Markov chain models based on genetic algorithms for texture and speech recognition

Shiueng-Bien Yang
You-Ping Huang
Leader University
Department of Computer Science and Information Engineering
No. 188 Sec. 5 An-chung Road
Tainan City, Taiwan

Abstract. *Markov chain models (MCMs) were recently adopted in many recognition applications. The well-known clustering algorithm, the k-means algorithm, is used to design the codebooks of the MCM, and then each code word in the codebook is regarded as one state of MCM. However, users usually have no idea how to determine the number of states before the design of the MCM, and therefore doubt whether the MCM produced by the k-means algorithm is optimal. We propose a new MCM based on the genetic algorithm for recognition applications. Genetic algorithms combine the clustering algorithm and the MCM design. The users do not need to define the size of the codebook before the design of the MCM. The genetic algorithm can automatically find the number of states in MCM, and thereby obtain a near-optimal MCM. Furthermore, we propose the fuzzy MCM (FMCM) and the fuzzy genetic algorithm (FGA) to enhance the recognition rate. Experimental results show that the proposed MCM outperforms the traditional MCM and other texture and speech recognition methods.* © 2006 SPIE and IS&T.
[DOI: 10.1117/1.2234731]

1 Introduction

The hidden Markov model (HMM) and its variants have recently been successfully adopted in recognition and classification applications.[1–4] However, the HMM has two limitations. One such is that a high computation cost is required to calculate the likelihood function of a texture HMM. If an input vector consists of T features, then the likelihood score for the N-state HMM takes $N(N+1)(T-1)+N$ multiplications and $N(N-1)(T-1)$ additions to compute the likelihood score for the N-state HMM. The second limitation is that the users must determine the number of states before designing the HMM. However, the users cannot easily determine the number of states in HMM.

This paper applies the Markov chain model (MCM) to texture and speech recognition to reduce the computation time and thus improve the overall performance of HMM. In an MCM, the computation time required to evaluate the likelihood score is $T-1$ multiplications, which is less than that required in HMM. The MCM is designed in two steps.[5] First, a clustering algorithm such as the k-means algorithm[6] is applied to the training data set to design the MCM codebook. Then, each code word in the codebook is regarded as a state in MCM. Step 2 of the MCM design continues after the k-means algorithm creates the codebook. The initial state and state-transaction probabilities of MCM are estimated to achieve the highest possible likelihood function. The MCM is thus produced after these two steps are finished. Figure 1 shows an example of this process. Let the training data set be a collection of 2-D vectors, as shown in Fig. 1(a). Notably, the real speech and texture data can be represented as vectors, which normally have more than two dimensions. Assume that the user designs an MCM with three states. The k-means algorithm is then applied to the training data set to search for three clusters, C_1, C_2, and C_3, which are shown in Fig. 1(b). A cluster indicates a group of vectors. The variation is high between the different clusters, and is low within each cluster. Each cluster center S_i can be obtained by calculating the mean of vectors contained in the cluster C_i. The cluster center S_i can represent all of the vectors contained in the cluster C_i because the vectors within the same cluster have low variation. The unknown vector is compared with these three centers—S_1, S_2, and S_3—to determine the closest one to it. The collection of these three centers—S_1, S_2, and S_3—can also be regarded as a codebook V and each center S_i can be represented as a code word v_i in V. Each code word v_i in the MCM is also regarded as a state in the MCM. The following section describes the design of the initial state and state-transaction probabilities between these three states. The MCM with three states is then produced as shown in Fig. 1(c).

However, the users cannot easily determine the number of states before the MCM is designed, since the k-means algorithm requires the users to input the number of clusters in the data set, which the users usually do not know. Hence, the users are forced to try different numbers of clusters when using these clustering algorithms. This approach is tedious and is likely to produce poor clustering results, especially when the number of clusters is large and not easy to guess. The number of states in MCM is generally set to a fixed value. However, the fixed number of states is not suitable for each MCM. Additionally, k-means is an iterative hill-climbing algorithm with a solution depending on the initial clustering. Although k-means has been applied to many practical clustering problems successfully, it may fail to converge to a local minimum under certain conditions.[7] However, users usually doubt whether the MCM produced

Paper 05161RR received Sep. 6, 2005; revised manuscript received Jan. 23, 2006; accepted for publication Jan. 25, 2006; published online Jul. 24, 2006.
1017-9909/2006/15(3)/033004/12/$22.00 © 2006 SPIE and IS&T.

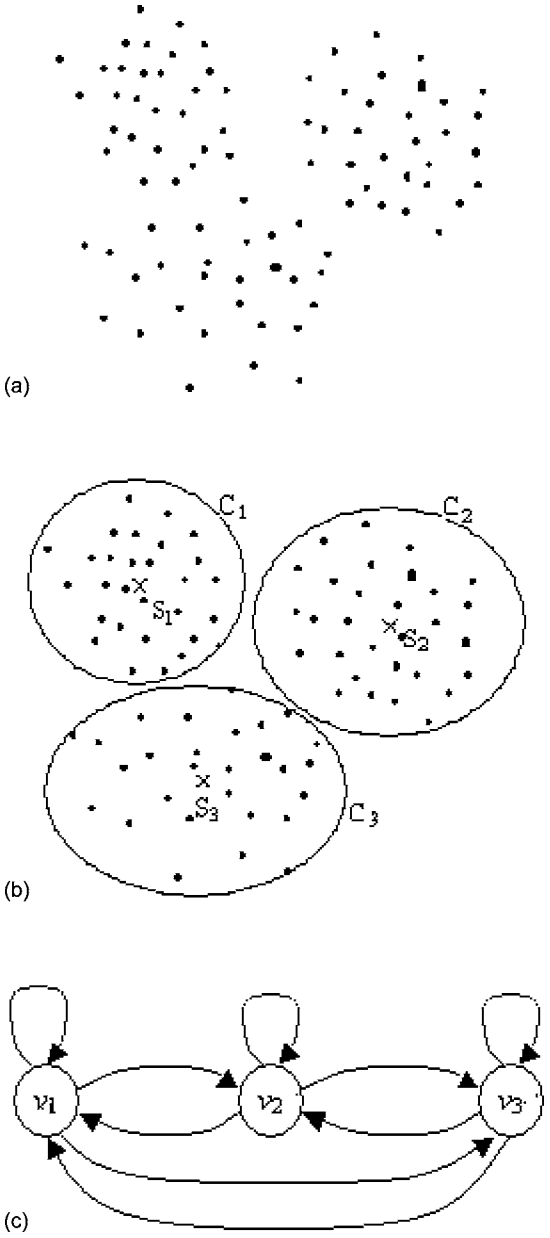

Fig. 1 Design of the traditional MCM by the *k*-means algorithm: (a) training data set, (b) three clusters generated by the *k*-means algorithm, and (c) MCM with three states.

is optimal, because *k*-means is used only to design the codebook, and it never maximizes the likelihood function of the MCM.

