the characters gathered from the database as discussed in Section 3.2. Then, some features are computed on the connection parts and on the character-shapes to help selecting

Table 1
Summary of related work with a list of abbreviations below.

Citation	Approach/task	Original DB	Original results	Synthesized data	Synthesized data results	Classifier	Injection
Bayoudh et al. [7]	• Top-down • Online • Writer-dependent recognition • Lowercase • Isolated Latin characters	• Twelve Writers • Each writing 10 versions of the 26 lowercase letters	≈ 82.5% CRR[b]	• Generation of letter 300 versions • Distortions and analogy	≈ 95% CRR	Radial basis function network	Yes
				• Image distortions	≈ 92.5% CRR		
				• On – line and Image distortions	≈ 94% CRR		
			≈ 93% CRR	• Generation of letter 300 versions by: • Distortions and analogy	≈ 96.5% CRR	Support vector machines	Yes
				• Image distortions	≈ 95.5 CRR		
				• On-line and image distortions	≈ 95.75 CRR		
Helmers and Bunke [2]	• Bottom-up • Offline • Text recognition • Latin script	• Subset of the IAM[c] database • Total of 1190 words • From a lexicon of 37 words • Training 80% • Evaluation 20%	70.8% WRR[f]	• Concatenation-based synthesis of original data	65.8% WRR	Hidden Markov models	No
				• Segmented character samples	68.5% WRR		
				• n-tuples of characters	71.1% WRR		
Varga and Bunke [3], [16]	• Bottom-up • Offline • Writer-independent • Cursive • Handwriting recognition	• 3899 word instances • 6 writers • From a lexicon of 412 words	33.1% WRR	• 5 synthesized lines added to each original line • All distortions	Substantial change 49% WRR	Hidden Markov Models	Yes
				• Line level geometrical transformations	47.1% WRR		
				• Connected component level geometrical transformations	38.7% WRR		
Saabni and El-Sana [47]	• Bottom-up • Online • Arabic • PAW-recognition	• 500 PAWs [d] • 6 different writers	81% PAWRR[e]	• Same database synthetically generated • Using concatenation	82% PAWRR	Elastic matching technique	No
Saabni and El-Sana [6]	• Bottom-up • Online and offline • Arabic • PAW recognition	• From 48 experts • ADAB [a] (online) 2200 PAWs 16 356 shapes	• 80.21% precision	PAWs generated from ADAB and experts' data	In precision. • 81.16% synthetic • 81.09% user-synthetic	Dynamic time warping	No
		• From 48 experts • IFN/ENIT (offline)	• 78.21% precision	PAWs generated from IFN/ENIT and experts' data	• 78.64% synthetic • 80.43% user-synthetic		
Miyao and Maruyama [4]	• Bottom-up • Off-line • Character recognition • Japanese hiragana	• 10 Japanese Hiragana characters • 50 writers • 50 real	≈ 97.5% CRR	• 50 samples • Using on-line affine transformation	≈ 97.3% CRR	Support vector machines	No
Plamondon et al. [11]	• Top-down • Online • Gestures • Robust increments in training	• 11 Gestures • 7 Writers • 100 each	≈ 27.5% Error rate	• 10 synthetic gestures for each real learning sample	50% Error reduction	Evolve++	Yes

[a] ADAB: Arabic data base for on-line recognition.
[b] CRR: character recognition rate.
[c] IAM: Institut für Informatik und angewandte Mathematik (Institute of Computer Science and Applied Mathematics).
[d] PAW: Piece of Arabic Word.
[e] PAWRR: PAW recognition rate.
[f] WRR: word recognition rate.

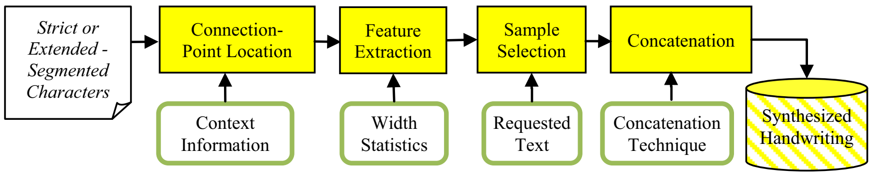
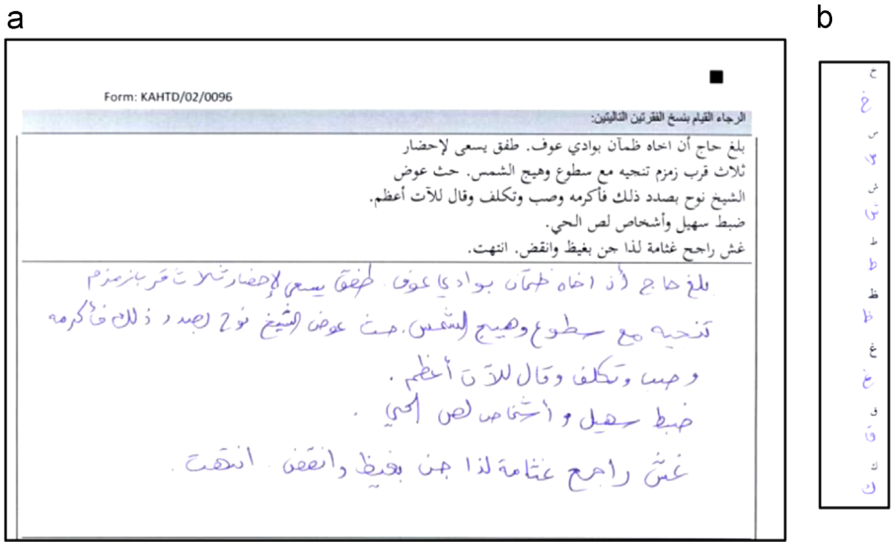


Fig. 2. Block diagram of the steps to obtain an image dataset in filled boxes.


Form: KAHTD/02/0096

الرجاء القيام بنسخ الفقرتين التاليتين:

بلغ حاج أن اخاه ظمآن بوادي عوف. طفق يسعى لإحضار
ثلاث قرب زمزم تنجيه مع سطوع وهيج الشمس. حث عوض
الشيخ نوح بصدد ذلك فأكرمه وصب وتكلف وقال للآت أعظم.
ضبط سهيل وأشخاص لص الحي.
غش راجح غثامة لذا جن بغيظ وانقض. انتهت.

بلغ حاج أن اخاه ظمآن بوادي عوف. طفق يسعى لإحضار ثلاث قرب زمزم
تنجيه مع سطوع وهيج الشمس. حث عوض الشيخ نوح بصدد ذلك فأكرمه
وصب وتكلف وقال للآت أعظم.
ضبط سهيل وأشخاص لص الحي.
غش راجح غثامة لذا جن بغيظ وانقض. انتهت.

Fig. 3. (a) A sample paragraph of the used dataset. (b) Isolated character-shapes part of the dataset.

suitable character-shape instances for concatenation. This step is further detailed in Section 3.3. The Sample Selection block represents the matching step where appropriate character-shape samples of the underlying text are selected for concatenation, based on their suitability measures on features, as described in Section 3.4. In the concatenation step, the selected samples are positioned on an image, with or without extensions, as described in Section 3.5.

3.1. *Dataset description*

For a concatenation system, the acquisition of a dataset with enough samples to synthesize the target vocabulary is essential. For this sake, we design a dataset of 54 samples each containing the text in Fig. 3(a) which covers all Arabic character-shapes, some ligatures but no digits. There are databases for digits [59]. Besides, digits always appear in their isolated form (A) and never need concatenation.

The number of unique shapes in the paragraph is 106 out of 160 total collected shapes. Adding the eight isolated character-shapes of Fig. 3(b), 168 distinct character and ligatures are covered by each dataset form.

Total counts of character-shape samples range from 38 to 486. The 'ظـ' character-shape appears in one word in the text that is written 54 times. Moreover, it is written in a position that allows the writer to make a ligature out of it and its successor; hence, we discarded 16 of these appearances. The 'ـا' character-shape is one that appears frequently in Arabic and nine times in the database text; hence, we have a total of 486 samples of it from all the 54 writers. These and other statistics are summarized in Table 2.

