Appl Intell (2011) 35:445–456
DOI 10.1007/s10489-010-0235-2

# Recognition of Arabic (Indian) bank check digits using log-gabor filters

**Sabri A. Mahmoud · Wasfi G. Al-Khatib**

Published online: 29 May 2010


**Abstract** In this paper we present a technique for the automatic recognition of Arabic (Indian) bank check digits based on features extracted by using the Log Gabor filters. The digits are classified by using the K-Nearest Neighbor (K-NN), Hidden Markov Models (HMM) and Support Vector Machines (SVM) classifiers. An extensive experimentation is conducted on the CENPARMI data, a database consisting of 7390 samples of Arabic (Indian) digits for training and 3035 samples for testing extracted from real bank checks. The data is normalized to a height of 64 pixels, maintaining the aspect ratio. Log Gabor filters with several scales and orientations are used. In addition, the filtered images are segmented into different region sizes for feature extraction. Recognition rates of 98.95%, 98.75%, 98.62%, 97.21% and 94.43% are achieved with SVM, 1-NN, 3-NN, HMM and NM classifiers, respectively. These results significantly outperform published work using the same database. The misclassified digits are evaluated subjectively and results indicate that human subjects misclassified 1/3 of these digits. The experimental results, including the subjective evaluation of misclassified digits, indicate the effectiveness of the selected Log Gabor filters parameters, the implemented image segmentation technique, and extracted features for practical recognition of Arabic (Indian) digits.

**Keywords** Arabic (Indian) digits · Recognition of Arabic bank check digits · Feature extraction · Log-Gabor filters · Hidden Markov Models · Support Vector Machines · Nearest Neighbor

S.A. Mahmoud (✉) · W.G. Al-Khatib
Information and Computer Science, King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia
e-mail: smasaad@kfupm.edu.sa

W.G. Al-Khatib
e-mail: wasfi@kfupm.edu.sa

## 1 Introduction

Automatic recognition of Arabic (Indian) digits has received renewed attention in recent years. Many applications benefit from advances in this area, including banking applications for check digits recognition, handwritten forms processing and postal zip code reading. Although Arabic text is cursive, Arabic (Indian) numerals are not cursive, which simplifies their processing as there is no need to segment the digits as is the case with Arabic text.

Sadri et al. [1] developed a technique using four views based on the structural features of the digits. It transforms the Arabic/Persian digits into four one-dimensional features based on top-, bottom-, left-, and right-views. These views represented the number of white pixels from the side of the view to the boundary of the digit. Support Vector Machines (SVM) and Multilayer Perceptron (MLP) neural networks were used for classification. They used the Arabic check database of CENPARMI [2]. Digits were normalized to 64 × 64 pixels before the feature extraction stage. Recognition rates of 94.14% and 91.25% were reported using SVM and MLP, respectively. Bernoulli mixture models [3] were used to model binary representations of Arabic (Indian) digits using maximum likelihood estimation for digit classification. The digits were preprocessed by pasting each digit onto a white square background whose center was aligned with the digit center of mass. Then each digit was sub-sampled into a smaller square of pixels from which the binary vector was extracted. Different sizes of sub-sampled squares were tested by using the Bernoulli mixture classifier. A sub-sampling value of 20 was considered appropriate

for the task. The paper presented a graph with an error rate of ≈2.5% when using the CENPARMI database. Mahmoud in [4] used spatial Gabor filter-based features with 8 orientations and 3 scales using K-nearest neighbor (K-NN) and nearest mean (NM) classifiers. To our knowledge, these are the only publications for Arabic digits using CENPARMI Bank check database. In the present work, we use Log Gabor based features with different number of orientations and scales. Moreover, we use HMM and SVM in addition to K-NN and NM classifiers. In addition, the filtered images are segmented to different region sizes in extracting the features. The different sizes were tested and best performing ones are identified.

Recently many researchers addressed the recognition of Arabic (Indian) and Persian numerals [5–7]. Researchers in these publications used their own data. In [6], angle, ring, horizontal, and vertical span features were used with a left-to-right Hidden Markov Model (HMM). Salah et al. [8] developed a serial model for visual digit classification based on the primitive selective attention mechanism. The technique was based on parallel scanning of a down-sampled image to find interesting locations through a saliency map, and by extracting key features at high resolution. The authors in [5] mapped a digit image to a 12-segment pattern. The ratio of the black pixels in each segment to the black pixels of the digit was taken as the feature for that segment. Two cascaded Multi-Layer Perceptron neural networks were used, the first to identify the control points of the digit, and the second to recognize the digit. Those authors reported 97.6% recognition rate for their own data. Mozaffari et al. [7] used structural decomposition of the Farsi/Arabic digits where the skeleton of the digit was extracted and decomposed into primitives. Terminal and intersection points were the commonly used features in that approach. Changes in the average and variance of the $X$ and $Y$ coordinates of each primitive were used to form the feature vector. To increase the accuracy of their technique, the algorithm was applied to each quadrant of the digit, resulting in a feature vector of 32 features (i.e. 8 features per quadrant). A recognition rate of 94.44% was reported.

The present paper is organized as follows. Section 2 addresses log Gabor filters. Section 3 presents a summarized theory of Support Vector Machines and Hidden Markov Models. Experimental results are reported in Sect. 4. The conclusions are presented in Sect. 5.

## 2 Log-Gabor filters

The two-dimensional Gabor function can be viewed as a complex sinusoidal plane wave modulated by a low-pass Gaussian function in the space domain. The even and odd Gabor filters in the 2-dimensional spatial domain can be formulated as:

$$g_e(x, y; \lambda, \theta) = e^{-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2}\right)} \cos\left(2\pi \frac{x}{\lambda}\right) \tag{1}$$

$$g_o(x, y; \lambda, \theta) = e^{-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2}\right)} \sin\left(2\pi \frac{x}{\lambda}\right) \tag{2}$$

where $\lambda$ is the Gabor filter wavelength in pixels, $\theta$ is the angle of filter in degrees (zero angle gives a filter that responds to vertical features), $k_x$ and $k_y$ are scale factors relative to the wavelength of the filter. The bandwidth of the filter is controlled in the $x$-direction by $k_x$, whereas its orientation selectivity is controlled across the filter by $k_y$ ($\sigma_x = \lambda k_x$ and $\sigma_y = \lambda k_y$).