Genetic algorithms search in complex and large solution spaces and provide a near-optimal solution for an optimization problem. The genetic strategy uses a fitness function to evaluate the goodness of a chromosome. A chromosome represents a possible solution, and chromosomes with greater fitness function values are more likely to be reproduced in the next generation. This paper proposes a genetic algorithm that combines a clustering algorithm and an MCM design. The users do not have to set the number of states in the MCM. The genetic algorithm can automatically determine the proper number of states in MCM by maximizing the likelihood function of MCM. Additionally, the smoothing parameters are applied to the MCM and the genetic algorithm to enhance the recognition rate. Thus, this paper also proposes the fuzzy MCM (FMCM) and fuzzy genetic algorithm (FGA). The experimental results show that the proposed MCM based on the genetic algorithm outperforms the traditional MCM based on the *k*-means algorithm in texture and speech recognition.

The rest of this paper is organized as follows. Section 2 describes the basic design of the MCM. Section 3 then describes the design of the MCM based on the genetic algorithm. Next, Sec. 4 presents the designs of the FMCM and the FGA. Section 5 summarizes the experimental results. Conclusions are finally drawn in Sec. 6.

2 Basic Concept of the MCM

Let $X=X_1, X_2, \ldots, X_t, \ldots$ denote a series in discrete time, where each X_t is the t'th variable in the observation. The probability that variable X_t takes value x_t generally depends on the previous variables. Thus, the conditional probability that variable X_t takes value x_t is given by

$$P(X_t = x_t | X_{t-1} = x_{t-1}, X_{t-2} = x_{t-2}, \ldots, X_1 = x_1). \quad (1)$$

However, such a model is too complex for speech applications. To reduce the complexity of Eq. (1), MCM assumes that variable X_t takes value x_t based on the immediately preceding outcome x_{t-1}. Then, the probability in Eq. (1) can be rewritten as

$$P(X_t = x_t | X_{t-1} = x_{t-1}). \quad (2)$$

The design of the traditional MCM is described as follows. Let $x=(x_1, x_2, \ldots, x_T)$ denote an observation, where each $\boldsymbol{x}_i$ denotes a vector in x. In MCM, each vector $\boldsymbol{x}_i$ can be mapped to a codebook V consisting of M code words, given by $v_1, v_2, \ldots, v_M$. As the time duration of x is limited, a Markov chain can be considered to have a limited duration, given by $X=X_1, X_2, \ldots, X_T$, with a state space defined in the codebook V. That is, each code word in V is a specific state of the Markov chain, allowing $x_1, x_2, \ldots, x_T$ to be represented as a sequence of states. The initial state and state-transaction probabilities of MCM can then be defined as

$$q(i) = P(X_1 = v_i) \quad i = 1,2, \ldots, M, \quad (3)$$

$$p(i,j) = P(X_t = v_j | X_{t-1} = v_i) \quad i,j = 1,2, \ldots, M. \quad (4)$$

Let $\lambda=(\hat{q}, \hat{p})$ denote an MCM, where $\hat{q}=[q(i)]$ and $\hat{p}=[p(i,j)]$. Then, the probability

$$P(X = x | \lambda) = q(x_1) \prod_{t=2}^{T} p(x_{t-1}, x_t) \quad (5)$$

is used to measure the likelihood of observing sequence x in the MCM λ. In recognition applications, an unknown observing sequence x is the input for all the MCMs and, then, the MCM with the largest probability defined in Eq. (5) is regarded as the recognition result of x.

The user must give the number of states, in the model, given by M, before designing the traditional MCM. The *k*-means algorithm, which is a well-known clustering

method, is then applied to the training data set to obtain M clusters. The center of each cluster is regarded as a codeword in the codebook V. Moreover, the maximum likelihood method can be adopted to estimate the Markov chain parameters. Let the set $Z=\{x^1,x^2,\ldots,x^N\}$ contain N training observations, with each training observation x^f represented by n_f vectors, $\boldsymbol{x}_1^f,\boldsymbol{x}_2^f,\ldots,\boldsymbol{x}_{n_f}^f$. Then, the joint likelihood function should be maximized.

$$P(Z|\lambda)=\prod_{f=1}^{N}P(\boldsymbol{x}^f|\lambda), \tag{6}$$

where $P(\boldsymbol{x}^f|\lambda)$ is defined in Eq. (5). As in Ref. 5, the following estimates can be obtained by taking the logarithm of Eq. (6) and using the Lagrangian method:

$$q(i)=\frac{m_i}{\sum_{k=1}^{M}m_k}, \tag{7}$$

and

$$p(i,j)=\frac{m_{i,j}}{\sum_{k=1}^{M}m_{i,k}}, \tag{8}$$

where m_i indicates the number of vectors x_1^f, for $1\le f\le N$, mapped to the codeword v_i, and $m_{i,j}$ indicates the number of vectors that the code word v_j comes from the code word v_i in $\boldsymbol{x}^f$ for $1\le f\le N$.

The traditional design of the MCM (Ref. 5) is described as follows.

Input

$Z=\{x^1,x^2,\ldots,x^N\}$ containing N observations, where each observation $\boldsymbol{x}^f$ contains n_f vectors, for $1\le f\le N\cdot M$: number of states.

Output

MCM λ with M states.

Step 1. Parameter M is the input of the k-means algorithm. Then, the k-means algorithm is applied to all vectors in Z to generate M clusters. Each cluster is regarded as the state of the MCM.
Step 2. Calculate the initial state and state-transaction probabilities, $q(i)$ and $p(i,j)$, as defined in Eqs. (7) and (8).
Step 3. Output the MCM λ with M states.
Step 4. End.

3 Design of the Markov Chain Models Based on the Genetic Algorithm

The traditional MCM has two drawbacks. First, the users must give the number of states before designing the MCM before the k-means algorithm can be applied to the training data set. Unfortunately, the users usually do not know the number of clusters in the training data set, and therefore cannot easily decide the number of states in the MCM.