The dataset forms were filled by 54 native Arabic speakers and scanned at a resolution of 300 dpi before the dataset was ground-truthed at the character-level. Ground-truthing encompassed associating each pixel in an image to a label that corresponds to the character it belongs to or to a Kashida label. Finally, noise is filtered out based on the size of connected components. The dataset ground-truth uses a common label for components that do not constitute parts of the characters, but that appear with them such as remaining noise components and Kashida extensions. We extract 1174 of these by the Extraction process explained in Section 3.5.2.

Table 2
General statistics on our dataset.

Statistic	Value
Total number of words	2322
Total number of character-shape samples	8799
Average number of character-shape samples	83.8
Total number of explicit Kashida extensions	1174
Maximum number of character-shape samples with repetition (for 'ـا')	486
Minimum number of character-shape samples (for 'ظـ') discarding those that participate in ligatures	38

3.2. *Connection-point location*

Connection-point location is necessary for the feature extraction and concatenation steps. We investigate connection-point location for two scenarios: with and without aid of ground-truth information. The former scenario can only be used when the data is ground-truthed to the pixel level, which is not a common case. The latter scenario is adequate for images that result from blind segmentation but is more prone to errors that occur in identifying the baseline zone.

When ground-truth information is not considered, connection points can be methodically located from character-shape images based on their baseline zone and on the distance from the right and left edges of their containing bounding boxes. We search for connection points at the right side of (E) and (M) character-shapes and at the left side of (B) and (M) character-shapes. Examples of (B), (M) and (E) character-shape extensions are shown in Fig. 4.

We choose connection points at character parts that are nearest to the corresponding edges of their containing bounding boxes and to the baseline range of the character in a special distance measure. The distance measure doubles the vertical component of the distance (measured from the baseline level) to favor the most proximate to the edges. The doubling factor of the distance is chosen based on empirical trials on our dataset. Since the baseline cannot be accurately estimated with a single character-shape, we receive baseline information that is computed on chunks of characters from the dataset.

We only search for connection-points in the sides where the characters have them, as per their (B), (M) and (E) contexts, as depicted in the algorithm of Fig. 5.

We report error rates for our methodical connection-point location algorithm based on 1462 ground-truthed samples with labels for the connection parts. The average error rate of our approach is 1.88%.

3.3. *Feature for character selection*

We mainly extract features from the connection-parts (Kashida features) but also use the relative-widths of the character-shapes as a *Width* feature.

Kashida features are intended to assure matching characters within a single Piece of Arabic Word (PAW) in a smooth way. They measure the thickness and the direction of connection-parts within a window of N columns from the corresponding edge of the bounding box of the character-shape image. N is chosen based on the resolutions of the images and the average font sizes such that features are computed on Kashidas and avoid reaching the character bodies. In our case where the resolution is 300 dpi and where the forms encourage normal sizes for written character shapes, the range of 5 to 9 pixels is found reasonable. We choose N as 7.

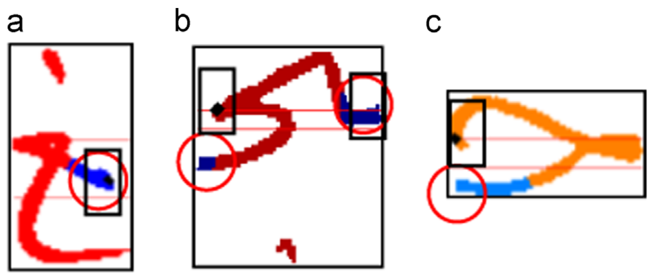


Fig. 4. Correct extensions (in circles) and erroneous extension location samples (in small rectangles) for (a) an ending character-shape, (b) a middle character-shape and (c) a beginning character-shape.

The thickness at Column j is taken as the vertical distance between the upper and the lower contours of the connection-part. The direction is taken as the difference between the relative middle y-co-ordinate of the connection part pixels at Column j and the corresponding value for Column $j+1$. Hence, N thickness features and N-1 direction features can be computed per connection-part. Kashida features are illustrated in Fig. 6(a).

The thickness and direction features at a Column j are formalized in Eqs. (1) and (2), respectively.

$$Thickness_j = |UCD_j - LCD_j| \tag{1}$$

$$Direction_j = (UCD_j + LCD_j)/2 - \ (UCD_{j+1} + LCD_{j+1})/2 \tag{2}$$

where the UCD and LCD stand for the Upper and Lower Contour Directions and j denotes an arbitrary column in the image.

The width-ratio feature (*Width*) refers to the ratio of the width of a character-shape sample to the average width of its class samples. The average widths per sample are pre-computed and stored for their use in this feature. Fig. 6(b) illustrates the effect of the Width feature.

We compute and store all features in a $2 \times N$ sized structure. Kashida features are stored so that the values of the right outer pixels are matched with the values of the left inner pixels, and vise versa, as depicted in Fig. 7.

The width-ratio features are matched separately because they typically have smaller values than thickness. Width-ratio is multiplied by a factor before being considered in the matching. The Kashida features are divided by the window size to cope for the N values. N equals to 7 was found to be a reasonable choice by encouraging a semi-standard font size in our collected forms. The weighted matching is used to select samples of character shapes for synthesis, as detailed in the next section.

3.4. *Sample selection*

The particular samples of the character-shapes that are to contribute to the synthesis of a given text need to be selected so that they collaboratively pursue a fitting appearance. The features of neighboring samples are evaluated by the Manhattan (city-block) distance measure. The collection of samples that minimizes the sum of the measured distances is selected. When synthesizing several versions of a Piece of Arabic Word (PAW), we assure each selection is unique.

The search space of sample selection is affected by the number of *units* (U) to be jointly selected and by the number of samples per character-shape. The value of U represents the count of the units needed for concatenation, whether character-shapes or Kashidas.

```
Algorithm: Connection-Point Location
Input: Arabic Handwritten Images of Characters I
Character Context C containing A, B, M or E
Baseline Ranges [u, d]
Output: Sets of Connection-Point Locations (each as an (x, y) ordered pair)
Procedure:
If (character context of Image I is B or M)
  Set H as the x-coordinate of the left edge of the bounding box of I
If (character context of Image I is E or M)
  Set H as the x-coordinate of the right edge of the bounding box of I
For each foreground pixel p in I
  If (the y-coordinate of p is within [u,d])
    Set V to 0
  Else
    Set V to the nearest to the y-coordinate of p from u and d.
  Compute distance D as $|H - p_{x\text{-coordinate}}| + 2*|V- p_{y\text{-coordinate}}|$)
Select p* with minimum distance D and return its x-coordinate and y-coordinate
END Algorithm
```

Fig. 5. Algorithm for connection-point location.

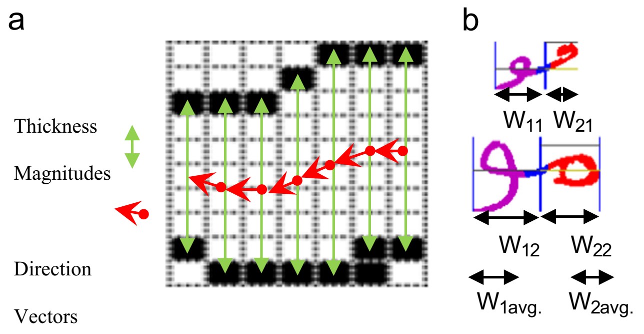


Fig. 6. Illustration of (a) the features of the 7 leftmost pixels of a left connection part and (b) the two matches based on the width-ratio feature.

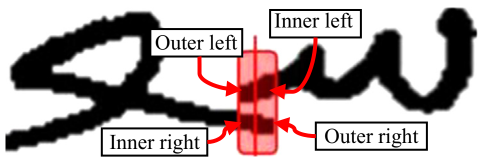


Fig. 7. Comparing inner and outer features of a left and a right Kashidas.

We estimate the number of comparisons required for a selection by *Comparisons(U)*, the number of distance matching for a unit of U character-shapes. In the following, let U_i be the number of samples of the ith character-shape in the synthesized unit. Eq. (3) estimates the search space for brute-force selection.

$$Comparisons(U) = \prod_{i=1}^{U} U_i \tag{3}$$

Brute-force search for sample selection is impractical except for small values of U. One solution to this problem is to limit the usage of brute-force selection to PAWs, since more than 99.5% of PAWs consist of 5 or less character-shapes [54]. Then, the different PAWs are linked based on the width features of their two neighboring characters.