The response of the filter in (1) to an image $i(x, y)$ can be calculated with the convolution:

$$G_e(x, y; \lambda, \theta) = g_e(x, y; \lambda, \theta) * i(x, y) \tag{3}$$

$$G_o(x, y; \lambda, \theta) = g_o(x, y; \lambda, \theta) * i(x, y) \tag{4}$$

$$G_a(x, y; \lambda, \theta) = \sqrt{G_e^2(x, y; \lambda, \theta) + G_o^2(x, y; \lambda, \theta)} \tag{5}$$

where $G_e$, $G_o$, and $G_a$ are the even, odd, and amplitude responses of the even and odd filters. The filter orientations are computed from

$$\theta_k = \frac{2\pi k}{n}, \quad k = \{0, 1, \ldots, n-1\} \tag{6}$$

where $n$ is the number of orientations used. Some researchers use different forms of the Gabor filter than (1) as in [9–11]. In order to speed up the computation, our filtering was implemented in the frequency domain. Our process was implemented as follows:

$$\textit{Filtered Image} = \text{FFT}^{-1}[\text{FFT}(\textit{Image}) \times \text{FFT}(\textit{Filter})] \tag{7}$$

where FFT and $\text{FFT}^{-1}$ are respectively the fast Fourier transform and the inverse fast Fourier transform.

The FFT of the filter of (7) is defined in the frequency domain for further reduction of the computation time. Filter banks of different orientations and scales are tested. The highest recognition rates are achieved with 6 orientations (0, 30, 60, 90, 120 and 150) and 4 scales (wavelengths of 3, 6, 12 and 24). The filtered image is segmented into smaller segments. We experimented with different segment sizes. The number of segments that resulted in the highest recognition rates (i.e., $3 \times 3$ segments) is used. To extract the features, the filtered image is segmented into $n \times m$ segments ($n$ is the number of horizontal slices and $m$ is the number of vertical slices), where each segment is of size $\frac{\textit{Image Width}}{n} \times \frac{\textit{Image Length}}{m}$. The mean and variance of each segment are taken as the features of the segment. This is repeated for all filtered images at different scales, orientations

and image segments of the digits. This results in a feature vector of $6 \times 4 \times 3 \times 3 \times 2 = 432$ features for 6 orientations, 4 scales, 3 horizontal segments, 3 vertical segments, and using the mean and variance.

## 3 Classification

In this work, K-NN, NM, SVM and HMM classifiers are used. K-NN and NM classifiers are simple classifiers that are used to test the suitability of the features, in addition to selecting the filter parameters and the filtered images segmentation sizes that result in the highest recognition rates. HMM and SVM classifiers, which are more powerful classifiers, are used to obtain the highest recognition rates using the selected filter parameters and segment sizes. HMM has another advantage as it can be used with Arabic text recognition, which is cursive, without prior segmentation. In the following sections, we will present a summary of the theory of these classifiers.

### 3.1 Support Vector Machines (SVM)

Support Vector Machines (SVM) are modern learning systems that deliver state-of-the-art performance in real world Pattern Recognition and in data mining applications such as text categorization, hand-written character recognition, image classification and bioinformatics. Camastra [12] used SVM for cursive character recognition. Mozaffari et al. used SVM and direction and curvature features of the skeleton images to recognize Arabic/Persian zip code numerals [7]. Soltanzadeh used the profile of the digit images along with SVM to recognize Arabic/Persian digits [13], whereas Mozaffari et al. used the SVM classifier to perform feature comparison between fractal codes and wavelet transforms [14]. Mowlaei et al. [15] used wavelet transforms with SVM to recognize Arabic/Persian digits extracted from postal addresses.

The SVM was developed by Vapnik and other researchers [16–20]. Within a short period of time SVM classifiers became competitive with the best available systems for pattern recognition applications [21, 22]. Here we briefly describe the basic theory behind SVM for pattern recognition, especially for the two-class classification problem, and we refer readers to [17, 23] for a full description of the technique.

For a two-class classification problem, assume that we have a set of samples, i.e. a series of input vectors: $x_i \in R^d$ $(i = 1, 2, \ldots, N)$ with corresponding labels, $y_i \in \{+1, -1\}$ $(i = 1, 2, \ldots, N)$. Here, $+1$ and $-1$ indicate the two classes.

The goal is to construct a binary classifier from the available samples which has a small probability of misclassifying a future sample. SVM maps the input vectors $x_i \in R^d$ into a high dimensional feature space $\Phi(x) \in H$ and it constructs an Optimal Separating Hyper-plane (OSH). OSH maximizes the margin, the distance between the hyper plane and the nearest data points of each class in the space $H$. Different mappings construct different SVMs. The mapping $\Phi(x)$ is performed by a kernel function $K(x_i, x_j)$ which defines an inner product in the space $H$. The decision function implemented by the SVM can be written as:

$$f(x) = \operatorname{sgn}\left(\sum_{i=1}^{N} y_i \alpha_i \cdot K(x, x_i) + b\right) \tag{8}$$

where the coefficients $\alpha_i$ are obtained by solving the following convex Quadratic Programming (QP) problem:

Maximize

$$\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j \cdot y_i y_j \cdot K(x_i, x_j),$$
$$0 \le \alpha_i \le C \tag{9}$$

subject to

$$\sum_{i=1}^{N} \alpha_i y_i = 0, \quad i = 1, 2, \ldots, N \tag{10}$$

where $C$ is a regularization parameter which controls the trade off between margin and misclassification error. These $x_j$ are called Support Vectors only if the corresponding $\alpha_i > 0$.

Several typical kernel functions are:

$$\textit{Polynomial}: \quad K(x_i, x_j) = (x_i \cdot x_j + 1)^d \tag{11}$$

*Radial Basis Function* (*RBF*):

$$K(x_i, x_j) = \exp\left(-\gamma \|x_i - x_j\|^d\right) \tag{12}$$

$$\textit{Linear}: \quad K(x_i, x_j) = x_i^T x_j \tag{13}$$

$$\textit{Sigmoid}: \quad K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r) \tag{14}$$

Here $\gamma$, $r$ and $d$ are kernel parameters.

SVMs differ radically from comparable approaches such as neural networks. SVM training always finds a global minimum.

### 3.2 Hidden Markov Models (HMM)

Several research papers have been published using HMM for text recognition [24–29]. In order to use HMMs several researchers computed the feature vectors as a function of an independent variable. This simulated the use of HMM in speech recognition where sliding frames/windows were used. The same technique was used in off-line text recognition where the independent variable was in the direction of the line length [25, 26].

A hidden Markov model assumes that the sequence of observations representing each check digit is generated by a Markov model. A Markov model is a finite-state machine that changes its state at each time unit. With each change of state, a vector is generated.

The probability of generating an observation vector $O$ by model $\lambda$ through the state sequence $S$ is the product of the probabilities of the outputs and the probabilities of the transitions.

$$P(O, Q|\lambda) = \pi_1 b_1(o_1) a_{12} b_2(o_2) a_{23} b_3(o_3) \cdots$$

where $O = \langle o_1, o_2, o_3, \ldots, \rangle$ is a sequence of observations; $Q$, is the state sequence; $\lambda = (A, B, \pi)$; $\pi_i$, initial state transition; $a_{ij}$ the transition probability from state $i$ to state $j$; and $b_i$ is the output probability at state $i$.