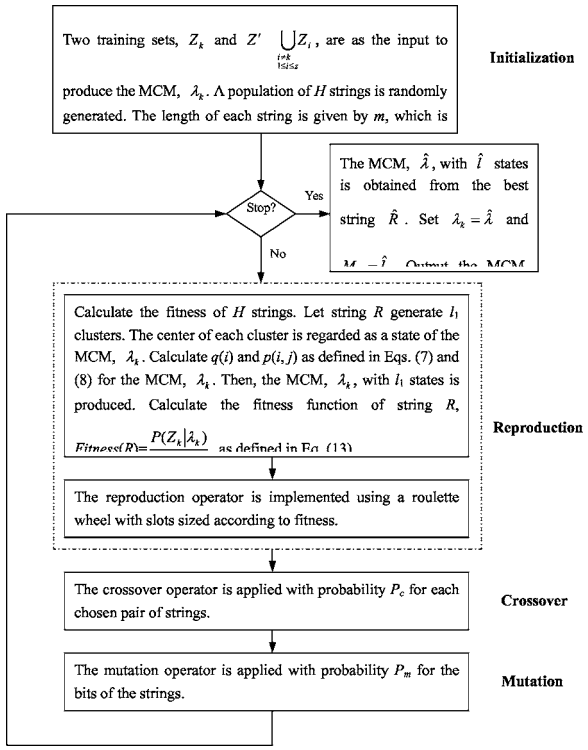

Fig. 2 Flow chart of the genetic algorithm for designing the MCM λ_k.

Therefore, selecting the number of states in MCM is a tedious task of trial and error. For example, the number of states M is set to 128 in Ref. 5. If the number of states in MCM is too large, then the values of $q(i)$ and $p(i,j)$ are too low, because the values of both $q(i)$ and $p(i,j)$ depend on the number of states in MCM. Additionally, if the number of states in MCM is too small, then the training data set used to design the MCM is divided into a small number of large clusters. However, the training data cannot easily be represented in a large cluster using only one center, especially, when the large cluster consists of many different training data. The second drawback of the traditional MCM is that the users cannot be sure that the MCM obtained is optimal. The users must maximize the value of Eq. (6) when designing the MCM, but cannot know whether the codebook produced by the clustering algorithm, such as the k-means algorithm, is suitable for the MCM, because the k-means algorithm is used only to find a clustering result from the training data set, and it never emphasizes maximizing the value of Eq. (6) during the of the MCM codebook design.

This section proposes the genetic algorithm to design the traditional MCM. The genetic algorithm can automatically search the proper number of states, and seeks to maximize the value of Eq. (6) during the design of the MCM codebook. Thus, the MCM produced by the genetic algorithm is better than that produced by the k-means algorithm. The genetic algorithm design is described as follows. Assume that z training data sets, $Z_1,Z_2,\ldots,Z_z$, are used to design z MCMs, $\lambda_1,\lambda_2,\ldots,\lambda_z$, respectively. To enhance the recognition rate of the applications, the design aim of the genetic algorithm is to enhance the distinction between the z MCMs. The genetic algorithm in the proposed method de-

Table 1 The speech recognition rates of the proposed MCMs and the other MCMs.

Methods	MCM (KM&fixed)	MCM(KM)	MCM(GA)	MCM(FGA)	FMCM(GA)	FMCM(FGA)
Average recognition rates (%)	89.8	92.1	97.3	98.3	98.5	99.1

signs one MCM at a time. This paper describes the design of the MCM λ_k for $1 \le k \le z$, by using the genetic algorithm, which is shown in Fig. 2. The genetic algorithm consists of an initialization step and iterations with three phases in each generation.

3.1 *Initialization Phase*

Let $Z_k = \{x^1, x^2, \ldots, x^{N_k}\}$ consist of N_k training observations, where each observation $\boldsymbol{x}^f$ is represented by n_f sequential vectors, $\boldsymbol{x}^f = (\boldsymbol{x}_1^f, \boldsymbol{x}_2^f, \ldots, \boldsymbol{x}_{n_f}^f)$, for $1 \le f \le N_k$. That is, Z_k consists of $\Sigma_{f=1}^{N_k} n_f$ vectors. Set $Z' = \cup_{\substack{1 \le i \le z}}^{i \ne k} Z_i$, where Z' consists of $\Sigma_{\substack{i=1}}^{z} \Sigma_{f=1}^{N_i} n_f$ vectors. A good MCM maximizes the likelihood function of the model λ_k when inputting the training observations in Z_k, and minimizes it when inputting the training observations in Z'. To achieve the performance of the genetic algorithm for processing the large data set, the pairwise nearest-neighbor (PNN) algorithm[8] is first applied to the training data set Z_k. The two closed features (or clusters) in PNN can be merged to form a new cluster, and this merge processing continues until the desired number of clusters is obtained. Denote m as the number of clusters, $B_1, B_2, \ldots, B_m$, obtained after the PNN is applied to the training data set, and let u_i indicate the center of cluster B_i, for $1 \le i \le m$. Each cluster B_i is regarded as a component, and is divided during the genetic algorithm. That is, the genetic algorithm only clusters m components ($m \ll N_k$). The PNN algorithm can be applied to reduce the computation time in the genetic algorithm. Therefore, the genetic algorithm can efficiently process the large data set.

A population of H strings is randomly generated in the initialization step of the genetic algorithm. The length of each string is given by m, which is the number of the components obtained by the PNN algorithm. The algorithm generates H strings, where the 1's in the strings are uniformly distributed within $[1, m]$. Each string represents a subset $\{B_1, B_2, \ldots, B_m\}$. If B_i is in a subset, then the i'th position of the string is 1, and is 0 otherwise. Each B_i in the subset is a seed to generate a cluster.