Another approach that avoids intractable brute-force selection is the forward algorithm [55] that performs optimal matching for the first pair of the character-shapes and sequentially matches neighboring character-shapes in a chain Eq. (4), below, represents the number of vector comparisons for our greedy forward algorithm.

$$Comparisons(U) = U_1 \times U_2 + \sum_{i=3}^{U} U \tag{4}$$

Curtailed and broken connection parts may result in thickness values of zero. When matching features-structures for sample selection, the zero thickness features may undesirably match. For this reason, we discard zero-thickness extension parts.

3.5. Concatenation

In this step, images of cursive text are composed from individual character-shape samples. This is accomplished through one of two concatenation approaches, viz., the Extended Glyph approach (EG) and the Synthetic-Extension approach (SE).

The aggregation of the character-shape with its Kashida in a single unit, as illustrated in Fig. 8(a), is referred to as an extended glyph and is the basis of the EG approach. Extended-glyphs can be of the beginning, middle or ending shapes, denoted as (*Bx*), (*xMx*) and (*xE*), respectively; where the '*x*' prefix/suffix indicates the presence of a Kashida extension before/after a character-shape. The regular expression of a multi-character PAW under this model is given by $(Bx)(xMx)^*(xE)$, where '*' indicates zero or more occurrences of middle character-shapes before the ending character-shape.

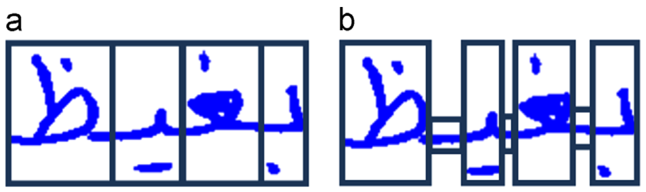


Fig. 8. Examples of (a) Extended-Glyphs connection model consisting of 4 EG units, and (b) synthetic extensions connection model consisting of 4 units separated by 3 synthetic extensions.

On the other hand, SE concatenation utilizes synthesized Kashida between character-shapes that were extracted with minimal Kashida extensions (K), as shown in Fig. 8(b). The regular expression for SE concatenation is given by $(B)(K(M))^*K(E)$. The search space of samples is larger in SE than in EG due to the greater number of units in SE. Below, we discuss some issues of the EG and the SE models.

3.5.1. Extended glyph connection model

Extended-glyphs are extracted from the dataset described in Section 3.1. They include the character-shapes along with their neighboring Kashida extensions. Then, the Kashida extensions are trimmed so that they are only few (2–6) pixels out of the extended glyph. Trimming extensions of the extended character-shape model not only keeps the extension length natural, but also leaves the connection point at a clean cut.

The EG model uses direct-connection, as introduced in Section 2, to concatenate Piece of Arabic Word (PAW) and no-connection between PAWs. Extended character-shapes are placed in juxtaposition where character-shapes within a PAW are vertically aligned so that their extensions overlap as it was illustrated in Fig. 7.

Then, spaces are added between PAWs and words. We randomize the widths of these spaces, within specified constrains, in order to increase the variability of the outputs. For example, if the text to be synthesized explicitly specifies a space, we insert a gap of a size drawn from some probability distribution. These values are selected by inspection of the original dataset.

Another type of spacing is the displacement made between PAWs. The displacement values are selected from a certain distribution preferably producing positive and negative values that correspond to more gapping or probable overlapping displacements, respectively. We choose the mean and the standard deviation values by inspecting the original dataset to mimic gaps in real data. Clearly the distribution favors gaps over overlaps.

3.5.2. Synthetic-extension connection model

The Synthetic-Extension (SE) model uses a modeled-connection stroke to concatenate characters in PAWs. The rest of the procedure is similar to that of Extended-Glyph (EG). Hence, in this section, we explain connection stroke analysis, modeling and synthesis.

A statistical model is designed to learn the shapes of Kashida in our dataset. It analyses the features of extracted Kashidas and estimates Probability Density Functions (PDFs) that capture their trends. The PDFs are later used to render a Kashida. The following sections elaborate on Kashida extraction, representation and modeling.

3.5.2.1. Kashida extraction. Kashida extensions are extracted from the dataset (Section 3.1) based on their ground-truth labels. All Kashida and noise components share a common label value. Hence, to isolate Kashida from noise components, we constrain the extracted components to be adjacent to two consecutive characters. The labels of these neighboring characters are stored

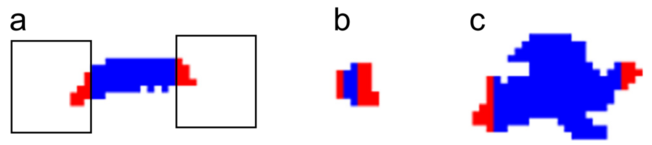


Fig. 9. Samples of (a) trimmed Kashida and (b) Kashida discarded based on size (c) Kashida discarded based on the aspect ratio.

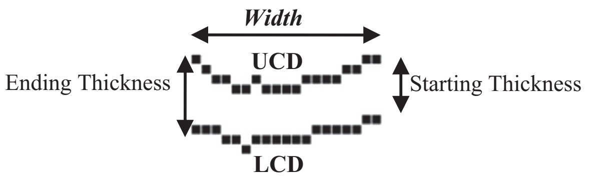


Fig. 10. Kashida Width, upper contour directions (UCD) and lower contour directions (LCD).

along with the corresponding Kashida for the later computation of conditional statistics that depend on these characters.

To make clean edges for Kashidas, we vertically trim few pixels from the left and right sides of them. Kashidas are also filtered based on size and aspect ratio thresholds. Fig. 9 displays samples of trimmed and discarded Kashida.

3.5.2.2. Kashida representation. Each extracted Kashida is represented by three sets of features: its width (*Width*), the directions of its upper contour (UCD) and the directions of its lower contour (LCD). Fig. 10 shows these features. Note that we do not need to model Kashida starting nor Kashida ending thicknesses. The starting thicknesses are dictated by the previous character to be concatenated, and the ending thicknesses are dictated by the synthesis process.

We show how we identify *Width*, UCD and LCD of our previously extracted Kashida as in the algorithm that is listed in Fig. 11. The Kashidas collectively contribute with more than 1000 widths and 13,000 pixel-directions (slopes) for each of the UCD and LCD. In the next section, we discuss computing the PDFs for these features that model Arabic Kashida under different scenarios.

3.5.2.3. PDF estimation. The probability density functions (PDFs) for Width, UCD, and LCD of Kashida are computed for the Kashidas as well as for subsets of the Kashida population. The Kashidas are divided into subsets per their writers, per the characters they emerge from, and per the characters they reach.

Two types of PDFs are estimated: *Width* PDFs and Contour PDFs. *Width* PDFs are estimated based on bins that are eight pixels wide. Strokes shorter than a certain threshold are filtered out during extraction; hence, the first bin is usually under-populated. Contour PDFs are estimated for the upper and the lower contours.

We compute upper contour PDFs (UCD-PDFs) for a whole Kashida as well as for each of five equal portions of its UCD. We also compute UCD-PDFs conditionally on the UCD value of the predecessor column. Lower contour PDFs (LCD-PDFs) are also estimated both: independently and conditionally on the corresponding UCD value.

Table 3 summarizes the Kashida PDFs and their conditional subsets. Together, these PDFs sum to 2459 *Width* and contour statistic. To represent a Kashida, we need to select a PDF from each of the three row sets that appear between thick horizontal borders in the table. From these, the *Width*, 5-Portioned UCD-PDFs and Conditional-on-Upper LCD-PDFs (highlighted in the table) are chosen to model and synthesize our Kashida. We discard the plain UCD and LCD PDFs because they bear less amount of context than conditional PDFs. The conditional UCD PDF is discarded because it is too sensitive to noise in the Kashida UCD. This is because the conditional UCD makes the selection of a PDF solely conditional on one previous pixel direction.

To choose a conditional subset of the representative classes of PDFs, we inspected all PDFs and noticed that the “conditional on the next character” subset captured some writing styles features. For example, only two *Width* histograms were non-descending, and when inspected, these resulted to belong to the “conditional on the next character” subset of two characters that are often written over a flying Kashida. Other interesting notes are reported here for the histograms of Figs. 12–14. The positive bin numbers indicate ascending directions while the negative bin numbers indicate descending directions, each by their indicated magnitudes.