As the state sequence is unknown, the probability is computed by summing over all possible state sequences. Since this is time-consuming, it is approximated by the following equation:

$$P(O|\lambda) = \max_{Q} \prod_{i=1}^{T} \pi_{q_{i-1}} a_{q_{i-1}q_i} b_{q_i}(O_i)$$

where $Q = \langle q_1, q_2, q_3, \ldots \rangle$ is the state sequence of the model.

This equation is usually computed through recursion with the assumption that the parameters $a_{ij}$ and $b_i$ are known for each model $\lambda_i$. The model parameters are estimated in the training phase by using the Baum-Welch algorithm. The sequence of states $S$ that gives the highest probability is determined by the Viterbi algorithm.

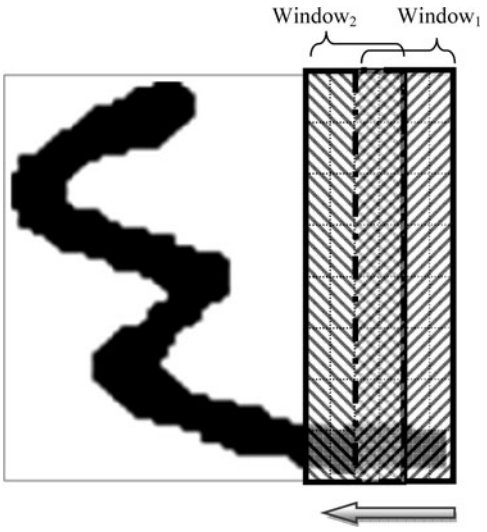


**Fig. 1** A right-to-left sliding window of 4 pixels of width is used to extract features from the digit image

The sliding window technique is applied to the multi-scale, multi-orientation filtered images. For each filtered image, a sliding window is used to extract the features. Different sizes and horizontal overlap of the sliding window are used (window width of 4 pixels and overlap of 1 and 2 pixels; window width of 3 pixels and overlap of 1 and 2 pixels). In addition, the window is divided into a number of vertical overlapping and non-overlapping segments. We tested 3, 4, 6 and 8 segments with a vertical overlapping of 1/2 and 1/4 of the window height. The highest recognition rates are achieved with a window of 4 pixels wide, a horizontal overlap of 2 pixels and 8 vertical non-overlapping segments. Figure 1 shows this case on Digit 4.

In this paper, we use a left-to-right HMM for our Arabic (Indian) handwritten check digit recognition. Figure 2 shows a 5-state HMM. This is in line with several research works using HMM [25, 26]. Our model allows relatively large variations in the horizontal position of the Arabic (Indian) digit. The sequence of state transition in the training and testing of our model is related to each digit feature observation. In this work, we experimented with different numbers of states, and we selected the best performing one. Although each digit model could have a different number of states, we decided to use the same number of states for all digits as was done in [25, 26].

Each Arabic (Indian) digit is represented by a 432-dimensional feature vector, as noted in Sect. 2. Each digit requires a number of observations to train and test the HMM. Each digit is, therefore, represented by 27 observations of 16 features each.

## 4 Experimental results

The CENPARMI Arabic checks database, which is extracted from real Arabic bank checks, is used in this work [2]. The database includes Arabic (Indian) digits, and legal and courtesy amounts. In our experiments, the Arabic (Indian) check digits are used. The database consists of 7390 isolated digits for training and 3035 digits for testing. Samples of digits 0 to 9 are shown in Fig. 3.

The distribution of the number of samples of each digit in the training and testing sets is shown in Table 1. It is clear

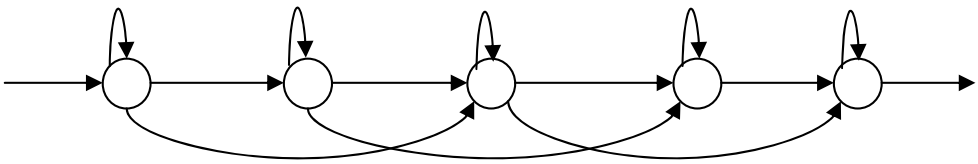

**Fig. 2** A 5-state Hidden Markov Model (HMM)

**Table 1** The number of samples of each digit in the training and testing sets

| Digit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Total |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Training set | 3793 | 782 | 545 | 362 | 307 | 649 | 279 | 233 | 246 | 194 | 7390 |
| Testing set | 1574 | 304 | 225 | 144 | 133 | 263 | 111 | 109 | 98 | 74 | 3035 |
| Total | 5367 | 1087 | 772 | 509 | 444 | 917 | 396 | 349 | 352 | 277 | 10425 |

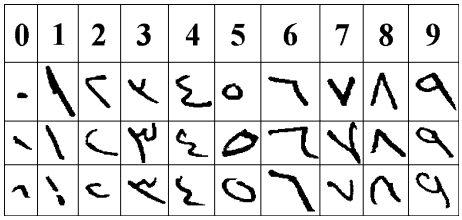


**Fig. 3** Samples of CENPARMI Arabic (Indian) bank check digits [2]

that the number of samples of each digit in the database is not equal. For example, the number of samples of digit zero is nearly equal to the total number of samples of all the other digits.

In our experimentation, the training and testing data are used with and without preprocessing. In the case of preprocessing, data is normalized before the feature extraction stage. The check digits are normalized to a height of 64 pixels while maintaining the aspect ratio. This helps in making digits zero and one of different normalized shape, whereas normalizing both the width and height may result in making some samples of digit '0' and digit '1' look alike. In our experiments, 7390 samples were used in training, and the remaining 3035 samples were used in testing, as specified by CENPARMI and for comparison with related published work. Section 4.1 presents the results of using the K-Nearest Neighbor and Nearest Mean classifiers. Section 4.2 shows the results of using Hidden Markov Models. Section 4.3 describes the results of using Support Vector Machines. Finally, we analyze the results of the different classifiers, and we compare them with the results of published work in Sect. 4.4.

### 4.1 K-NN and NM classifiers

K-Nearest Neighbor (K-NN) and Nearest Mean (NM) are simple classifiers that are used to measure the effectiveness of the extracted features. Both, 1-NN and 3-NN classifiers are used. With respect to NM, the mean of all the features of the training samples for each digit is used as the model of the digit.

In order to extract the feature vector of the digit image, the image is filtered by a number of filters with different numbers of scales and orientations. For each scale, 6 filtered images are produced. The filtered images are segmented. The mean $\mu_{\langle o,w,s\rangle}$ and standard deviation $\sigma_{\langle o,w,s\rangle}$ are computed for each filtered image segment; where $o$ refers to the orientation, $w$ refers to the wavelength (scale), and $s$ represents the filtered digit segment number. The numbers are in sequence top-down and right-left. The feature vectors of all the used scales and orientations are concatenated to form the feature vector of the Arabic check digit, $V = [\mu_{\langle 1,1,1\rangle}, \sigma_{\langle 1,1,1\rangle}, \ldots, \mu_{\langle 4,6,9\rangle}, \sigma_{\langle 4,6,9\rangle}]$.