The method for generating a clustering from the seeds is described before the three phases are elucidated. Let $R = (b_1, b_2, \ldots, b_m)$ be a bit string in the population. Each bit b_i indicates the corresponding component B_i. The string R then includes two sets of components, L_1 and L_2, which are defined as

$$L_1 = \{B_i | b_i = 1, 1 \le i \le l_1\}, \tag{9}$$

$$L_2 = \{B_j' | b_j = 0, 1 \le j \le l_2\}, \tag{10}$$

where $l_1 + l_2 = m$. In L_1, l_1 components, B_i for $1 \le i \le l_1$, are used as the seeds to generate l_1 clusters. Initially, each cluster C_i contains only one component, B_i, and then the center S_i of cluster C_i is set to u_i. The components in L_2 are then considered one at a time, and the Euclidean distances between each component and the centers S_i, for $1 \le i \le l_1$ are calculated. Then,

$$B_j' \subset C_i \text{ if } \|u_j' - S_i\| \le \|u_j' - S_q\|, \quad \text{for } 1 \le q \le l_1. \tag{11}$$

After each B_j' is classified into the cluster C_i to form the new cluster $\hat{C}_i$, the new center $\hat{S}_i$ is updated as

$$\hat{S}_i = \frac{\sum_{B_j' \subset \hat{C}_i} |B_j'| u_j'}{\sum_{B_j' \subset \hat{C}_i} |B_j'|}, \tag{12}$$

where $|B_j'|$ indicates the number of features in the component B_j'. The components in L_2 yield l_1 clusters, denoted C_i for $1 \le i \le l_1$, the string R. Notably, the center of each cluster C_i is represented as the code word in the codebook V, and the number of code words in V is regarded as the number of states in MCM.

3.2 *Reproduction Phase*

The main issue of the reproduction phase is the design of the fitness function for the string R. Let string R generate l_1 clusters, C_i for $1 \le i \le l_1$, and consider the center of each cluster C_i as a state in MCM. Thus, the MCM λ_k is generated with l_1 states from the string R. Calculate $q(i)$ and $p(i,j)$ as defined in Eqs. (7) and (8) for the MCM λ_k. The fitness function of string R is then defined as,

$$\text{Fitness}(R) = \frac{P(Z_k|\lambda_k)}{P(Z'|\lambda_k)}, \tag{13}$$

where $P(Z_k|\lambda_k)$ and $P(Z'|\lambda_k)$ are defined as Eq. (6). Equation (13) maximizes the likelihood function of the model λ_k when the training data set Z_k is the input, and minimizes it when the training data set Z' is the input.

After the fitness of each string in the population is calculated, the reproduction operator is implemented using a roulette wheel with slots sized according to fitness.

3.3 *Crossover Phase*

When the crossover operator is applied to a selected pair of strings R and Q, then two random numbers e and f in $[1,m]$ are generated to determine the pieces of strings to be interchanged. If $e<f$, then the bits from position e to position f of string R are interchanged with the bits in the same positions of string Q. The crossover operator is applied with probability P_c for each chosen pair of strings.

3.4 *Mutation Phase*

During the mutation phase, the bits of the strings in the population are chosen from $[1,m]$ with probability P_m. Each chosen bit is then changed from 0 to 1 or from 1 to 0. That is, if one bit is chosen, then a selected cluster is discarded or produced in a string.

The user may specify the number of generations over which to run the genetic algorithm before obtaining the string with the best fitness. Suppose that model $\hat{\lambda}$ with $\hat{l}$ states is obtained from the best string $\hat{R}$ after many generations. Then, $\lambda_k=\hat{\lambda}$ and $M_k=\hat{l}$ are set and the MCM λ_k with M_k states are obtained.

In the following, we describe how to design z MCMs, $\lambda_1,\lambda_2,\ldots,\lambda_z$, by the genetic algorithm.

Input

Training data sets, $Z_1,Z_2,\ldots,Z_z$, are used to design z MCMs, $\lambda_1,\lambda_2,\ldots,\lambda_z$, respectively.

Output

z MCMs, $\lambda_1,\lambda_2,\ldots,\lambda_z$.

Step 1. Set $k=1$.
Step 2. Set $Z'=\cup_{1\le i\le z}^{i\neq k} Z_i$. The genetic algorithm designs the MCM λ_k as shown in Fig. 1.
Step 3. Output the MCM λ_k with M_k states.
Step 4. Set $k=k+1$. If $k\le z$, then go to step 2.
Step 5. End.

The following algorithm describes how to use the MCMs when an unknown observation is input.

Input

z MCMs, $\lambda_1,\lambda_2,\ldots,\lambda_z$, and the unknown observation, $\boldsymbol{x}^t=(\boldsymbol{x}_1^t,\boldsymbol{x}_2^t,\ldots,\boldsymbol{x}_{n_t}^t)$, where $\boldsymbol{x}_i^t$ is a vector for $1\le i\le n_t$.

Output

The MCM, $\lambda^*\in\{\lambda_1,\lambda_2,\ldots,\lambda_z\}$, that is closest to $\boldsymbol{x}^t$.

Step 1. Calculate $g=\text{argmax}_{1\le k\le z}P(\boldsymbol{x}^t|\lambda_k)$, where $P(\boldsymbol{x}^t|\lambda_k)$ is defined as in Eq. (5).
Step 2. Set $\lambda^*=\lambda_g$. The MCM λ^* is output.

The time complexity of the genetic algorithm is analyzed as follows. The genetic algorithm consists of an initialization step and iterations with three phases in each generation. Let N be the number of training features. Before the genetic algorithm is applied to these N features, the PNN algorithm takes $O(N^2)$ time to calculate the distances between pairs of objects, and $O(N)$ time to find the minimum. Briefly, the PNN algorithm takes $O(N^2)$ to find m components from these N training features. In the genetic algorithm, let H denote the size of population, and let m denote the total number of components. Each component takes $O(m^2)$ time to find the nearest cluster. The time complexity of the genetic algorithm is dominated by the calculation of the fitness function. The algorithm takes $O(Nm^2)$ time in the worst case. Suppose the genetic algorithm is asked to run G generations, then the time complexity is given by $O(GNm^2)$. Hence, the time complexity of the whole genetic algorithm is $O(N^2+GNm^2)$.

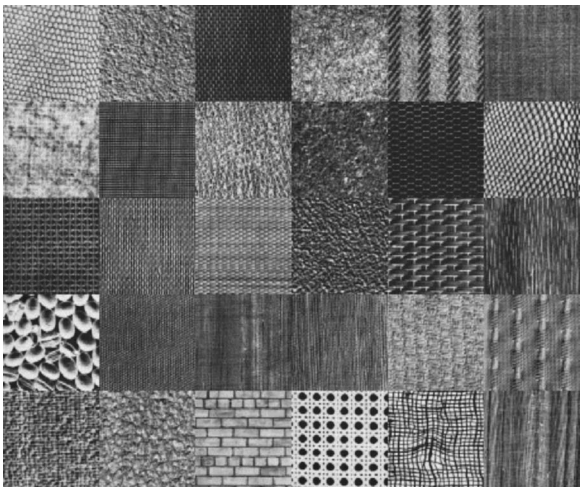

Fig. 3 Thirty textures from Brodatz album. Row 1: D3, D4, D6, D9, D11, D16; row 2: D19, D21, D24, D29, D34, D36; row 3: D52, D53, D55, D57, D65, D68; row 4: D74, D77, D78, D79, D82, D83; and row 5: D84, D92, D95, D102, D103, D105.