From Fig. 12, we can see that the trend to descend in the first portions of a Kashida is more than it is in the last ones. Both Figs. 13 and 14 show histograms that are conditional on UCD values.

Fig. 13 shows that it is more probable to have a descending LCD when the corresponding UCD is straight or descending. Similarly, when the corresponding UCD is ascending, it is more probable to have an ascending LCD. In all cases, a straight LCD is most probable. This coupling between corresponding UCD and LCD values is expected.

Fig. 14 shows that if the previous UCD is straight, it is about 4 times more probable that the next UCD will be descending than ascending. Only when the previous UCD is ascending by a slope of 2, the current UCD is more probable to be ascending. On the other hand, only when the previous UCD is descending by a slope of 2 it is highly probable that the current UCD will be also descending by the same slope. These trends can also be justified from the usual handwriting styles; however, the main benefit of the PDFs is to quantify the trends so that they can be used as statistical models for our Kashida synthesis that is described next.

3.5.2.4. Kashida synthesis. To synthesize a Kashida, we draw a width value *W* from the *Width*-PDF, then, we draw *W/5* UCD values from each of the portion-PDFs of the 5-portioned UCD and *W* LCD values conditional on the corresponding UCD. We impose some constraints on the distance between each UCD and its corresponding LCD values so that the Kashida thickness is always within a pre-specified range. Once the contours are selected, we fill the range between them with black pixels. Two samples are shown in Fig. 15. We use synthesized Kashidas similar to these to concatenate strictly segmented character-shapes in the Synthetic-Extension (SE) approach.

4. Experimental results

Synthesized images can be evaluated both subjectively and objectively. In this section, we present some results of our handwriting synthesis system and investigate their impact on the performance of a state-of-the-art text recognizer [1]. We synthesize text as per the settings detailed in Section 4.1 and present their recognition results in Section 4.2.

4.1. Synthesis Results

In this section, we present the settings and results of our synthesis system. We use the dataset described in Section 3.1 as a source of the character-shapes. The dataset contains 2322 words where the character-shape with minimum occurrence is repeated 38 times (Table 2).

We synthesize Tunisian towns/villages names from IFN/ENIT, a popular database of handwritten Arabic images [56]. We synthesize 721 out of the 937 names that do not contain digits; because

```
Algorithm: Kashida Features Extraction
Input: Arabic Handwritten Images with Character-Level Ground-Truth Labels
Output: Arrays for the values of Width, UCD and LCD
Procedure:
For each labeled image i:
For each Kashida connected components K:
Filter out noise based on the size & aspect ratio
Trim leftmost and rightmost pixel slices of K
Compute and Save the width Width of K in Array Widths
Compute $\frac{dK}{dy}$ to obtain upper and lower contours of K and put the y-coordinates of upper and lower contours of K in Vectors UC and LC, respectively.
Compute $\frac{dUC_y}{dx}$ and $\frac{dLC_y}{dx}$ to obtain UCD and LCD, respectively.        // slope vectors of UD and LD
Reverse UCD and LCD to suit the Arabic right-to-left writing direction.
Save UCD and LCD in Arrays UCDs and LCDs for later statistics
END Algorithm
```

Fig. 11. Algorithm Kashida features extraction.

Table 3
The computed PDFs per Kashida subsets and types along with their sizes in terms of the Kashida width *W*.

Sets PDF	Statistic per Kashida	Proper set	Subset per previous character-shape	Subset per next character-shape	Subset per writer
Width	1	1×1	42×1	**50×1**	44×1
UCD	*W*	1×1	42×1	50×1	44×1
Conditional UCD	$(W-1)$	1×5	42×5	50×5	44×5
5-Portioned UCD	$(W/5)$	1×5	42×5	**50×5**	44×5
LCD	*W*	1×1	42×1	50×1	44×1
Conditional LCD	*W*	1×5	42×5	**50×5**	44×5

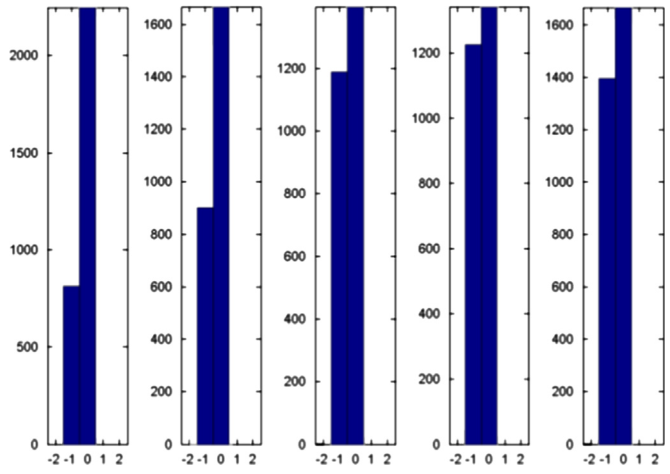


Fig. 12. 5-Portioned UCD histograms where the rightmost histogram corresponds to the first (rightmost) portion of a Kashida.

our source dataset does not cover digits. Table 4 displays some statistics on the text that we synthesize.

A set of parameters affects the quality of synthesis and the time it consumes. These parameters are shown in Table 5.

We synthesize six versions of the (possibly multi-word) names. To synthesize unique versions of the same word, the combinations of character shapes that form a PAW are prevented from appearing again.

Fig. 16 shows some samples of the results of our extended-glyphs (EG) and synthetic-extension (SE) synthesis along with the printed versions of their corresponding names for comparison. With our features, we noticed that the connections of the EG images are smooth enough to fool the native eye. The connections synthesized by the SE approach have relatively less natural appearance, but they allow for more character-shape samples for PAW sample-selection.

We show cases that represent some of our synthesis issues in Fig. 17. The first row shows synthesized images for city names that contain three occurrences of the character Alef 'ا' (encircled in the figure). Alef 'ا' can bear diacritics such as Hamza 'ء' and Madda '~' in some Arabic contexts. The issue of writing or omitting these diacritics is taken lightly in informal Arabic handwriting. Hence, the dataset contains cases where these diacritics are automatically labeled inconsistently with the written images. In case of 'أولاد هلال', for example, a diacritic is supposed to appear on the rightmost Alef 'ا' but not on the two other Alef instances. However, this is opposite to the shown synthesized images in both of our EG and SE samples that are illustrated in the figure. This problem is from the dataset and can be solved by manual correction of the labels for Alef 'ا' instances.

The second row of the figure shows two additional issues that can also appear in both of our approaches. In the second row of Fig. 17, the encircled Alef 'ا' in the EG sample of the word 'العرايس' appears in a relatively lower position than usual. Such position is determined based on its baseline information which is received from the dataset. The baseline is computed on chunks of words and hence, can be inaccurate for some character-shapes. This well-known problem of baseline estimation is dealt with in the literature. Fortunately, it does not have dramatic consequences on the naturalness of our synthesized images.

Although the previous problem is not evident in the SE sample of the same word, such sample shows another problem with the relative height of the Yaa 'يـ' character-shape. The encircled Yaa 'يـ' cusp appears too high compared to the rest of the character-shapes in the word image. This occurs because we do not have measures to assure relative height compatibility in selecting character-shape samples as the ones we have for relative widths. We can implement this feature in some future update of the system.

In the third row of Fig. 17, the EG sample shows a Kashida that might be perceived as too long (encircled). Such long/short Kashidas can occur in EG because the algorithm selects character-shapes for concatenation without consideration of their respective extensions lengths. In general, this is not a problem. This can even benefit the variability of the results. The SE approach avoids this minor issue by making the length of the Kashida rely on a PDF that is conditional to the next character-shape.