We use different numbers of horizontal and vertical segments per image in our experiments. The highest recognition rates are achieved with $3 \times 3$ segments. We also performed an experiment on digit images without normalization. In this experiment, the 1-NN, 3-NN and NM classifiers achieve average recognition rates of 99.04%, 98.68% and 44.22%, respectively. The noticeably low performance of the NM classifier is largely attributed to a recognition rate of 0% for digit 0. This was a result of the high variations in the samples of digit '0' resulting in the model being far from representing the digit. This result was also reported when using the NM classifier with the spatial Gabor filters [4]. The rest of the experiments were conducted with normalized data. The data is normalized to a height of 64 pixels while maintaining the aspect ratio of the samples. These experiments are carried out by using different numbers of scales and orientations. Scales of 3 and 4 with minimum wavelengths of 1, 1.5, 2, 3 and 4 pixels with multiplication factors of 1.3, 1.6, 2 and 2.1 are used. A number of orientations ranging from 4, 6, 8, etc., were used in (3) to produce the orientation angles. A number of horizontal and vertical segments of the check digit image were used, viz., $2 \times 2$, $2 \times 3$, $3 \times 2$, $3 \times 3$, $4 \times 4$, $3 \times 4$ and $4 \times 3$. The highest recognition rates were achieved with 6 orientations and 4 scales. Those parameters that achieved the highest recognition rates were used in 9 experiments. These experiments will be referred to as $e_1, e_2, \ldots, e_9$. The rest of the experiments are not reported as they did not produce high recognition rates. Figures 4, 5 and 6 show the recognition rates achieved by using the 1-NN, 3-NN and NM classifiers, respectively, for all experiments that achieved the highest recognition rate for at least one digit. The experiments are shown on the left legend of the figure. Table 2 shows the parameters used for each of these experiments.

Figure 4 shows the recognition rates of digits 0 to 9 achieved by using 1-NN and different numbers of scales, orientations and image segments. It is clear from the figure that digits 4 and 9 have the lowest recognition rates. Some configurations give better recognition rates than others. The highest average recognition rate, viz. 98.75% is shown by

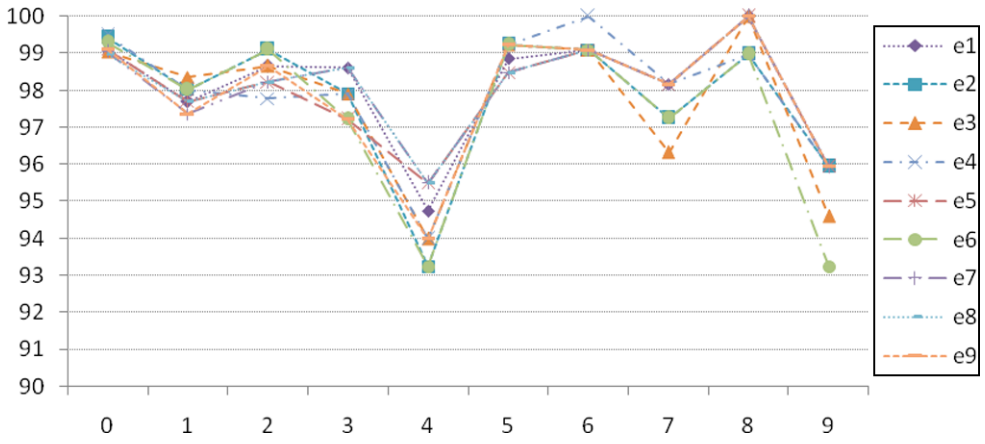


**Fig. 4** The recognition rates of digits 0 to 9 with 1-NN and different orientations, scales and segments

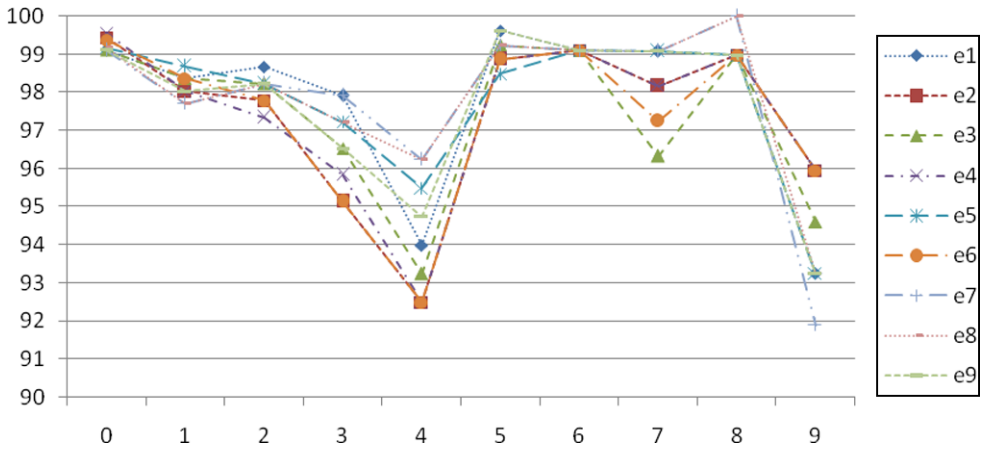


**Fig. 5** The recognition rates of digits 0 to 9 with 3-NN and different orientations, scales and segments

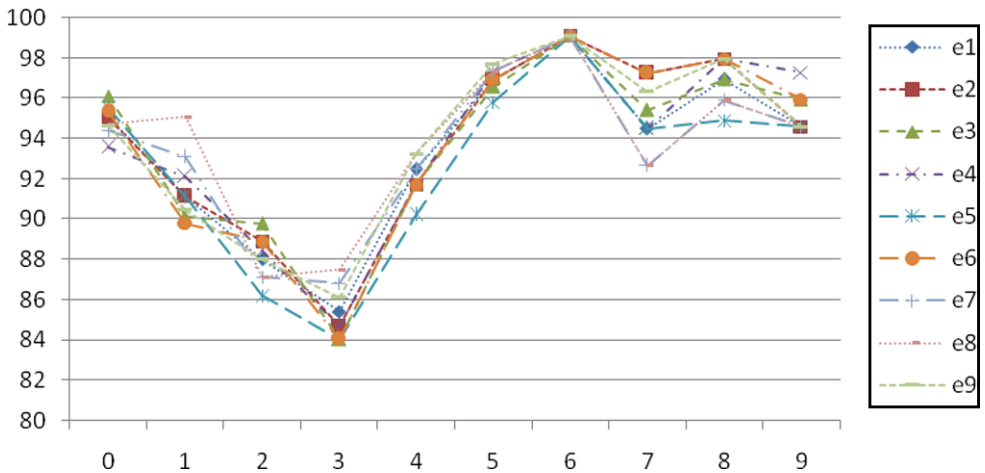


**Fig. 6** The recognition rates of digits 0 to 9 with NM and different orientations, scales and segments

the curve of Experiment $e_4$ and it is achieved by using 4 scales (wavelengths of 1, 2, 4, and 8), 6 orientations (0°, 30°, 60°, 90°, 120° and 150°) and $3 \times 3$ segments.