The proposed MCM based on the genetic algorithm has two benefits. First, the genetic algorithm can automatically generate the proper number of states for each MCM. The users do not need to give the number of states before designing the MCM in the proposed method, while the k-means algorithm must have the proper number of states before designing the traditional MCM. The second advantage of the proposed method is that the MCM designed by the genetic algorithm is a near- and globally-optimal MCM, because the genetic algorithm maximizes that the fitness function defined in Eq. (13). That is, when the MCM λ_k is designed, $P(Z_k|\lambda_k)$ is maximized in Eq. (13), making the observations in Z_k close to λ_k. Additionally, $P(Z'|\lambda_k)$ is minimized, so that the observations in Z' are far from λ_k. Thus, the proposed method enhances the recognition rate of λ_k. Therefore, the recognition rate of the proposed MCM based on the genetic algorithm is better than the traditional MCM based on the k-means algorithm.

4 Design of the FMCMs and the FGA

In the design of the MCM, the amount of training observations is required to estimate the parameters of the MCM reliably. However, the currently available training data is limited, and is not enough to cover all possible data. Thus, parameter smoothing is applied to the design of the FMCM and the FGA.

Table 2 The number of states of each MCM by using MCM(GA).

Letters	A	B	C	D	E	F	G	H	I	J	K	L	M
No. states	24	26	25	17	24	25	27	22	25	26	25	18	18
Letters	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
No. states	16	25	16	17	24	28	16	19	24	18	19	24	25

4.1 *FMCMs*

Let $\boldsymbol{x}$ denote an input vector and let $v_1, v_2, \ldots, v_M$ represent the centers of M clusters. In the FMCM, $\boldsymbol{x}$ be a member of any cluster. The degrees of membership that $\boldsymbol{x}$ has in the clusters are represented by functions $m_i(\boldsymbol{x})$ for $i=1,2,\ldots,M$, which satisfy the constraints that $0 \le m_i(\boldsymbol{x}) \le 1$ and $\Sigma_i m_i(\boldsymbol{x})=1$. The membership function $m_i(\boldsymbol{x})$ is defined as

$$m_i(\boldsymbol{x}) = \frac{1}{\sum_{k=1}^{M} (\|\boldsymbol{x}-v_i\|/\|\boldsymbol{x}-v_k\|)}. \tag{14}$$

Using the membership function $m_i(\boldsymbol{x})$ the FMCM can be estimated as follows. Let the set $Z=\{\boldsymbol{x}^1,\boldsymbol{x}^2,\ldots,\boldsymbol{x}^N\}$ contain N training observations and represent each observation $\boldsymbol{x}^f$ as $\boldsymbol{x}^f=(\boldsymbol{x}_1^f,\boldsymbol{x}_2^f,\ldots,\boldsymbol{x}_{T_f}^f)$. Then, Eq. (6) and (7) can be redefined as Eq. (15) and (16) in FMCM, respectively.

$$q(i) = \frac{1}{N}\sum_{k=1}^{N} m_i(\boldsymbol{x}_1^k), \quad i=1,2,\ldots,M, \tag{15}$$

$$p(i,j) = \frac{\sum_{k=1}^{N}\sum_{t=2}^{T_k} m_i(x_{t-1}^k)m_j(x_t^k)}{\sum_{k=1}^{N}\sum_{t=2}^{T_k} m_i(x_{t-1}^k)}, \quad i,j=1,2,\ldots,M. \tag{16}$$

Equations (15) and (16) are referred to as fuzzy estimation in FMCM.

4.2 *FGA*

The FGA is similar to the genetic algorithm described in Sec. 3. The main difference between the fuzzy and nonfuzzy genetic algorithms is in their approach of calculating the cluster centers. The smooth parameters are used to calculate the centers of clusters in the FGA. Let the string R represent two sets of components, L_1 and L_2, which are defined as Eqs. (9) and (10). In L_1, n_1 components, B_i for $1 \le i \le n_1$, are used as the seeds to generate n_1 clusters. The membership function, $m_j(i)$, is used to measure the distances between component B_j', in L_2 and cluster C_i.

$$m_j(i) = \frac{1}{\sum_{k=1}^{n_1} (\|u_j'-S_i\|/\|u_j'-S_k\|)}, \tag{17}$$

where u_j' and S_i indicate the centers of component B_j' and cluster C_i, respectively.

Then, the new center, $\hat{S}_i$, of the cluster C_i is updated as

$$\hat{S}_i = \frac{\sum_{B_j' \subset C_i} u_j'|B_j'|m_j(i)}{\sum_{B_j' \subset C_i} |B_j'|m_j(i)}, \tag{18}$$

where $|B_j'|$ indicates the number of objects contained in the component B_j'. Using Eq. (17) above, the FGA works essential as does the genetic algorithm in Sec. 3.

5 Experiments

Two data sets, speeches and texture images, were used to test the performance of the proposed MCM based on the genetic algorithm. The proposed method was compared with other methods in the applications, speech recognition and texture image recognition.

5.1 *Speech Recognition*

The ISOLET database, using the 26 letters of the English alphabet, was used in the isolated word recognition test. The training data set consisted of 4680 utterances from 90 speakers, and the testing data set consisted of 1560 utterances from 30 speakers. The ISOLET database was sampled at 16 kHz with a 16-bit resolution. A Hamming window of 20 ms with 50% overlap was used to further process each utterance by fast Fourier transform (FFT).

Table 3 The recognition rates and the computation time of the proposed MCMs and the other HMMs.

Methods	DHMM	CHMM	FMCM(FGA)
Average recognition rates (%)	92.5	98.4	99.1
Average computation time (second)	230	940	195

Table 4 The texture recognition rates of the proposed MCMs and Ref. 11.