The shown SE sample illustrates a case of an unsmooth synthesized Kashida (encircled in the figure) caused by the limited context (dependence on previous values) in the synthesis

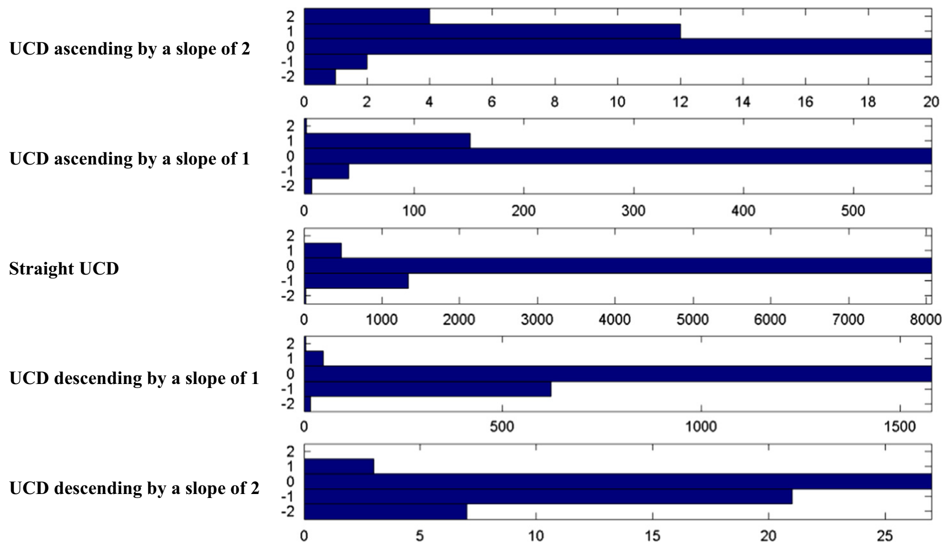


Fig. 13. Conditional-on-UCD LCD histograms.

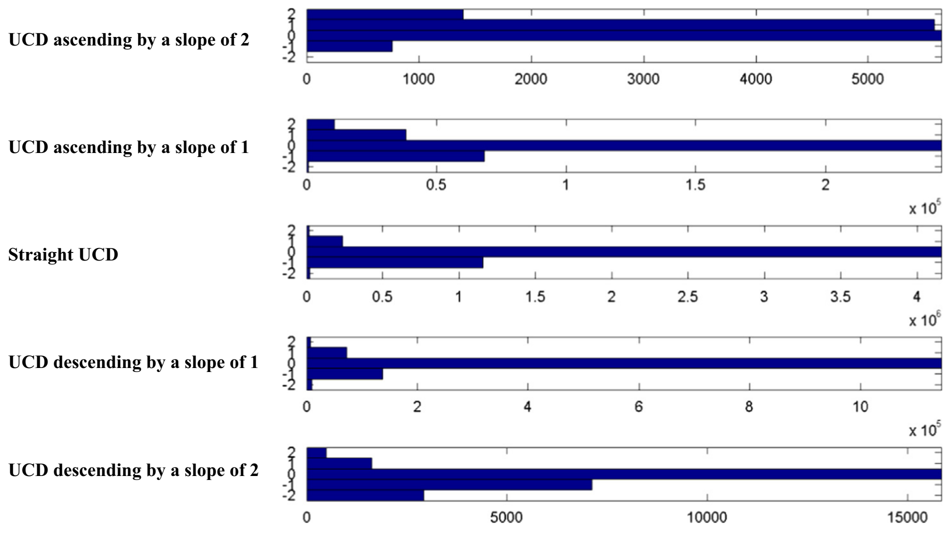


Fig. 14. Conditional on previous UCD histograms.

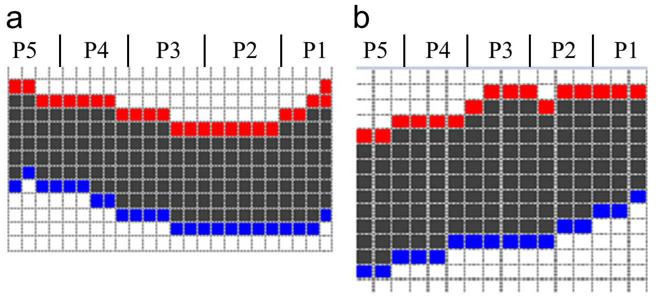


Fig. 15. Synthesized Kashidas (a) with the overall upper contour PDF and (b) with the portion-wise upper PDFs. Approximate portions are indicated above the images.

Table 4
General statistics on our synthesis test bed.

Feature	Value
Total PAWs (Pieces of Arabic Words)	1445
Total character-shapes	3847
Avg. number of character-shapes per town name	5.34
Maximum number of character-shapes in a town name	13
Avg. number of character-shapes per PAW	2.66
Maximum number of character-shapes in a PAW	7
Avg. number of PAWs per town name	2.00
Number of PAWs with 1 character-shape	64
Number of PAWs with 2 character-shapes	709
Number of PAWs with 3 character-shapes	422
Number of PAWs with 4 character-shapes	174
Number of PAWs with 5 character-shapes	55
Number of PAWs with 6 character-shapes	19
Number of PAWs with 7 character-shapes	2

algorithm. The same problem can be seen in the fourth sample of the SE image, where its position is encircled.

On the other hand, the EG sample of the same word manifests another interesting problem; which is the miss-location of the connection-point of the Yaa 'ـيـ' character-shape. The left connection-point of the encircled character-shape Yaa 'ـيـ' is erroneously located at the dots that come below the character. This miss-location leads to the connection of the dots with the next character-shape instead of the character-body that appears floating above the PAW in the last row of Fig. 17. This is because the writer has used a too long stroke below the character body instead of the two separated dots of the character-shape Yaa.

4.2. OCR Results

In this section, we intend to demonstrate the possibility of benefitting from the injection of synthesized data into the training set of text recognition systems. We evaluate our system on Set *'d'* and Set *'e'* of the IFN/ENIT database consisting of 937 city names [56]. We selected set *'d'* and set *'e'* for evaluation as it is commonly used by other researchers for evaluating their text recognition systems. As we do not use any data from IFN/ENIT for training our system, in principle, we could have used any set for evaluation.

Table 5
Setup parameters for the synthesis.

Setting	Value
Maximum number of character-shapes for brute force selection	2
Zero-thickness penalty	Yes
WT weight for the W/W_{avg} features	10
Intra-PAW spacing distribution	Uniform between 14 and 28 pixels
Inter-PAW spacing distribution	Normal distribution with a mean of 5 and a standard deviation of 1.75
Threshold to discard short Kashida in SE	Shorter than 6 pixels

Our text recognition system is a continuous Hidden Markov Model (HMM) based system that was implemented using the HMM ToolKit (HTK) [55]. A left-to-right HMM with Bakis topology was used. Each character shape was modeled with a separate HMM. Each character-shape HMM is modeled with the same number of states. We extracted nine statistical features from the word images. These features are adapted from [1,57]. We append nine derivative features to the original features such that the dimension of the feature vector is 18. The optimal number of states is decided based on the evaluation results for Set *'d'*.

Our baseline system is trained on the 2322 collected word samples from the dataset described in Section 3.1. We inject samples of the Extended Glyph (EG) concatenation model for one set of experiments and samples of the SE concatenation model for another set of experiments. We experiment on incremental numbers of injected data and summarize the results in Table 6. We report the top 1 word recognition rates (WRR), along with the statistical significance at 95% confidence level. In addition, we also

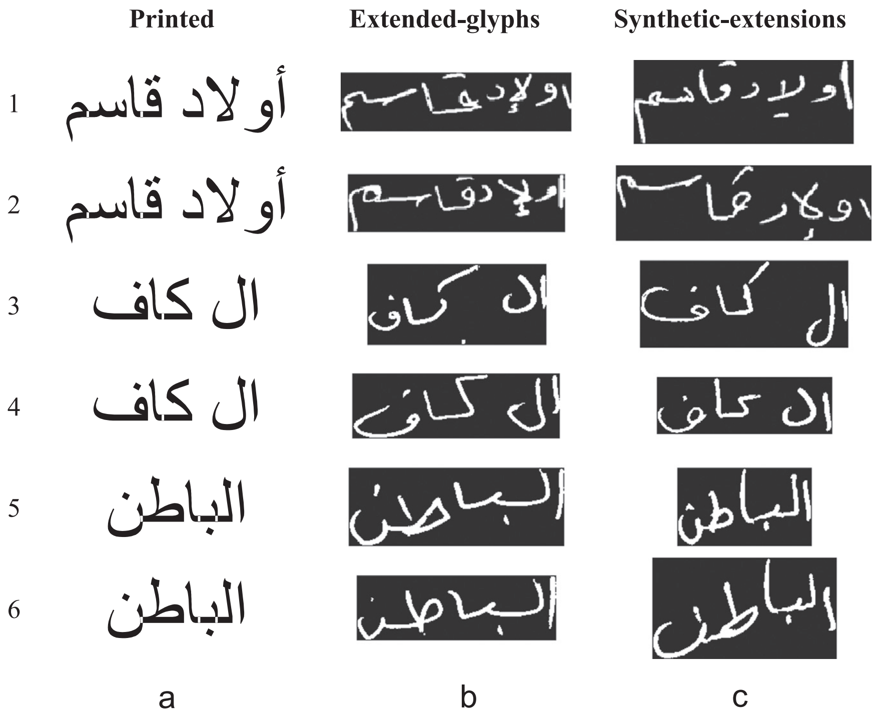


Fig. 16. The printed text (a) along with some samples of our extended-glyphs (b) and synthetic-extension (c) synthesized images for three city names of IFN/ENIT.