Figure 5 shows recognition rates of digits 0 to 9 achieved by using 3-NN and different numbers of scales, orientations and image segments, as in Fig. 4. Similar to the 1-NN classifier, Digits 4 and 9 have the lowest recognition rates. The highest average recognition rate, viz. 98.62% is shown by the curve of Experiment $e_1$ and it is achieved by using 4 scales (wavelengths of 3, 6, 12, and 24), 6 orientations (0°, 30°, 60°, 90°, 120° and 150°) and $3 \times 3$ segments.

Figure 6 shows the recognition rates of Digits 0 to 9 achieved by using NM classifier with the parameters of the experiments of Figs. 4 and 5. In this case, Digit 3 has the lowest recognition rate. In general, NM classifier has lower recognition rates than 1-NN and 3-NN. The highest aver-

**Table 2** Parameter values used in (5) for Experiments 1 to 9

| Parameter | $e_1$ | $e_2$ | $e_3$ | $e_4$ | $e_5$ | $e_6$ | $e_7$ | $e_8$ | $e_9$ |
|---|---|---|---|---|---|---|---|---|---|
| Min. Wavelength ($\lambda$) | 3 | 1.5 | 2 | 1 | 3 | 3 | 3 | 4 | 4 |
| multi | 2 | 2 | 2 | 2 | 2 | 2 | 2.1 | 2 | 2 |
| Number of Orientations | 6 | 6 | 6 | 6 | 8 | 6 | 6 | 6 | 6 |
| Number of Scales | 4 | 4 | 4 | 4 | 4 | 3 | 4 | 4 | 4 |
| $\sigma$ | 0.65 | 0.65 | 0.65 | 0.65 | 0.65 | 0.65 | 0.65 | 0.65 | 0.65 |
| $d\theta$ on sigma | 1.5 | 1.5 | 1.5 | 1.5 | 1.5 | 1.5 | 1.5 | 1.5 | 1 |

**Table 3** The highest average recognition rates with different numbers of states

| State no. | 27 | 31 | 48 | 49 | 53 | 56 |
|---|---|---|---|---|---|---|
| 0 | 96.80 | 97.50 | 98.20 | 98.30 | **98.50** | 98.20 |
| 1 | 93.10 | 94.10 | 95.40 | 96.10 | 94.40 | **96.40** |
| 2 | 91.60 | 91.10 | 92.40 | 94.20 | **94.70** | 92.90 |
| 3 | 95.10 | 94.40 | 96.50 | **97.20** | 96.50 | 96.50 |
| 4 | 94.00 | **95.50** | 94.70 | 93.20 | 94.00 | 94.00 |
| 5 | 98.50 | 98.50 | **98.90** | 98.50 | 98.50 | 98.50 |
| 6 | 99.10 | 99.10 | 99.10 | **99.10** | 99.10 | 99.10 |
| 7 | 83.50 | 89.90 | 96.30 | 94.50 | **97.20** | 94.50 |
| 8 | 89.70 | 94.80 | 96.90 | 96.90 | **96.90** | 96.90 |
| 9 | **95.90** | 93.20 | 94.60 | 91.90 | 89.20 | 91.90 |
| Avg. | 95.343 | 96.130 | 97.151 | 97.209 | **97.212** | 97.093 |

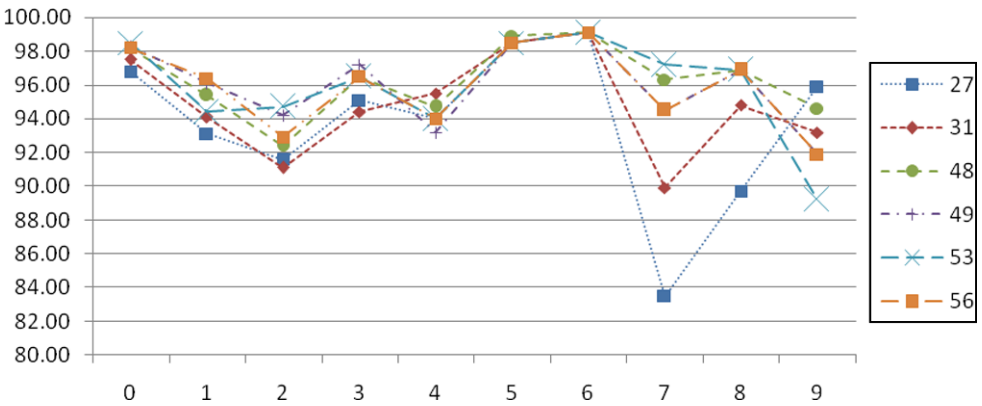


**Fig. 7** Recognition rates for each digit with HMM classifier by using 27, 31, 48, 49, 53 and 56 states

age recognition rate, viz. 94.43%, is shown by the curve of Experiment $e_3$ and it is achieved by using 4 scales (wavelengths of 2, 4, 8, and 16), 6 orientations (0°, 30°, 60°, 90°, 120° and 150°) and $3 \times 3$ segments.

### 4.2 HMM classifier

A number of experiments are conducted by using different widths of the sliding window and horizontal overlaps (window width of 4 pixels and overlap of 1 and 2 pixels; window width of 3 pixels and overlap of 1 and 2 pixels), and different numbers of vertical segments of the sliding window and vertical overlap (3, 4, 6 and 8 segments with a vertical overlapping 1/2 and 1/4 of the window height). The highest recognition rates are achieved with a window of 4 pixels wide, a horizontal overlap of 2 pixels and 8 vertical non-overlapping segments. Several experiments are conducted with these parameters by using a number of states for each experiment ranging from 8 states to 60 states, inclusive, for a total of 53 experiments. We achieve the highest recognition rates by using 27, 31, 48, 49, 53, and 56 states. Figure 7 shows the average recognition rate for these experiments for all digits.