	Recognition Rates (%)					
Textures	MCM(KM)	MCM(GA)	MCM(FGA)	FMCM(GA)	FMCM(FGA)	Ref. 11
D3	100	100	100	100	100	—
D4	66.7	75	75	83.3	83.3	—
D6	100	100	100	100	100	—
D9	75	75	75	75	83.3	—
D11	100	100	100	100	100	—
D16	100	100	100	100	100	—
D19	58.3	66.7	75	75	75	—
D21	100	100	100	100	100	—
D24	91.7	100	100	100	100	—
D29	100	100	100	100	100	—
D34	100	100	100	100	100	—
D36	83.3	100	100	100	100	—
D52	83.3	100	100	100	100	—
D53	100	100	100	100	100	—
D55	83.3	91.7	91.7	91.7	91.7	—
D57	100	100	100	100	100	—
D65	100	100	100	100	100	—
D68	100	100	100	100	100	—
D74	100	100	100	100	100	—
D77	83.3	91.7	91.7	91.7	91.7	—
D78	100	100	100	100	100	—
D79	100	100	100	100	100	—
D82	100	100	100	100	100	—
D83	100	100	100	100	100	—
D84	100	100	100	100	100	—
D92	100	100	100	100	100	—
D95	100	100	100	100	100	—
D101	91.7	100	100	100	100	—
D104	75	100	100	100	100	—
D105	100	100	100	100	100	—
Average recognition rate	93.05	96.67	96.94	97.22	97.5	96.9

Table 1 shows the average recognition rates of 26 letters by using the proposed method and the other methods. In Table 1, MCM(KM) denote the MCMs based on the *k*-means algorithm. The proposed methods can be represented as MCM(GA), MCM(FGA), FMCM(GA), and FMCM(FGA), which combine the fuzzy and non-fuzzy genetic algorithms with fuzzy and non-fuzzy MCMs. To compare the proposed method fairly with the other methods, the genetic algorithm was first used to search the proper number of states for each MCM in MCM(GA), and then the *k*-means algorithm designed the same number of states for each MCM in MCM(KM). Table 2 shows the number of states for each MCM in MCM(GA), and shows that the number of states for each word MCM is variable, because the genetic algorithm can automatically find the proper number of states in each MCM. Additionally, MCM(KM&fixed) indicates that the number of states in each MCM is fixed. Since the average number of states for each MCM in Table 2 approximates to 22, the number of states of each MCM was set to 22 in MCM(KM&fixed). Table 1 shows that MCM(KM) outperformed MCM(KM&fixed), because the MCMs do not all have to be set to the same number of states when the MCMs are designed. Additionally, as shown in Table 1, MCM(GA) outperformed MCM(KM), because the MCMs designed by the genetic algorithm were better than those designed by the *k*-means algorithm. The genetic algorithm usually finds the near-optimal solutions, while the *k*-means algorithm obtains the local solutions. Furthermore, MCM(FGA) outperformed MCM(GA), because the fuzzy genetic algorithm generates a better clustering result than the non-fuzzy genetic algorithm. FMCM(GA) outperformed MCM(GA), indicating that the fuzzy MCM had a higher recognition performance than the non-fuzzy MCM. Finally, Table 1 shows that FMCM(FGA), the combination of the fuzzy genetic algorithm and the fuzzy MCM, is the best method among those studied.

Table 3 compares the analytical results of the proposed method with those of the HMM methods, which are seven-state left-to-right models. The proposed methods are compared with discrete HMM (DHMM) and continuous HMM (CHMM) in this experiment. The VQ level M is set to 128 in HMMs. The segmental *k*-means algorithm was run to estimate the parameters when using HMM, and the Viterbi algorithm was used in recognition. Table 3 shows that FMCM(FGA) outperformed DHMM and CHMM, and the computation time required by using FMCM(FGA) was less than that required by DHMM and CHMM.

Table 5 The texture recognition rates of the proposed MCMs and Ref. 12.

Textures	Recognition Rates (%)					
	MCM(KM)	MCM(GA)	MCM(FGA)	FMCM(GA)	FMCM(FGA)	Ref. 12
D1	91	100	100	100	100	95.34
D3	100	100	100	100	100	90.47
D6	100	100	100	100	100	93.47
D10	75	83.3	91.7	91.7	91.7	91.11
D11	100	100	100	100	100	82.35
D12	100	100	100	100	100	100
D14	75	83.3	83.3	91.7	91.7	84.09
D15	91.7	100	100	100	100	95.23
D16	100	100	100	100	100	43.90
D19	58.3	66.7	75	75	75	59.90
D20	83.3	91.7	91.7	91.7	100	91.30
D26	100	100	100	100	100	100
D37	91.7	100	100	100	100	92.85
D49	91.7	91.7	100	100	100	90.69
D52	83.3	100	100	100	100	95.23
D56	66.7	83.3	83.3	83.3	83.3	76.19
D66	91.7	91.7	91.7	91.7	100	93.18
D87	100	100	100	100	100	100
D94	83.3	100	100	100	100	88.63
D101	91.7	100	100	100	100	97.72
Average recognition rate	88.72	94.59	95.83	96.26	97.09	87.91

Table 6 The texture recognition rates of the proposed MCMs and Ref. 13.

Textures	Recognition Rates (%)					
	MCM(KM)	MCM(GA)	MCM(FGA)	FMCM(GA)	FMCM(FGA)	Ref. 13
D1	75	83.3	100	100	100	98.89
D3	100	100	100	100	100	97.78
D6	100	100	100	100	100	100
D11	100	100	100	100	100	99.44
D16	100	100	100	100	100	100
D17	100	100	100	100	100	99.89
D20	100	100	100	100	100	100
D21	100	100	100	100	100	99.72
D24	83.3	100	100	100	100	98.33
D28	75	91.7	91.7	91.7	91.7	89.71
D29	100	100	100	100	100	89.44
D32	75	91.7	100	100	100	98.89
D34	100	100	100	100	100	99.72
D35	75	91.7	100	100	100	94.72
D46	75	91.7	91.7	100	100	95.00
D47	91.7	100	100	100	100	94.44
D49	100	100	100	100	100	100
D51	91.7	100	100	100	100	97.50
D52	83.3	100	100	100	100	100
D53	100	100	100	100	100	100
D55	83.3	91.7	91.7	91.7	91.7	98.89
D56	66.7	75	75	75	83.3	75.19
D57	100	100	100	100	100	88.61
D65	100	100	100	100	100	91.39
D78	100	100	100	100	100	99.72
D82	100	100	100	100	100	90.28
D84	100	100	100	100	100	97.50
D85	91.7	100	100	100	100	98.17
D101	91.7	100	100	100	100	100
D104	75	100	100	100	100	97.50
Average recognition rate	91.11	97.22	98.34	98.61	98.89	96.34