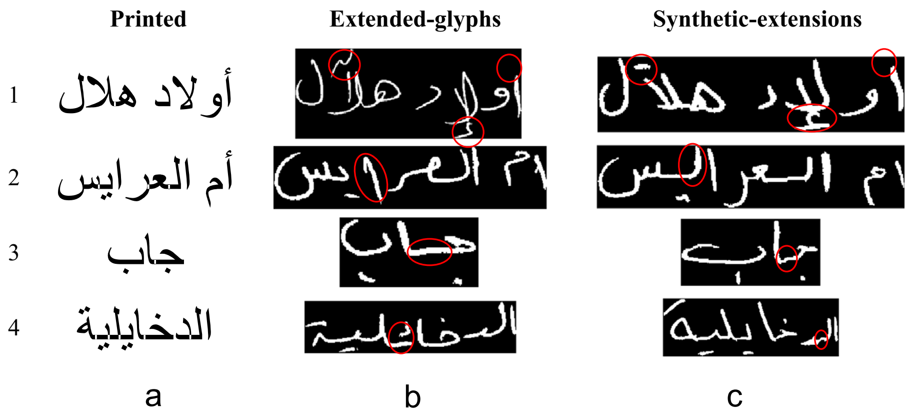


Fig. 17. Samples illustrating some printed city names (a) along with some bad results of their synthesis for the extended-glyphs (b) and synthetic-extensions (c) approaches.

report the top 5, and the top 10 best results. In general, SE results are higher than GE results due, in part, to the SE components' variability.

We note that injecting synthesized data significantly increases the word recognition accuracy by about 16%. It is also noted that injecting six different samples of the city names produces the highest WRR at 70.13%, in which the injected samples plateaus afterwards. Adding more samples has no statistically significant impact on the WRR. The relative stabilization in the WRR after six samples can be due to lack of diversity in the synthesized city names samples, or for reaching the limits of the implemented features and classifier.

Table 6
Results of injecting different number of 'SE' synthesized samples in the original training data in percentage.

Number of samples injected for each of the 721 city names	Word recognition rates (WRR)			
	Top 1	Statistical significance	Top 5	Top 10
Baseline system	48.52	(±1.00)	64.17	67.74
One sample	64.51	(±0.97)	78.09	81.67
Two samples	66.76	(±0.95)	81.05	84.09
Three samples	67.86	(±0.94)	81.66	84.68
Four samples	69.00	(±0.94)	82.67	85.38
Five samples	69.18	(±0.94)	82.05	84.89
Six samples	**70.13**	**(±0.93)**	**82.94**	**85.53**
Seven samples	69.82	(±0.93)	82.62	85.42
Eight samples	69.29	(±0.93)	82.54	85.55
Nine samples	69.74	(±0.93)	82.89	85.59
Twelve samples	70.58	(±0.92)	84.22	87.03

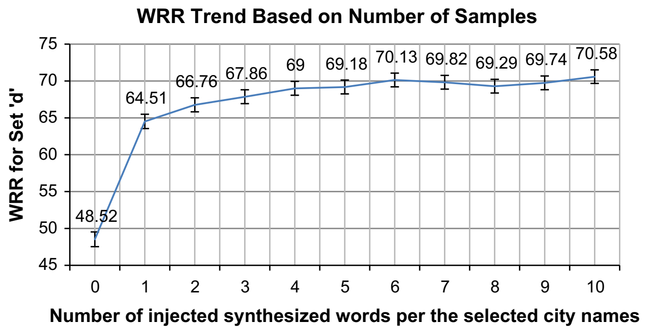


Fig. 18. Recognition result and significance for injecting different number of 'SE' synthesized samples in the original training data.

The WRR trend with number of injected images for each city name is graphically shown in Fig. 18.

We summarize the results of text recognition experiments in Table 7. It can be clearly seen from the table that adding synthesized training data to the baseline training set significantly improves the results for both, the EG and the SE techniques, although SE leads to a better improvement. From the table, we can see that the EG technique reports a WRR of 63.67%, an improvement of 9.93% whereas the SE technique reports a WRR of 70.13%, an improvement of 16.39% over the baseline system. The trend is the same for Set *'d'* and Set *'e'*.

In order to make sure that the improvements are indeed due to the concatenation of character-shapes and not solely due to simple re-training of the same data, we conducted two more set of experiments where we first double the baseline training data by adding a copy of the baseline images and then add a copy of data that is synthesized by generation, not concatenation, from the baseline. The results using the double number of training samples did not show any significant improvement over the baseline system. In the second set of additional experiments, we did some simple perturbations based synthesis on the original data by once thickening, and once thinning the strokes of the training text images. Thickening and thinning of strokes for synthesis was used by some researchers as reported in [3]. Thus, in these experiments, the training set consists of three sub sets; the original training data, original training image with strokes thickened, and the original data with strokes thinned. Using this configuration significantly improves the recognition results over the baseline system but still the improvements are significantly lower than those achieved by both of our concatenation approaches; thereby further corroborating the conclusions drawn on improvements due to synthesized data.

In context of Table 1, we report the following: we use a bottom-up approach for the synthesis of off-line Arabic handwriting samples. We reshuffle the letters of 2322 words into 6 versions of 721 other words and use these, along with the original baseline database, to train an HMM classifier. The WRR results improve recognition by a 16.39% and 17.99% for Sets *'d'* and *'e'*, respectively. These conclusions are summarized in Table 8.

Table 7
Word Recognition Rates (WRR) for text recognition task on IFN/ENIT database in percentage.

System	Set *'d'*				Set *'e'*			
	Top 1	Statistical significance	Top 5	Top 10	Top 1	Statistical significance	Top 5	Top 10
Baseline system	53.74	(±1.00)	64.17	67.74	48.52	(±1.06)	67.31	70.35
Doubled baseline system	53.82	(±1.00)	64.29	67.86	48.44	(±1.06)	67.30	70.29
Baseline system+thickening+thinning	58.02	(±1.00)	72.79	76.50	53.53	(±1.06)	69.96	73.80
Expanded by EG synthesis	63.67	(±0.97)	74.44	77.98	58.54	(±1.05)	77.65	80.67
Expanded by SE synthesis	**70.13**	**(±0.93)**	**81.19**	**84.19**	**66.51**	**(±1.01)**	**82.94**	**85.53**
Best Improvement	16.39	–	17.02	16.45	17.99	–	15.63	15.18

Table 8
Word Recognition Rates (WRR) for text recognition task on IFN/ENIT database.

Task	Original DB	Original results	Synthesized data	Synthesized data results	Classifier	Injection
• Bottom-up • Offline • Text recognition • Arabic	• 2322 words	Set *'d'* 53.7% Set *'e'* 48.52% WRR	6 versions of 721 IFN/ENIT city names	Set *'d'* 70.1% Set *'e'*66.51% WRR	HMM	Yes

5. Conclusion

Handwriting synthesis has gained increasing interest because of its applications, especially in the improvement of handwriting recognition systems. We proposed two concatenation-based synthesis approaches for Arabic: one based on character-shape glyphs with inherent character extensions and another with synthetic extensions. We synthesized six versions of 721 Tunisian town/village names taken from the IFN/ENIT database. We experimented by injecting synthesized handwriting data generated from a baseline training set to an HMM based Arabic handwriting recognition system. Significant improvement in recognition performance due to our synthesis system was observed.

In one set of experiments, our baseline system reports a word recognition rate (WRR) of 53.74%. The Extended-Glyphs (EG) technique reports a WRR of 63.67%, an improvement of 9.93%. The Synthetic Extension technique reports a WRR of 70.13%, an improvement of 16.39% over the baseline system and of 6.46% over the EG technique.

