

# Image Classification by a Two-Dimensional Hidden Markov Model

Jia Li, Amir Najmi, and Robert M. Gray, *Fellow, IEEE*

**Abstract**—For block-based classification, an image is divided into blocks, and a feature vector is formed for each block by grouping statistics extracted from the block. Conventional block-based classification algorithms decide the class of a block by examining only the feature vector of this block and ignoring context information. In order to improve classification by context, an algorithm is proposed that models images by two dimensional (2-D) hidden Markov models (HMM's). The HMM considers feature vectors statistically dependent through an underlying state process assumed to be a Markov mesh, which has transition probabilities conditioned on the states of neighboring blocks from both horizontal and vertical directions. Thus, the dependency in two dimensions is reflected simultaneously. The HMM parameters are estimated by the EM algorithm. To classify an image, the classes with maximum a posteriori probability are searched jointly for all the blocks. Applications of the HMM algorithm to document and aerial image segmentation show that the algorithm outperforms CARTM, LVQ, and Bayes VQ.

**Index Terms**—Classification, hidden Markov models, image classification.

## I. INTRODUCTION

FOR MOST block-based image classification algorithms, such as BVQ [43], images are divided into blocks, and decisions are made independently for the class of each block. This approach leads to an issue of choosing block sizes. We do not want to choose a block size too large since this obviously entails crude classification. On the other hand, if we choose a small block size, only very local properties belonging to the small block are examined in classification. The penalty then comes from losing information about surrounding regions. A well-known method in signal processing to attack this type of problem is to use context information. Trellis coding [22] in image compression provides an example. Previous work [19], [31] has looked into ways of taking advantage of context information to improve classification performance. Both block sizes and classification rules can vary according to context. The improvement achieved demonstrates the potential of context to help classification. In this paper, a two dimensional (2-D) hidden Markov model (HMM) is introduced as a general framework for context dependent classifiers.

Manuscript received November 24, 1998; revised July 4, 1999. This work was supported by the National Science Foundation under NSF Grant MIP-931190 and by gifts from Hewlett-Packard, Inc. and SK Telecom, Inc. The associate editor coordinating the review of this paper and approving it for publication was Dr. Xiang-Gen Xia.

The authors are with the the Information Systems Laboratory, Department of Electrical Engineering, Stanford University, Stanford, CA 94305 USA (e-mail: jiali@isl.stanford.edu; zaalim@leland.stanford.edu; rmgray@stanford.edu).

Publisher Item Identifier S 1053-587X(00)01015-1.

## A. One-Dimensional HMM

The theory of hidden Markov models in one dimension (1-D HMM's) was developed in the 1960s by Baum *et al.* [3]–[6]. HMM's have earned their popularity in large part from successful application to speech recognition [2], [12], [23], [40], [45]. Underlying an HMM is a basic Markov chain [33]. In fact, an HMM is simply a “Markov Ssource,” as defined by Shannon [46] and Gallager [20]: a conditionally independent process on a Markov chain or, equivalently, a Markov chain viewed through a memoryless noisy channel. Thus, at any discrete unit of time, the system is assumed to exist in one of a finite set of states. Transitions between states take place according to a fixed probability, depending only on the state of the system at the unit of time immediately preceding (one-step Markovian). In an HMM, at each unit of time, a single observation is generated from the current state according to a probability distribution, depending only on the state. Thus, in contrast to a Markov model, since the observation is a random function of the state, it is not, in general, possible to determine the current state by simply looking at the current observation. HMM's owe both their name and modeling power to the fact that these states represent abstract quantities that are themselves never observed. They correspond to “clusters” of contexts having similar probability distributions of the observation.