Table 3 shows the recognition rates of Digits 0 to 9 for experiments with the number of states that produce the highest recognition rates for at least one digit. The highest average recognition rate of 97.212% is achieved by using 53 states

**Table 4** The confusion matrix of the tested samples with SVM classifier

| Actual Category | Predicted Category | | | | | | | | | | Recognition Rate |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | |
| 0 | 1568 | 5 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 99.62 |
| 1 | 8 | 296 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 97.37 |
| 2 | 1 | 0 | 222 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 98.67 |
| 3 | 0 | 0 | 0 | 142 | 2 | 0 | 0 | 0 | 0 | 0 | 98.61 |
| 4 | 1 | 0 | 3 | 0 | 129 | 0 | 0 | 0 | 0 | 0 | 96.99 |
| 5 | 2 | 0 | 0 | 0 | 0 | 261 | 0 | 0 | 0 | 0 | 99.24 |
| 6 | 0 | 1 | 0 | 0 | 0 | 0 | 110 | 0 | 0 | 0 | 99.10 |
| 7 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 107 | 0 | 0 | 98.17 |
| 8 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 96 | 0 | 97.96 |
| 9 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 72 | 97.30 |
| | | | | | | | | | Average | | 98.95 |

and the above configuration. This resulted in the highest recognition rates for digits 0, 2, 7, 8 and 9. The table shows that the average recognition rates of states 48, 49, 53 and 56 are very close.

### 4.3 SVM classifier

The check digits of Table 1 are used with the SVM classifier. The data is normalized before the feature extraction stage as was done in HMM. 70% of the data is used in training the SVM classifier and the remaining 30% is used for testing. A 10% V-fold on the training data is used for the estimation of the SVM parameters that results in the highest recognition rates. These parameters are then used in experimenting with SVMs by using the extracted features of the testing data. The number of scales, orientations and segments that resulted in the highest average recognition rates with 1-NN, 3-NN and NM are used. A 432-feature vector is extracted as described in Sect. 4.1.

Table 4 shows the confusion matrix of the tested samples by using 4 scales, 6 orientations, and $3\times3$ segments. The recognition rate was highest for Digit 0 at 99.62% and lowest for Digit 4 at 96.99%. It is clear that 13 out of 32 erroneous samples, 40.6% of the errors, are between Digits 0 and 1. It will be shown, in analyzing the errors in the following paragraphs, that this recognition rate is very high, given the type of samples that were misclassified.

The misclassified samples of the SVM classifier are shown in Fig. 8. Subjective evaluation of the misclassified digits was conducted. The samples were given to 22 undergraduate students to label each sample with the digit they perceive in the image. The misclassified samples are put in a form randomly so that the subjects are not biased by the sequence of the samples shown in Fig. 8. The students were told that these are images of Arabic (Indian) digits. These

| Number | Digit Image | Digit Class | SVM Class |
|---|---|---|---|
| 1 |  | 0 | 5 |
| 2 |  | 0 | 1 |
| 3 |  | 0 | 1 |
| 4 |  | 0 | 1 |
| 5 |  | 0 | 1 |
| 6 |  | 0 | 1 |
| 7 |  | 1 | 0 |
| 8 |  | 1 | 0 |
| 9 |  | 1 | 0 |
| 10 |  | 1 | 0 |
| 11 |  | 1 | 0 |
| 12 |  | 1 | 0 |
| 13 |  | 1 | 0 |
| 14 |  | 1 | 0 |
| 15 |  | 2 | 0 |
| 16 |  | 2 | 5 |

| Number | Digit Image | Digit Class | SVM Class |
|---|---|---|---|
| 17 |  | 2 | 5 |
| 18 |  | 3 | 4 |
| 19 |  | 3 | 4 |
| 20 |  | 4 | 2 |
| 21 |  | 4 | 0 |
| 22 |  | 4 | 2 |
| 23 |  | 4 | 2 |
| 24 |  | 5 | 0 |
| 25 |  | 5 | 0 |
| 26 |  | 6 | 1 |
| 27 |  | 7 | 3 |
| 28 |  | 7 | 0 |
| 29 |  | 8 | 0 |
| 30 |  | 8 | 0 |
| 31 |  | 9 | 1 |
| 32 |  | 9 | 6 |

**Fig. 8** The misclassified digits from the SVM classifier where the Digit Class column represents the correct digit whereas the SVM class column shows the erroneous decision of the classifier

results are summarized in Table 5. The responses of the students are classified into four categories. The first category includes the number of responses that correctly classified the digit. The second category indicates the number of responses that misclassified the digit to the predicted digit of the SVM classifier. The third category includes the number of responses that misclassified the digit differently from the SVM predicted digit. The final category includes all the responses that were either left blank or marked as "unknown".

The overall results show that the subjects were not able to conclusively recognize Digits 0 and 5. In fact, a lot of

**Table 5** The overall classification results of the human subjects

| Digit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Misclassified Samples Count | 6 | 8 | 3 | 2 | 4 | 2 | 1 | 2 | 2 | 2 | |
| Correctly classified | 6.82 | 86.36 | 89.39 | 100 | 63.64 | 43.18 | 86.36 | 93.18 | 100 | 68.18 | 67.19 |
| Incorrectly classified similar to the prediction of SVM classifier | 83.33 | 3.41 | 4.55 | 0 | 18.18 | 0 | 13.64 | 4.55 | 0 | 25.00 | 21.45 |
| Incorrectly classified different from SVM classifier prediction | 4.55 | 7.39 | 4.55 | 0 | 14.77 | 52.27 | 0 | 0 | 0 | 2.27 | 8.38 |
| Left blank or undetermined | 5.30 | 2.84 | 1.52 | 0 | 3.41 | 4.55 | 0 | 2.27 | 0 | 4.55 | 2.98 |

**Table 6** Recognition rates of the digits and the average recognition rates with 1-NN, 3-NN, NM, HMM and SVM Classifiers

| | 1-NN | 3-NN | NM | HMM | SVM |
|---|---|---|---|---|---|
| 0 | 99.49 | 99.11 | 96.12 | 98.50 | **99.62** |
| 1 | 98.03 | **98.36** | 90.13 | 94.40 | 97.37 |
| 2 | 97.78 | 98.67 | 89.78 | 94.70 | 98.67 |
| 3 | 97.92 | 97.92 | 84.03 | 96.50 | **98.61** |
| 4 | 93.98 | 93.98 | 91.73 | 94.00 | 96.99 |
| 5 | 99.24 | **99.62** | 96.58 | 98.50 | 99.24 |
| 6 | **100** | 99.10 | 99.10 | 99.10 | 99.10 |
| 7 | 98.17 | 99.08 | 95.41 | 97.20 | 98.17 |
| 8 | 98.98 | 98.98 | 96.94 | 96.90 | 97.96 |
| 9 | 95.95 | 93.24 | 95.95 | 89.20 | **97.30** |
| Avg. | **98.75** | **98.62** | **94.43** | **97.21** | **98.95** |

samples of Digit 0 have been classified by the surveyed students similar to the SVM classifier prediction. Digits 4 and 9 were correctly classified by no more than 69% of the human subjects. Some digit samples of 0, 5, and to some extent 4 and 9, were not reflecting the proper way of writing these digits. Overall, 67% of the students correctly recognized the digits, and the remaining 33% either misclassified the digits or were undecided. Therefore, one third of the digits misclassified by the SVM classifier were also misclassified by humans. Figure 8 shows that samples 1, 20–25 and 32 cannot be recognized by humans without context. Samples 20–23 are incomplete, due to wrong segmentation. Sample 24 is Digit 5 that was cut from the middle. Sample 25 does not resemble any digit. Sample 32 is digit 9 cut from its upper part.