Acknowledgment

We would like to thank King Fahd University of Petroleum and Minerals and Qassim University for their support. This work is funded by the King Abdul Aziz City for Science and Technology, Grant GSP-18–112.

References

[1] I. Ahmad, G.A. Fink, and S.A. Mahmoud, Improvements in sub-character HMM model based Arabic text recognition, in: proceedings of the Fourteenth International Conference on Frontiers of Handwriting Recognition (ICFHR), Crete, Greece, 2014 (in press).

[2] M. Helmers, H. Bunke, Generation and use of synthetic training data in cursive handwriting recognition, in: F.J. Perales, A.J.C. Campilho, N.P. de la Blanca, A. Sanfeliu (Eds.), Pattern Recognition and Image Analysis, Springer, Berlin Heidelberg, 2003, pp. 336–345.

[3] T. Varga, H. Bunke, Perturbation models for generating synthetic training data in handwriting recognition, in: P.S. Marinai, P.H. Fujisawa (Eds.), Machine Learning in Document Analysis and Recognition, Springer, Berlin Heidelberg, 2008, pp. 333–360.

[4] H. Miyao and M. Maruyama, Virtual example synthesis based on PCA for off-line handwritten character recognition, in: Proceedings of the 7th International Conference on Document Analysis Systems, Berlin, Heidelberg, 2006, pp. 96–105.

[5] Z. Lin, L. Wan, C.-H. Hu, and J. Wang, Handwriting recognition training and synthesis, US7657094 B202-Feb-2010.

[6] R.M. Saabni, J.A. El-Sana, Comprehensive synthetic Arabic database for on/off-line script recognition research, Int. J. Doc. Anal. Recognit. (IJDAR) 16 (3) (2013) 285–294.

[7] S. Bayoudh, H. Mouchère, L. Miclet, E. Anquetil, Learning a classifier with very few examples: analogy based and knowledge based generation of new examples for character recognition, in: J.N. Kok, J. Koronacki, R.L. de Mantaras, S. Matwin, D. Mladenič, A. Skowron (Eds.), Machine Learning: ECML, Springer, Berlin Heidelberg, 2007, pp. 527–534in: J.N. Kok, J. Koronacki, R.L. de Mantaras, S. Matwin, D. Mladenič, A. Skowron (Eds.), Machine Learning: ECML, Springer, Berlin Heidelberg, 2007, pp. 527–534.

[8] A. Almaksour, E. Anquetil, R. Plamondon, and C. O'Reilly, Synthetic handwritten gesture generation using sigma-lognormal model for evolving handwriting classifiers, in: Proceedings of the 15th Biennial Conference of the International Graphonomics Society, 2011, pp. 98–101.

[9] A. Almaksour, E. Anquetil, S. Quiniou, and M. Cheriet, Evolving fuzzy classifiers: application to incremental learning of handwritten gesture recognition systems, in: Proceedings of the 20th International Conference on Pattern Recognition (ICPR), 2010, 4056–4059.

[10] H. Miyao, M. Maruyama, Y. Nakano, and T. Hananoi, Off-line handwritten character recognition by SVM based on the virtual examples synthesized from on-line characters, in: Proceedings of the Eighth International Conference on Document Analysis and Recognition, vol. 1, 2005, pp. 494–498.

[11] R. Plamondon, C. O'Reilly, J. Galbally, A. Almaksour, É. Anquetil, Recent developments in the study of rapid human movements with the kinematic theory: Applications to handwriting and signature synthesis, Pattern Recognit. Lett. 35 (2014) 225–235.

[12] A. Almaksour, E. Anquetil, S. Quiniou, and M. Cheriet, Personalizable pen-based interface using lifelong learning, in: Proceedings of the International Conference on Frontiers in Handwriting Recognition (ICFHR), 2010, pp. 188–193.

[13] S. Cho, J.-H. Kim, and D. Kim, Apparatus and method of generating personal fonts, US20090189905 A130-Jul-2009.

[14] L. Ballard, D. Lopresti, F. Monrose, Forgery quality and its implications for behavioral biometric security, IEEE Trans. Syst., Man Cybern. B Cybern. 37 (5) (2007) 1107–1118.

[15] L. Ballard, F. Monrose, and D. Lopresti, Biometric authentication revisited: understanding the impact of wolves in sheep's clothing, in: Proceedings of the 15th Conference on USENIX Security Symposium, vol. 15, Article 3, Berkeley, CA, USA, 2006.

[16] T. Varga and H. Bunke, Generation of synthetic training data for an HMM-based handwriting recognition system, in: Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR), 2003, pp. 618–622.

[17] T. Varga and H. Bunke, Off-line handwritten textline recognition using a mixture of natural and synthetic training data, in: Proceedings of the 17th International Conference on Pattern Recognition (ICPR), 2, 2004, pp. 545–549.

[18] T.M. Ha, H. Bunke, Off-line handwritten numeral string recognition, Pattern Recognit. 31 (1997) 257–272.

[19] J. Cano, J.-C. Perez-Cortes, J. Arlandis, and R. Llobet, Training set expansion in handwritten character recognition, in: Structural, Syntactic and Statistical Pattern Recognition, LNCS 2396, 2002, pp. 548–556.

[20] VISTAWIDE: world languages and cultures, the Arabic language, Online, available: ⟨http://www.vistawide.com/arabic/arabic.htm⟩ (Accessed 04.01.14).

[21] Arabic alphabet, Encyclopedia Britannica (23.02.14).

[22] J.M. Hollerbach, An oscillation theory of handwriting, Biol. Cybern. 39 (2) (1981) 139–156.

[23] G. Gangadhar, D. Joseph, V.S. Chakravarthy, An oscillatory neuromotor model of handwriting generation, Int. J. Doc. Anal. Recognit. (IJDAR) 10 (2) (2007) 69–84.

[24] J. Galbally, R. Plamondon, J. Fierrez, J. Ortega-Garcia, Synthetic on-line signature generation. Part I: methodology and algorithms, Pattern Recognit. 45 (7) (2012) 2610–2621.

[25] M. Djioua, R. Plamondon, Studying the variability of handwriting patterns using the kinematic theory, Hum. Mov. Sci. 28 (5) (2009) 588–601.

[26] J. Galbally, J. Fierrez, J. Ortega-Garcia, R. Plamondon, Synthetic on-line signature generation. Part II: experimental validation, Pattern Recognit. 45 (7) (2012) 2622–2632.

[27] R. Plamondon, M. Djioua, A multi-level representation paradigm for handwriting stroke generation, Hum. Mov. Sci. 25 (4–5) (2006) 586–607.

[28] M. Djioua and R. Plamondon, An interactive system for the automatic generation of huge handwriting databases from a few specimens, in: Proceedings of the 19th International Conference on Pattern Recognition (ICPR), 2008, pp. 1–4.

[29] R. Plamondon, C. Feng, A. Woch, A kinematic theory of rapid human movement. Part IV: a formal mathematical proof and new insights, Biol. Cybern. 89 (2) (2003) 126–138.

[30] J. Wang, C. Wu, Y.-Q. Xu, H.-Y. Shum, Combining shape and physical models for online cursive handwriting synthesis, Int. J. Doc. Anal. Recognit. (IJDAR) 7 (4) (2005) 219–227.

[31] G. Terejanu, On the Handwritten CAPTCHA. Available: ⟨http://www.academia.edu/677243/On_the_Handwritten_CAPTCHA⟩ (Accessed 30.03.14).

[32] Y. Zheng and D. Doermann, Handwriting matching and its application to handwriting synthesis, in: Proceedings of the Eight International Conference on Document Analysis and Recognition (ICDAR), 2005, pp. 861–865.

[33] T. Wakahara, Y. Kimura, A. Tomono, Affine-invariant recognition of gray-scale characters using global affine transformation correlation, IEEE Trans. Pattern Anal. Mach. Intell. 23 (4) (2001) 384–395.

[34] D. Keysers, C. Gollan, and H. Ney, Local context in non-linear deformation models for handwritten character recognition, in: Proceedings of the 17th International Conference on Pattern Recognition (ICPR), 4, 2004, pp. 511–514.