Suppose that there are  $M$  states  $\{1, \dots, M\}$  and that the probability of transition between states  $i$  and  $j$  is  $a_{i,j}$ . Hence, the probability that at time  $t$  the system will be in the state  $j$ , given that at time  $t - 1$  it was in state  $i$ , is  $a_{i,j}$ . Define  $u_t$  as the observation of the system at time  $t$ . This observation is generated according to a probability distribution that is dependent only on the state at time  $t$ . Let  $b_i(u_t)$  be the probability distribution of  $u_t$  in state  $i$ . If  $\pi_i$  is the probability of being in state  $i$  at time  $t = 1$ , then the likelihood of observing the sequence  $\mathbf{u} = \{u_t\}_{t=1}^T$  is evaluated by summing over all possible state sequences, that is

$$P(\mathbf{u}) = \sum_{s_1, s_2, \dots, s_T} \pi_{s_1} b_{s_1}(u_1) a_{s_1, s_2} b_{s_2}(u_2) \cdots a_{s_{T-1}, s_T} b_{s_T}(u_T)$$

where  $s_t$  represents the state at time  $t$ . For simplicity, if the meaning is clear from context, we will be sloppy with notation  $P(\cdot)$ . When the argument is continuous,  $P(\cdot)$  refers to the probability density function. In most continuous density HMM systems used for speech recognition, the density of the observation  $u_t$  in a particular state is assumed to be a Gaussian mixture distribution. Further generalization can be made by assuming

single Gaussian distributions since a state with a number of mixture components can be split into substates with single Gaussian distributions. The density of the observation  $u_t$  in state  $i$  is thus

$$b_i(u_t) = \frac{1}{\sqrt{(2\pi)^k \det(\Sigma_i)}} e^{-\frac{1}{2}(u_t - \mu_i)^t \Sigma_i^{-1} (u_t - \mu_i)}$$

where  $k$  is the dimension of  $u_t$ , and where  $\mu_i$  and  $\Sigma_i$  are the mean vector and covariance matrix, respectively.

Estimation of 1-D HMM model parameters is usually performed using the Baum-Welch algorithm [6] (which is a special case of the EM algorithm [13]), which performs maximum likelihood estimation. Let  $L_i(t)$  denote the conditional probability of being in state  $i$  at time  $t$ , given the observations, and let  $H_{i,j}(t)$  denote the conditional probability of a transition from state  $i$  at time  $t$  to state  $j$  at time  $t+1$ , given the observations. The re-estimation formulae for the means, covariances, and the transition probabilities are

$$\begin{aligned} \hat{\mu}_i &= \frac{\sum_{t=1}^T L_i(t) u_t}{\sum_{t=1}^T L_i(t)} \\ \hat{\Sigma}_i &= \frac{\sum_{t=1}^T L_i(t) (u_t - \hat{\mu}_i)(u_t - \hat{\mu}_i)^t}{\sum_{t=1}^T L_i(t)} \\ \hat{a}_{i,j} &= \frac{\sum_{t=1}^{T-1} H_{i,j}(t)}{\sum_{t=1}^T L_i(t)}. \end{aligned}$$

To apply the above estimation formulae, the probabilities  $L_i(t)$  and  $H_{i,j}(t)$  must be calculated. This is done efficiently by the so-called *forward-backward* algorithm [6]. Define the forward probability  $\alpha_i(t)$  as the joint probability of observing the first  $t$  vectors  $u_\tau, \tau = 1, \dots, t$  and being in state  $i$  at time  $t$ . This probability can be evaluated by the recursive formula

$$\begin{aligned} \alpha_i(1) &= \pi_i b_i(u_1) \quad 1 \leq i \leq M \\ \alpha_i(t) &= b_i(u_t) \sum_{j=1}^M \alpha_j(t-1) a_{j,i} \quad 1 < t \leq T, \quad 1 \leq i \leq M. \end{aligned}$$

Define the backward probability  $\beta_i(t)$  as the conditional probability of observing the vectors after time  $t, u_\tau, \tau = t+1, \dots, T$ , given that the state at time  $t$  is  $i$ . As with the forward probability, the backward probability can be evaluated using the recursion

$$\begin{aligned} \beta_i(T) &= 1 \\ \beta_i(t) &= \sum_{j=1}^M a_{i,j} b_j(u_{t+1}) \beta_j(t+1) \quad 1 \leq t < T. \end{aligned}$$

The probabilities  $L_i(t)$  and  $H_{i,j}(t)$  are solved by

$$\begin{aligned} L_i(t) &= P(s_t = i | \mathbf{u}) = \frac{P(\mathbf{u}, s_t = i)}{P(\mathbf{u})} \\ &= \frac{1}{P(\mathbf{u})} \alpha_i(t) \beta_i(t) \\ H_{i,j}(t) &= P(s_t = i, s_{t+1} = j | \mathbf{u}) \\ &= \frac{1}{P(\mathbf{u})} \alpha_i(t) a_{i,j} b_j(u_{t+1}) \beta_j(t+1). \end{aligned}$$

For details, see any of the references on speech recognition [23], [40], [45], [52].