### 4.4 1-NN, 3-NN, NM, HMM, and SVM classifiers

Table 6 shows the recognition rates of Digits 0 to 9 and the average recognition rates with 1-NN, 3-NN, NM, HMM and SVM classifiers. It is clear that the highest average recognition rates are achieved by 1-NN and SVM. The 1-NN average recognition rate is 0.09% higher than SVM. However, 1-NN requires that all training samples be stored in memory. This is not the case with SVM. Together, 1-NN and SVM, have the highest recognition rates for all the digits. In particular, 1-NN is highest for Digits 1, 2, 4, 5, 7 & 8, and SVM is highest for Digits 0, 3 & 9, and both achieved the same recognition rate for Digit 6. Figure 9 shows the average recognition rates for all the digits with 1-NN, 3-NN, NM, HMM and SVM classifiers.

Although the HMM average recognition rate is 1.74% less than SVM, HMM is more suited for Arabic text recognition. In addition, it has an advantage for the case of touching digits, where many segmentation errors are expected. The above applied technique of HMM requires no segmentation. This will be the subject of future research.

Table 7 shows the average recognition rates achieved by using Log Gabor-based features and the three cited references that used the same data [1, 3, 4]. Reference [1] used structural views with SVM and MLP classifiers. Reference [3] used Bernoulli mixture models. Reference [4] used spatial Gabor filters with 1-NN, 3-NN and NM classifiers. In order to verify that our results represent statistically significant improvements over previous work, the level of confidence, denoted by $1-\alpha$ was computed. Although for clas-

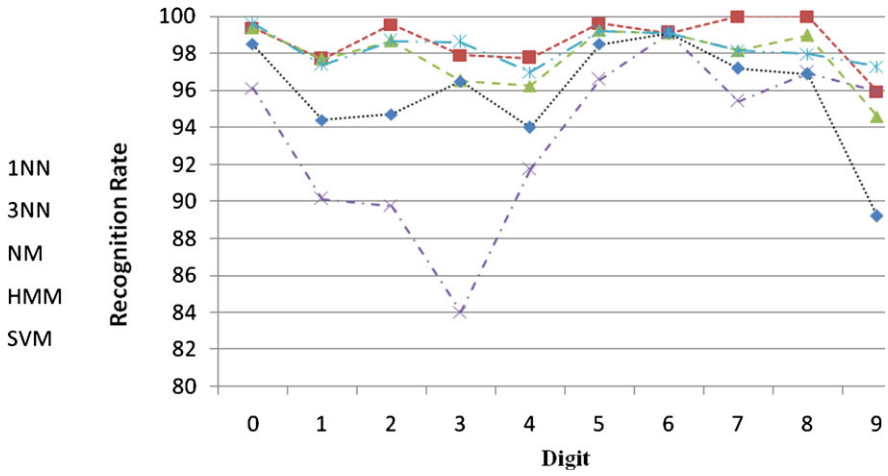


**Fig. 9** Comparing the average recognition results with 1-NN, 3-NN, NM, HMM and SVM

**Table 7** The average recognition rates using Log Gabor-based features of our technique vs. the techniques of references [1, 3, 4]

| Features and classifier | Average recognition rate | Statistical Significance Interval |
|---|---|---|
| Log Gabor Filters with 1-NN | **98.75%** | **±0.56** |
| Log Gabor Filters with 3-NN | **98.62%** | **±0.59** |
| Log Gabor Filters with NM | 94.43% | ±1.05 |
| Log Gabor Filters with HMM | 97.21% | ±0.79 |
| Log Gabor Filters with SVM | **98.95%** | **±0.53** |
| Structural views of [1] using SVM | 94.14% | ±1.07 |
| Structural views of [1] using MLP | 91.25% | ±1.27 |
| Bernoulli mixture models [3] | 97.5% | ±0.75 |
| Spatial Gabor Filters with 1-NN [4] | 97.99% | ±0.68 |
| Spatial Gabor Filters with 3-NN [4] | 97.37% | ±0.77 |
| Spatial Gabor Filters with NM [4] | 92.76% | ±1.17 |

sification, a level of confidence of $1 - 0.05 = 0.95$ (95%) is usually used [30], we present, below, our results computed with a level of confidence of $1 - 0.01 = 0.99$ (99%) that clearly highlight our strong results. The recognition rates using Log Gabor filters using 1-NN, 3-NN, and SVM all present statistically significant improvements over the best reported results in [1, 3, 4]. The results of Log Gabor filter using HMM represent a statistically insignificant degradation, compared to the best result reported in [4]. This result of HMM is in line with the results of other researchers when HMM is used for isolated digit recognition. However, HMM have an advantage with cursive text recognition as segmentation, which is problematic and error prone, is by-product of HMM. Other classifiers require the segmentation of cursive text a priori. Hence, HMM recognition rate is expected to be higher than other classifiers for cursive text. Given that such a simple classifier as K-NN is used, the results indicate the effectiveness of Log Gabor-based features in the automatic recognition of Arabic (Indian) bank check digits. It is to be noted that the suitable numbers of scales and orientations that produce the highest recognition rates are problem-dependent, and they need to be experimentally determined.

## 5 Conclusions

This paper presents a technique based on Log Gabor filters for the automatic recognition of Arabic (Indian) handwritten check digits. Log Gabor-based features at different numbers of scales, orientations, and different number of filtered images' sizes are tested with 1-NN, 3-NN, NM, SVM and HMM classifiers. A database of 10425 digit samples (7090 for training and 3035 for testing, as organized by CENPARMI) is used in the experimentation.

A number of scales, orientations and image segments are tested, and those that resulted in the highest recognition rates are presented. Average recognition rates of 98.75%, 98.62%, 94.43%, 97.21% and 98.95% are obtained by using 4 scales, 6 orientations and $3 \times 3$ segments per sample with 1-NN, 3-NN, NM, HMM and SVM classifiers, respectively. These results are compared with previously published

work using the same data. It is clear that Log Gabor-based features with 4 scales, 6 orientations, and $3 \times 3$ filtered image segments achieved highest recognition rates by using 1-NN and SVM. The misclassified digits are evaluated subjectively and results indicate that human subjects misclassified $1/3$ of these digits. The experimental results, including the subjective evaluation of misclassified digits, indicate the effectiveness of the selected Log Gabor filters parameters, the implemented image segmentation technique, and extracted features for practical recognition of Arabic (Indian) digits. The suitable numbers of scales, orientations and segments that produce the highest recognition rates are problem-dependent, and they need to be experimentally determined. The authors are investigating the use of Log Gabor filters in printed Arabic text recognition and in the recognition of touching bank check numerals.