[35] P. Viswanath, M.N. Murty, S. Bhatnagar, Partition based pattern synthesis technique with efficient algorithms for nearest neighbor classification, Pattern Recognit. Lett. 27 (14) (2006) 1714–1724.

[36] P. Viswanath, N. Murty, S. Bhatnagar, Overlap pattern synthesis with an efficient nearest neighbor classifier, Pattern Recognit. 38 (8) (2005) 1187–1195.

[37] H. Choi, S.-J. Cho, and J.H. Kim, Generation of handwritten characters with Bayesian network based on-line handwriting recognizers, in: Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR), Washington, DC, USA, 2003, pp. 995–999.

[38] O. Stettiner and D. Chazan, A statistical parametric model for recognition and synthesis of handwriting, in: Proceedings of the 12th International Conference on Pattern Recognition (ICPR), 2, 1994, pp. 34–38.

[39] M. Mori, A. Suzuki, A. Shio, and S. Ohtsuka, Generating new samples from handwritten numerals based on point correspondence Online at URL: ⟨http://www.temoa.info/node/186174⟩, 2004.

[40] J. Dolinsky and H. Takagi, Synthesizing handwritten characters using naturalness learning, in: Proceedings of the IEEE International Conference on Computational Cybernetics (ICCC), 2007, pp. 101–106.

[41] W. Cheng and D. Lopresti, Parameter calibration for synthesizing realistic-looking variability in offline handwriting, In: Proceedings of the Document Recognition and Retrieval XVIII, part of the IS&T-SPIE Electronic Imaging Symposium, San Jose, CA, USA, vol. 7874, January 24–29, 2011, pp. 1–10.

[42] J. Chen, W. Cheng, and D. Lopresti, Using perturbed handwriting to support writer identification in the presence of severe data constraints, In: Proceedings of the SPIE 7874, Document Recognition and Retrieval XVIII, 78740G, http://dx.doi.org/10.1117/12.876497, January 24, 2011.

[43] I. Guyon, Handwriting synthesis from handwritten glyphs, in Proceedings of the Fifth International Workshop on Frontiers of Handwriting Recognition, 1996, pp. 309–312.

[44] C.V. Jawahar and A. Balasubramanian, Synthesis of online handwriting in indian languages, in: Proceedings of the 10th International Workshop on Frontiers in Handwriting Recognition, 2006, pp. 108–113.
[45] C.V. Jawahar, A. Balasubramanian, M. Meshesha, A.M. Namboodiri, Retrieval of online handwriting by synthesis and matching, Pattern Recognit. 42 (7) (2009) 1445–1457.
[46] Y. Elarian, Husni Al-Muhtaseb, and Lahouari Ghouti, Arabic Handwriting Synthesis, in: International Workshop on Frontiers in Arabic Handwriting Recognition, Istanbul, 2010 .
[47] R. Saabni and J. El-Sana, Efficient generation of comprehensive database for online arabic script recognition, in: Proceedings of the 10th International Conference on Document Analysis and Recognition (ICDAR '09), 2009, pp. 1231–1235.
[48] J. Wang, C. Wu, Y.-Q. Xu, H.-Y. Shum, and L. Ji, Learning-based cursive handwriting synthesis, in: Proceedings of the 8th International Workshop on Frontiers in Handwriting Recognition, 2002, pp. 157–162.
[49] Z. Lin, L. Wan, Style-preserving English handwriting synthesis, Pattern Recognit. 40 (7) (2007) 2097–2109 (Jul.).
[50] P. Rao, Shape vectors: an efficient parametric representation for the synthesis and recognition of hand script characters, Sadhana 18 (1) (1993) 1–15.
[51] Y.-Q. Xu, H.-Y. Shum, J. Wang, and C. Wu, Learning-based system and process for synthesizing cursive handwriting, US7227993 B205-Jun-2007.
[52] L. Dinges, A. Al-Hamadi, and M. Elzobi, An approach for arabic handwriting synthesis based on active shape models, in: Proceedings of the 12th International Conference on Document Analysis and Recognition (ICDAR), 2013, pp. 1260–1264.
[53] L. Dinges, M. Elzobi, A. Al-Hamadi, Z.A. Aghbari, Synthizing handwritten arabic text using active shape models, in: R.S. Choraś (Ed.), Image Processing and Communications Challenges 3, Springer, Berlin Heidelberg, 2011, pp. 401–408.
[54] Y. Elarian and F. Idris, A lexicon of connected components for arabic optical text recognition, in: Proceedings of the 1st International Workshop on Frontiers in Arabic Handwriting Recognition, Istanbul, 2010.
[55] S.J. Young, G. Evermann, M.J.F. Gales, D. Kershaw, G. Moore, J.J. Odell, D.G. Ollason, D. Povey, V. Valtchev, and P.C. Woodland, The HTK book version 3.4, 2006.
[56] M. Pechwitz, S.S. Maddouri, V. Märgner, N. Ellouze, and H. Amiri, IFN/ENIT—database of handwritten Arabic words, in: Proceedings of CIFED, 2002, pp. 129–136.
[57] U.-V. Marti and H. Bunke, Handwritten sentence recognition, in: Proceedings of the 15th International Conference on Pattern Recognition (ICPR), 2000, pp. 463–466.
[58] R. Saabni and J. El-Sana, Hierarchical on-line Arabic handwriting recognition, in: Proceedings of the 10th International Conference on Document Analysis and Recognition (ICDAR), 2009, pp. 867–871.
[59] Y. Al-Ohali, M. Cheriet, C. Suen, Databases for recognition of handwritten Arabic cheques, Pattern Recognit. 36 (1) (2003) 111–121.

Dr. Yousef Elarian obtained his Ph.D. from the College of Computer Science and Engineering at the King Fahd University of Petroleum and Minerals, Saudi Arabia in 2014. He received his M.Sc. in Computer Engineering from the Jordanian University of Science and Technology, Jordan, in 2006. He received his B.Sc. in Computer Engineering from the Islamic University of Gaza in 2003. His research interest is in the field of Arabic-related systems, in particular, Arabic text recognition. Dr. Yousef has several conference and journal papers in the area of Arabic computing.

Mr. Irfan Ahmad is a lecturer at the Information and Computer Science Department, King Fahd University of Petroleum and Minerals (KFUPM), Saudi Arabia. He received his master's degree (MSc) in computer science from KFUPM in 2008. His research interests include statistical pattern recognition, Arabic document analysis and recognition (including printed and handwritten Arabic text recognition), image analysis, and software engineering. Mr. Ahmad has published several papers in peer-reviewed journals and conferences in addition to a book chapter and a patent.

Dr. Sameh Awaida is an Assistant Professor in the Computer Engineering Department in Qassim University, Saudi Arabia. He obtained his Ph.D. in Computer Science and Engineering (CSE) from KFUPM in 2011. He got his B.Sc. and M.Sc. degrees in Electrical Engineering from the University of Hartford (CT, USA) in 2003 and 2005, respectively. Previously, he worked as a Lecturer in KFUPM and PSUT. His research interests include pattern recognition, image processing and embedded systems. Along with two patents, he has published more than 10 research papers in international journals and conferences.

Dr. Wasfi G. Al-Khatib is an assistant professor at the King Fahd University of Petroleum and Minerals, Saudi Arabia. He received his BS degree in computer science from the Kuwait University in 1990, and his MS degree in computer science and PhD in Electrical and Computer Engineering from Purdue University in 1995 and 2001, respectively. He worked at Wright State University in Dayton, Ohio as an assistant professor from 2001–2002. His research interests include Arabic computing, multimedia computing, content-based retrieval, and artificial intelligence.

Dr. Abdelmalek Zidouri is a Senior IEEE member and an Associate Professor in the Department of Electrical Engineering at the King Fahd University of Petroleum and Minerals, Dhahran Saudi Arabia. He holds a Master of Science in Control Engineering from the Bradford University, UK, in 1984 and a Doctor of Engineering in Applied Electronics from Tokyo Institute of Technology Japan, in 1995. His research interests are in the field of Signal processing and Pattern recognition. In particular, character recognition and document image analysis. Dr. Zidouri has published many refereed journal and conference papers. He has supervised many projects, theses, and dissertations.

He was the head of the Digital Signal Processing Group at the EE Department. He is now the seminar coordinator of the Department. He is a member of the Engineering Education Society, Signal Processing Society and Communications Society.