An approximation to the maximum likelihood training provided by the Baum-Welch algorithm is what is often termed Viterbi training [52], in which each observation is assumed (with weight of 1) to have resulted from the single most likely state sequence that might have caused it. Denote the sequence of states  $\mathbf{s} = \{s_t\}_{t=1}^T$ . The state sequence with the maximum conditional probability given the observations is

$$\mathbf{s}^* = \max_{\mathbf{s}}^{-1} P(\mathbf{s} | \mathbf{u}) = \max_{\mathbf{s}}^{-1} P(\mathbf{s}, \mathbf{u}).$$

The second equality follows as  $\mathbf{u}$  is fixed for all possible state sequences. The Viterbi algorithm [48] is applied to maximize  $P(\mathbf{s}, \mathbf{u})$  since  $\max_{\mathbf{s}} P(\mathbf{s}, \mathbf{u})$  can be computed by the recursive formulae

$$\begin{aligned} \theta_i(1) &= \pi_i b_i(u_1) \quad 1 \leq i \leq M \\ \theta_i(t) &= \max_j \{\theta_j(t-1) a_{j,i}\} b_i(u_t) \quad 1 < t \leq M \\ &\quad 1 \leq i \leq M \\ \max_{\mathbf{s}} P(\mathbf{s}, \mathbf{u}) &= \max_j \theta_j(T). \end{aligned}$$

The model parameters are then estimated by

$$\begin{aligned} \hat{\mu}_i &= \frac{\sum_{t=1}^T I(s_t^* = i) u_t}{\sum_{t=1}^T I(s_t^* = i)} \\ \hat{\Sigma}_i &= \frac{\sum_{t=1}^T I(s_t^* = i) (u_t - \hat{\mu}_i)(u_t - \hat{\mu}_i)^t}{\sum_{t=1}^T I(s_t^* = i)} \\ \hat{a}_{i,j} &= \frac{\sum_{t=1}^{T-1} I(s_t^* = i) I(s_{t+1}^* = j)}{\sum_{t=1}^T I(s_t^* = i)}. \end{aligned}$$

As usual,  $I(\cdot)$  is the indicator function that equals one when the argument is true and zero otherwise. Note that the estimation formulae above differ from the Baum-Welch formulae by substitution of  $I(s_t^* = i)$  for  $L_i(t)$  and  $I(s_t^* = i) I(s_{t+1}^* = j)$  for  $H_{i,j}(t)$ . Thus, another way to view the Viterbi training is that the state sequence with the maximum *a posteriori* probability is assumed to be the real state sequence. With the real state sequence known, the probability of being in state  $i$  at time  $t, L_i(t)$  is either 1 or 0, depending on whether the real state at  $t$  equals  $i$ , i.e.,  $L_i(t) = I(s_t^* = i)$ . For the Baum-Welch algorithm, the assignment of observations to states is “soft” in the sense that each observation is assigned to each state with a weight  $L_i(t)$ . For the Viterbi training algorithm, however, the observations are uniquely assigned to the states according to the state sequence with the maximum *a posteriori* probability.

While more efficient computationally, Viterbi training does not in general result in maximum likelihood estimates. Note that an intermediate technique often used is to consider only the  $N$  most likely state sequences for each observation sequence for likelihood weighted training.

## B. Previous Work on 2-D HMM

To apply the HMM to images, previous work extended the 1-D HMM to a pseudo 2-D HMM [29], [51]. The model is “pseudo 2-D” in the sense that it is not a fully connected 2-D HMM. The basic assumption is that there exists a set of “superstates” that are Markovian. Within each superstate, there is a set of simple Markovian states. For 2-D images, the superstate

is first chosen using a first-order Markov transition probability based on the previous superstate. This superstate determines the simple Markov chain to be used by the entire row. A simple Markov chain is then used to generate observations in the row. Thus, superstates relate to rows and simple states to columns. In particular applications, this model works better than the 1-D HMM [29], but we expect the pseudo 2-D HMM to be much more effective with regular images, such as documents. Since the effect of the state of a pixel on the state below it is distributed across the whole row, the pseudo 2-D model is too constrained for normal image classification.