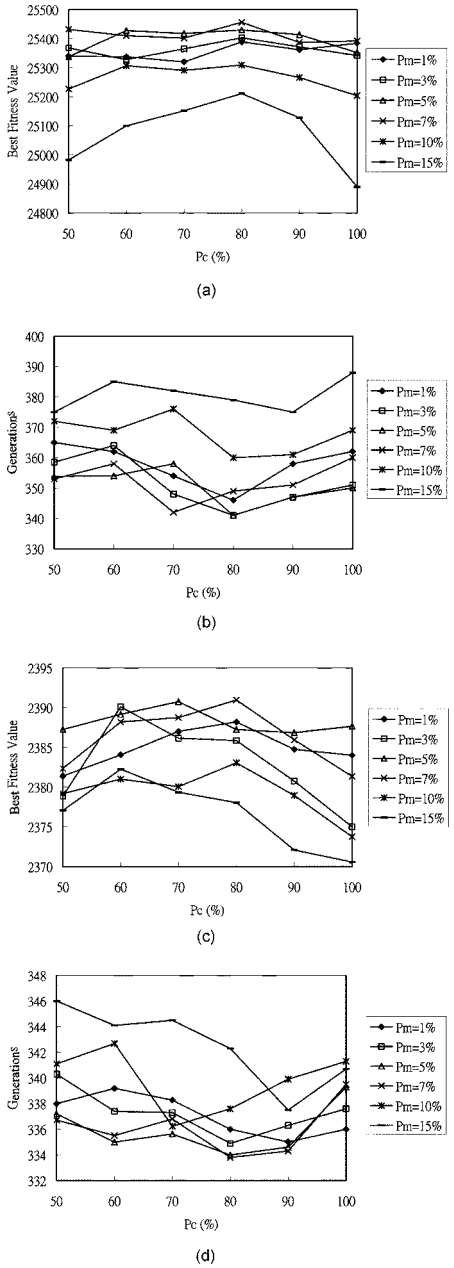

Fig. 4 Performance of the genetic algorithm using different P_c and P_m: (a) the best fitness of the string in the genetic algorithm when MCM(GA) is designed by the speech data set, (b) the number of generations required to obtain the best string in the genetic algorithm when MCM(GA) is designed by the speech data set, (c) the best fitness of the string in the genetic algorithm when MCM(GA) is designed by the texture data set, and (d) the number of generations required to obtain the best string in the genetic algorithm when MCM(GA) is designed by the texture data set.

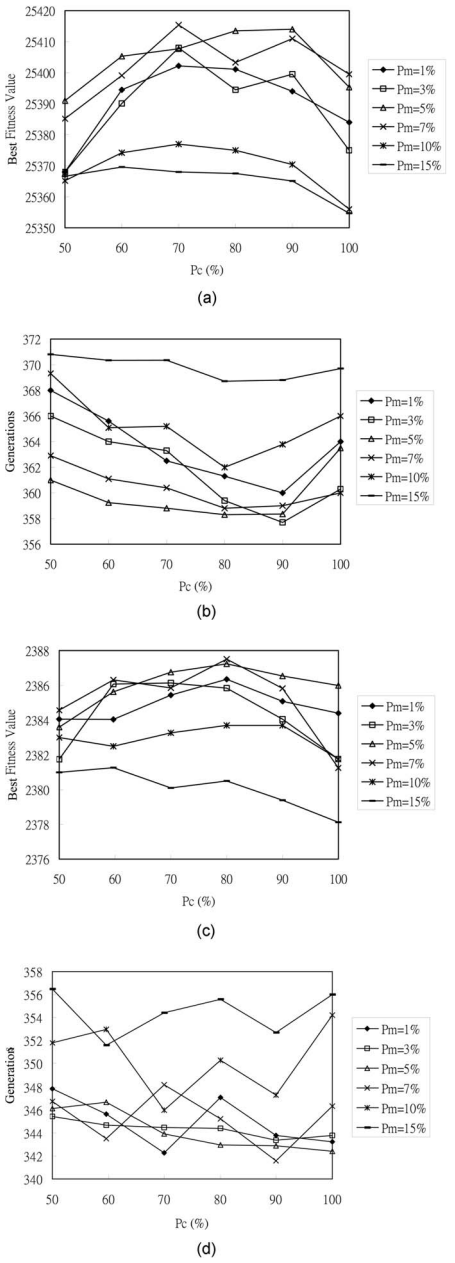

Fig. 5 Performance of the fuzzy genetic algorithm using different P_c and P_m: (a) the best fitness of the string in the fuzzy genetic algorithm when FMCM(FGA) is designed by the speech data set, (b) the number of generations required to obtain the best string in the fuzzy genetic algorithm when FMCM(FGA) is designed by the speech data set (c) the best fitness of the string in the fuzzy genetic algorithm when FMCM(FGA) is designed by the texture data set, and (d) the number of generations required to obtain the best string in the fuzzy genetic algorithm when FMCM(FGA) is designed by the texture data set.

5.2 *Texture Recognition*

The texture images obtained from Ref. 9 were used to test the proposed method in the experiments. Each texture had a size of 512×512 (pixels) and with 256 gray levels was divided into four subtextures of size 256×256 (pixels). Then, one of the four subtextures was randomly chosen as the training texture, and the other three subtextures were regarded as the testing textures. Before designing the MCM, the training textures were divided into blocks with the size of 16×16 (pixels). Each block was then transformed by a Haar wavelet transform[10] to obtain four subbands of a block. The mean values (MVs) and standard deviations (SDs) of four subbands were calculated as follows.

$$\mathrm{MV}=\frac{1}{N^2}\sum_{i,j=1}^{N} v(i,j), \tag{19}$$

$$\mathrm{SD}=\left\{\frac{1}{N^2}\sum_{i,j=1}^{N}[v(i,j)-mv]^2\right\}^{1/2}, \tag{20}$$

where N denotes the size of the subband, set to 8 in this paper, and $v(i,j)$ indicates the wavelet coefficient on the location (i,j) in the subband. Therefore, each block containing four subbands can be represented by a feature vector with eight values since two values, SD and MV for each subband.