**Acknowledgement** The authors would like to thank the reviewers for their invaluable comments. We would also thank King Fahd University of Petroleum & Minerals for supporting this research and providing the computing facilities.

## References

1. Sadri J, Suen CY, Bui TD (2003) Application of support vector machines for recognition of handwritten Arabic/Persian digits. In: Proceeding of the second conference on machine vision and image processing & applications (MVIP2003), Tehran, Iran, pp 300–307
2. Al-Ohali Y, Cheriet M, Suen C (2003) Databases for recognition of handwritten Arabic cheques. Pattern Recogn 36:111–121
3. Juan A, Vidal E (2004) Bernoulli mixture models for binary images. Pattern Recogn 3:367–370
4. Mahmoud SA (2009) Recognition of Arabic (Indian) check digits using spatial gabor filters. In: 5th IEEE-GCC conference, Kuwait
5. Harifi A, Aghagolzadeh A (2005) A new pattern for handwritten Persian/Arabic digit recognition. Int J Inf Technol 1:174–177
6. Mahmoud S (2008) Recognition of writer-independent off-line handwritten Arabic (Indian) numerals using hidden Markov models. Signal Process 88:844–857
7. Mozaffari S, Faez K, Ziaratban M (2005) Structural decomposition and statistical description of Farsi/Arabic handwritten numeric characters. In: Proceedings of the eighth international conference on document analysis and recognition. IEEE Computer Society, Los Alamitos, pp 237–241
8. Salah A, Alpaydin E, Akarun L (2002) A selective attention-based method for visual pattern recognition with application to handwritten digit recognition and face recognition. IEEE Trans Pattern Anal Mach Intell 24:420–425
9. Liu C, Koga M, Fujisawa H (2005) Gabor feature extraction for character recognition: comparison with gradient feature. In: ICDAR '05: Proceedings of the Eighth International Conference on Document Analysis and Recognition. IEEE Computer Society, New York, pp 121–125
10. Wang X, Ding X, Liu C (2005) Gabor filters-based feature extraction for character recognition. Pattern Recogn 38:369–379
11. Kruizinga P, Petkov N, Grigorescu S (1999) Comparison of texture features based on Gabor filters. In: Image analysis and processing, 1999. Proceedings. International conference on, pp 142–147
12. Camastra F (2007) A SVM-based cursive character recognizer. Pattern Recogn 40:3721–3727
13. Soltanzadeh H, Rahmati M (2004) Recognition of Persian handwritten digits using image profiles of multiple orientations. Pattern Recogn Lett 25:1569–1576
14. Mozaffari S, Faez K, Kanan HR (2004) Feature comparison between fractal codes and wavelet transform in handwritten alphanumeric recognition using SVM classifier. In: Proceedings of the pattern recognition, 17th international conference on (ICPR'04), vol 2. IEEE Computer Society, New York, pp 331–334
15. Mowlaei A, Faez K (2003) Recognition of isolated handwritten Persian/Arabic characters and numerals using support vector machines. In: Neural networks for signal processing, NNSP'03. 2003 IEEE 13th workshop on, pp 547–554
16. Vapnik VN (1999) The nature of statistical learning theory. Springer, Berlin
17. Vapnik V (1982) Estimation of dependences based on empirical data. Springer series in statistics. Springer, New York
18. Boser BE, Guyon IM, Vapnik VN (1992) A training algorithm for optimal margin classifiers. In: Proceedings of the fifth annual workshop on computational learning theory, Pittsburgh, Pennsylvania, United States. ACM, New York, pp 144–152
19. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20:273–297
20. Scholkopf B, Burges C, Vapnik V (1995) Extracting support data for a given task. In: Proceedings, first international conference on knowledge discovery & data mining. AAAI Press, Menlo Park
21. Schölkopf B, Burges C, Vapnik V (1996) Incorporating invariances in support vector learning machines. In: Proceedings of the 1996 international conference on artificial neural networks. Springer, Berlin, pp 47–52
22. Schölkopf B, Simard P, Smola A, Vapnik V (1998) Prior knowledge in support vector kernels. In: Proceedings of the 1997 conference on Advances in neural information processing systems 10, Denver, Colorado, United States. MIT Press, Cambridge, pp 640–646
23. Vapnik VN (1998) Statistical learning theory. Wiley-Interscience, New York
24. Alma'adeed S, Higgins C, Elliman D (2004) Off-line recognition of handwritten Arabic words using multiple hidden Markov models. Knowl-Based Syst 17:75–79
25. Bazzi I, LaPre C, Makhoul J, Raphael C, Schwartz R (1997) Omnifont and unlimited-vocabulary OCR for English and Arabic. In: Proceedings of the fourth international conference on document analysis and recognition, vol 2, pp 842–846
26. Bazzi I, Schwartz R, Makhoul J (1999) An omnifont open-vocabulary OCR system for English and Arabic. IEEE Trans Pattern Anal Mach Intell, vol 21, pp 495–504
27. Mohamed M, Gader P (1996) Handwritten word recognition using segmentation-free hidden Markov modeling and segmentation-based dynamic programming techniques. IEEE Trans Pattern Anal Mach Intell 18:548–554
28. Hu J, Gek Lim S, Brown MK (2000) Writer independent on-line handwriting recognition using an HMM approach. Pattern Recogn 33:133–147
29. Hassin AH, Tang X, Liu J, Zhao W (2004) Printed Arabic character recognition using HMM. J Comput Sci Technol 19:538–543
30. Plotz T (2005) Advanced stochastic protein sequence analysis. PhD Thesis, Faculty of Technology, Bielefeld University

**Sabri A. Mahmoud** is a Professor of computer Science in the Information and Computer Science Department, King Fahd University of Petroleum and Minerals. Dr. Mahmoud received his B.S. in electrical engineering from Sind University, Pakistan in 1972, received his M.S. in Computer Sciences from Stevens Institute of Technology, USA, in 1980 and his Ph.D. degree in Information Systems Engineering from the University of Bradford, UK, in 1987. His research interests include Pattern Recognition, Arabic Document Analysis and Recognition (including Arabic text recognition and writer identification), Image Analysis (including Time Varying Image Analysis and Computer Vision), and Arabic Computing. Dr. Mahmoud is a senior member of IEEE. He published over 60 papers in refereed journals and conference proceedings in his research areas of interest.

**Wasfi G. Al-Khatib** received his B.S. degree in computer science from Kuwait University in 1990, and his M.S. degree in computer science and Ph.D. in Electrical and Computer Engineering from Purdue University in 1995 and 2001, respectively. He is currently an assistant professor at King Fahd University of Petroleum and Minerals. His research interests include Arabic computing, multimedia computing, content-based retrieval, artificial intelligence, and software engineering. He is a member of the ACM and the IEEE Computer Society.