The effort devoted to applying a truly 2-D HMM to image segmentation was first made by Devijver [14]–[16]. Devijver proposed representing images as hidden Markov models with the state processes being Markov meshes, in particular, second- and third-order Markov meshes, the former being the focus of following sections. Applications to image segmentation, restoration, and compression were explored [16]–[18]. In [14], it was noted that the complexity of estimating the models or using them to perform maximum *a posteriori* (MAP) classification is exponential in  $w \times w$ : the size of an image. The analytic solution for estimating the models was not discussed. Instead, computationally feasible algorithms [14]–[16] were developed by making additional assumptions regarding models or using locally optimal solutions. The deterministic relaxation algorithm [14] for searching maximum *a posteriori* states [23] is worth noting. The algorithm optimizes states iteratively by making local changes to current states in such a way as to increase the likelihood of the entire image. The result depends critically on the initial states. In Section III, we derive analytic formulas for model estimation and show that computation is exponential in  $w$  by using a forward-backward-like algorithm. A suboptimal algorithm is described in Section V to achieve polynomial-time complexity.

Other work based on 2-D HMM's includes an algorithm for character recognition developed by Levin and Pieraccini [30] and an image decoding system over noisy channels constructed by Park and Miller [39]. In [39], 2-D HMM's with Markov meshes are used to model noisy channels, in which case, underlying states, corresponding to true indices transmitted by an encoder, are observable from training data. Consequently, it is straightforward to estimate the models, whereas estimation is the main difficulty for situations when states are unobservable.

### C. Outline of the Algorithm

An outline of our algorithm is as follows.

#### 1) Training

- a) Divide training images into nonoverlapping blocks with equal size and extract a feature vector for each block.
- b) Select the number of states for the 2-D HMM.
- c) Estimate model parameters based on the feature vectors and their hand-labeled classes.

#### 2) Testing

- a) Generate feature vectors (same as step 1a) for the testing image.
- b) Search for the set of classes with maximum *a posteriori* probability, given the feature vectors according to the trained 2-D HMM.

In Section II, we provide a mathematical formulation of the basic assumptions of the 2-D HMM. Section III derives the iterative estimation algorithm for the model according to the general EM algorithm. Computational complexity is analyzed in Section IV. In Section IV, backward and forward probabilities in the 2-D case are introduced to efficiently estimate the model. Our algorithm further lowers the computational complexity by using the Viterbi training. A suboptimal fast version of the Viterbi algorithm is described in Section V. Two applications of classification based on the 2-D HMM are presented in Section VI. We conclude in Section VII.

## II. ASSUMPTIONS OF THE 2-D HMM

As in all block-based classification systems, an image to be classified is divided into blocks, and feature vectors are evaluated as statistics of the blocks. The image is then classified according to the feature vectors.

The 2-D HMM assumes that the feature vectors are generated by a Markov model that may change state once every block. Suppose there are  $M$  states  $\{1, \dots, M\}$ , and the state of block  $(i, j)$  is denoted by  $s_{i,j}$ . The feature vector of block  $(i, j)$  is  $u_{i,j}$ , and the class is  $c_{i,j}$ . Denote  $(i', j') < (i, j)$  or  $(i, j) > (i', j')$  if  $i' < i$ , or  $i' = i$  and  $j' < j$ , in which case, we say that block  $(i', j')$  is before block  $(i, j)$ . For example, in the left panel of Fig. 1, the blocks before  $(i, j)$  are the shaded blocks. This sense of order is the same as the raster order of row by row. We would like to point out, however, that this order is introduced only for stating the assumptions. In classification, blocks are not classified one by one in such an order. The classification algorithm attempts to find the optimal combination of classes jointly for many blocks at once. A 1-D approach of joint classification, assuming a scanning order in classification, is usually suboptimal.