Table 4 shows the recognition comparison between the proposed methods and Ref. 11. To compare the proposed methods fairly with the method proposed in Ref. 11, thirty texture images used in Ref. 11, which are shown in Fig. 3, were used to test our method. In Table 4, the MCM(KM) designed the same number of states for each MCM as MCM(GA). From Table 4, MCM(FGA), FMCM(GA), and FMCM(FGA) had higher average recognition rates than the recognition result proposed in Ref. 11. The symbol "—" indicates "no information" in Ref. 11. Tables 5 and 6 show that the proposed methods also outperformed the methods proposed in Refs. 12 and 13. Notably, the textures used in Tables 5 and 6 are the same as that used in Refs. 12 and 13, respectively. The reason is that the genetic algorithm proposed herein can find the proper number of states for each MCM, and hence design MCMs that achieve the near-optimal models. Additionally, the fitness function in the genetic algorithm is designed to minimize the differences between textures in the same MCM, and maximize the differences between the textures in different MCMs.

5.3 *Analysis of the Genetic Algorithms*

The proposed genetic algorithm requires the population size, crossover, and mutation rates, as input parameters when being used to design the MCMs. Setting these parameters to suitable values can enhance the performance of the genetic algorithm when the MCMs are designed. The population size is generally not sensitive to the genetic algorithm. However, if the population size is too large, then extra computation time is required to process these bit strings in the genetic algorithm. If the population size is too small, then the genetic algorithm must spend more time to search for the solution, because the insufficient information is available to find the solution when a small population is used in the genetic algorithm. In the above experiments, the population size was set to 200, which is enough to search for the near-optimal solution in the genetic algorithm.

The performance of the genetic algorithm is sensitive to two parameters, namely the crossover and mutation rates. Figure 4 shows the performance of the genetic algorithm when the MCM(GA) is designed by the speech and texture data sets. According to Figs. 4(a) and 4(c), the best fitness of string obtained is highest when $70\% \le P_c \le 90\%$ and $3\% \le P_m \le 7\%$ are given to the genetic algorithm. Notably, the best fitness with higher value is better on the plot. The genetic algorithm does not easily find the solution when $P_m \ge 10\%$, because the bit strings with high variations do not readily converge to the near-optimal solution. Figures 4(b) and 4(d) depict the minimal number of generations required to search for the best string in the genetic algorithm. Notably, the lower number of generations is better on the plot. In Figs. 4(b) and 4(d), if P_c is too small, then the genetic algorithm spends a long time searching for the solution, because the bits in a string have a small probability of being interchanged with bits in other strings. Therefore, the genetic algorithm requires more generations to find the solution. However, if P_c approaches 100%, then all of the strings should be interchanged with bits in the crossover phase of the genetic algorithm, making the solutions of the genetic algorithm process unstable. Figure 5 shows the performance of the FMCM(FGA) design by the fuzzy genetic algorithm with varying values of P_c and P_m. Figures 4 and 5 demonstrate that $70\% \le P_c \le 90\%$ and $3\% \le P_m \le 7\%$ achieve good performance in the genetic algorithms. In our experiments, $P_c = 80\%$ and $P_m = 5\%$.

6 Conclusions

We proposed a genetic algorithm to design the MCMs. The genetic algorithm can automatically determine the number of states for each MCM, which can then achieve a near-optimal model. The experimental results show that the proposed MCMs based on the genetic algorithm outperformed the traditional MCMs based on the k-means algorithm. Furthermore, smoothing parameters were used in the genetic algorithm and the MCM to further enhance the performance of the MCM. Thus, this paper also proposes the FMCM and FGA. The experimental show that the combination of FMCM and FGA can enhance the recognition rates for texture and speech.

Acknowledgments

This work was supported by the National Science Council of the Republic of China under Constraint NSC 94-2213-E-426-003.

References

1. P. Kenny, M. Lenning, and P. Mermelstein, "A linear predictive HMM for vector-valued observations with applications to speech recognition," *IEEE Trans. Acoust., Speech, Signal Process.* **38**, 220–225 (1990).
2. J. Dai, I. G. MacKenzie, and J. E. M. Tyler, "Stochastic modeling of temporal information in speech for hidden Markov models," *IEEE Trans. Acoust., Speech, Signal Process.* **2**, 102–104 (1994).
3. B. H. Juang and L. R. Rabiner, "The segmental K-means algorithm for estimating parameters of hidden Markov models," *IEEE Trans. Acoust., Speech, Signal Process.* **38**, 1639–1641 (1990).
4. J. T. Chien, "Quasi-Bayes linear regression for sequential learning of hidden Markov models," *IEEE Trans. Speech Audio Process.* **10**, 268–278 (2002).
5. J. Dai, "Isolated word recognition using Markov chain models," *IEEE Trans. Speech Audio Process.* **3**, 458–463 (1995).
6. Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.* **28**, 84–95 (1980).
7. S. Z. Selim and M. A. Ismail, "K-Means-type algorithm: generalized convergence theorem and characterization of local optimality," *IEEE Trans. Pattern Anal. Mach. Intell.* **6**, 81–87 (1984).
8. W. H. Equitz, "A new vector quantization clustering algorithm," *IEEE Trans. Acoust., Speech, Signal Process.* 1568–1575 (1989).
9. P. Borodatz, *Textures—A Photographic Album for Artists and Designers*, Dover, New York (1966).
10. R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Addison-Wesley, Boston (1992).
11. M. N. Shirazi, H. Noda, and N. Takao, "Texture classification based on Markov modeling in wavelet feature space," *Image Vis. Comput.* **18**, 967–973 (2000).
12. J. Zhang and T. Tan, "Affine invariant classification and retrieval of texture images," *Pattern Recogn.* **36**, 657–664 (2003).
13. S. Li, J. T. Kwok, H. Zhu, and Y. Wang, "Texture classification using the support vector machines," *Pattern Recogn.* **36**, 2883–2893 (2003).

Shiueng-Bien Yang received his BS and PhD degrees from the Department of Applied Mathematics, National Chung Hsing University, Taichung, Taiwan, in 1993 and 1999, respectively. He is currently an associate professor with the Department of Computer Science and Information Engineering, Leader University, Tainan City, Taiwan. His research interests include pattern recognition, speech coding, image processing, and neural networks.

You-Ping Huang received his BS degrees from the Institute of Applied Information, Leader University, Tainan City, Taiwan, in 2005. His research interests include pattern recognition and image processing.