The first assumption made is that

$$P(s_{i,j} \mid s_{i',j'}, u_{i',j'} : (i', j') \in \Psi) = a_{m,n,l},$$

$$\text{where } \Psi = \{(i', j') : (i', j') < (i, j)\}$$

$$\text{and } m = s_{i-1,j}, n = s_{i,j-1}, \text{ and } l = s_{i,j}. \quad (1)$$

The above assumption can be summarized by two points. First, the state  $s_{i',j'}$  is a sufficient statistic for  $(s_{i',j'}, u_{i',j'})$  for estimating transition probabilities, i.e., the  $u$  are conditionally memoryless. Second, the state transition is first-order Markovian in a 2-D sense. The probability of the system entering a particular state depends on the state of the system at the adjacent observations in both horizontal and vertical directions. A transition from any state to any state is allowed. As shown in the left panel of Fig. 1, knowing the states of all the shaded blocks, we need only the states of the two adjacent blocks in the darker shade to calculate the transition probability to a next state. It is also assumed that there is a unique mapping from

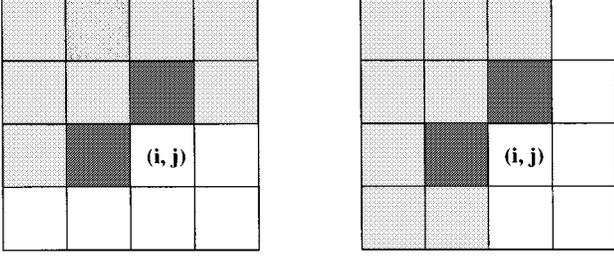


Fig. 1. Markovian property of transitions among states.

states to classes. Thus, the classes of the blocks are determined once the states are known.

The second assumption is that for every state, the feature vectors follow a Gaussian mixture distribution. Once the state of a block is known, the feature vector is conditionally independent of the other blocks. Since any state with an  $M$ -component Gaussian mixture can be split into  $M$  substates with single Gaussian distributions, the model restricts us to single Gaussian distributions. For a block with state  $s$  and feature vector  $u$ , the distribution has density

$$b_s(u) = \frac{1}{\sqrt{(2\pi)^k |\Sigma_s|}} e^{-\frac{1}{2}(u-\mu_s)^t \Sigma_s^{-1} (u-\mu_s)} \quad (2)$$

where  $\Sigma_s$  is the covariance matrix, and  $\mu_s$  is the mean vector.

The Markovian assumption on state transitions can simplify significantly the evaluation of the probability of the states, i.e.,  $P\{s_{i,j} : (i,j) \in \mathbb{N}\}$ , where  $\mathbb{N} = \{(i,j) : 0 \leq i < w, 0 \leq j < z\}$  refers to all the blocks in an image. To expand this probability efficiently by the conditional probability formula, we first prove that a rotated form of the 2-D Markovian property holds given the two assumptions. Recall the definition  $(i',j') < (i,j)$  if  $i' < i$  or  $i' = i$ , and  $j' < j$ . We then define a rotated relation of “<,” which is denoted by “ $\tilde{<}$ ,” which specifies  $(i',j') \tilde{<} (i,j)$ , or  $(i,j) \tilde{>} (i',j')$ , if  $j' < j$ , or  $j' = j$  and  $i' < i$ . An example is shown in the right panel of Fig. 1. To prove that

$$P(s_{i,j} | s_{i',j'}, u_{i',j'} : (i',j') \in \tilde{\Psi}) = a_{m,n,l}$$

$$\text{where } \tilde{\Psi} = \{(i',j') : (i',j') \tilde{<} (i,j)\}$$

$$\text{and } m = s_{i-1,j}, n = s_{i,j-1}, \text{ and } l = s_{i,j}$$

use the previous definition  $\Psi = \{(i',j') : (i',j') < (i,j)\}$  and introduce the following notation:

$$\tilde{\Psi} \cup \Psi = \{(i',j') : (i',j') < (i,j) \text{ or } (i',j') \tilde{<} (i,j)\}$$

$$\tilde{\Psi} \cap \Psi = \{(i',j') : (i',j') < (i,j) \text{ and } (i',j') \tilde{<} (i,j)\}$$

$$\tilde{\Psi} - \Psi = \{(i',j') : (i',j') \tilde{<} (i,j) \text{ and } (i',j') > (i,j)\}.$$

Note that  $\tilde{\Psi} = (\tilde{\Psi} \cap \Psi) \cup (\tilde{\Psi} - \Psi)$ . Denote  $\gamma_0 = P(s_{i',j'}, u_{i',j'} : (i',j') \in \tilde{\Psi})$  and  $\gamma_1 = P(s_{i',j'}, u_{i',j'} : (i',j') \in \tilde{\Psi} \cap \Psi)$ . We can then derive (3)–(5), shown at the bottom of the page, where  $m = s_{i-1,j}$ ,  $n = s_{i,j-1}$ , and  $l = s_{i,j}$ . Equality (3) follows from the expansion of conditional probability. Equality (4) follows from the Markovian assumption. Equality (5) holds due to both the Markovian assumption and the assumption that the feature vector of a block is conditionally independent of other blocks given its state.

From the derivation, there follows an even stronger statement, that is

$$P(s_{i,j} | s_{i',j'}, u_{i',j'} : (i',j') \in \tilde{\Psi} \cup \Psi) = P(s_{i,j} | s_{i-1,j}, s_{i,j-1}). \quad (6)$$

The reason is that in the derivation, if we change  $\tilde{\Psi} \cap \Psi$  to  $\Psi$  and  $\tilde{\Psi}$  to  $\tilde{\Psi} \cup \Psi$ , all the equalities still hold. Since (6) obviously implies the original Markovian assumption and its rotated version, we have shown the equivalence of the two assumptions

$$P(s_{i,j} | s_{i',j'}, u_{i',j'} : (i',j') \in \Psi) = P(s_{i,j} | s_{i-1,j}, s_{i,j-1})$$

and

$$P(s_{i,j} | s_{i',j'}, u_{i',j'} : (i',j') \in \tilde{\Psi} \cup \Psi) = P(s_{i,j} | s_{i-1,j}, s_{i,j-1}).$$

We point out that the underlying state process defined is a special case of a Markov random field (MRF) [21], [26], which was referred to as Markov mesh and proposed by Abend *et al.* [1], [25] for the classification of binary random patterns. The Markov mesh is called a “causal” MRF [7], [25], [44] because

$$\begin{aligned} & P(s_{i,j} | s_{i',j'}, u_{i',j'} : (i',j') \in \tilde{\Psi}) \\ &= \frac{1}{\gamma_0} \cdot P(s_{i,j}, s_{i',j'}, u_{i',j'}, s_{i'',j''}, u_{i'',j''} : (i',j') \in \tilde{\Psi} \cap \Psi, (i'',j'') \in \tilde{\Psi} - \Psi) \\ &= \frac{\gamma_1}{\gamma_0} \cdot P(s_{i,j} | s_{i',j'}, u_{i',j'} : (i',j') \in \tilde{\Psi} \cap \Psi) \\ &\quad P(s_{i'',j''}, u_{i'',j''} : (i'',j'') \in \tilde{\Psi} - \Psi | s_{i,j}, s_{i',j'}, u_{i',j'} : (i',j') \in \tilde{\Psi} \cap \Psi) \end{aligned} \quad (3)$$

$$\begin{aligned} &= \frac{\gamma_1}{\gamma_0} \cdot P(s_{i,j} | s_{i-1,j}, s_{i,j-1}) \\ &\quad P(s_{i'',j''}, u_{i'',j''} : (i'',j'') \in \tilde{\Psi} - \Psi | s_{i,j}, s_{i',j'}, u_{i',j'} : (i',j') \in \tilde{\Psi} \cap \Psi) \end{aligned} \quad (4)$$

$$\begin{aligned} &= P(s_{i,j} | s_{i-1,j}, s_{i,j-1}) \\ &\quad \frac{\gamma_1}{\gamma_0} \cdot P(s_{i'',j''}, u_{i'',j''} : (i'',j'') \in \tilde{\Psi} - \Psi | s_{i',j'}, u_{i',j'} : (i',j') \in \tilde{\Psi} \cap \Psi) \end{aligned} \quad (5)$$

$$\begin{aligned} &= P(s_{i,j} | s_{i-1,j}, s_{i,j-1}) \\ &= a_{m,n,l} \end{aligned}$$

states in condition are the states of “past”: blocks above and to the left of a current block. The causality enables the derivation of an analytic iterative algorithm to estimate an HMM and to estimate states with the maximum *a posteriori* probability.

Now, we are ready to simplify the expansion of  $P\{s_{i,j} : (i,j) \in \mathbb{N}\}$

$$P\{s_{i,j} : (i,j) \in \mathbb{N}\} = P(T_0) \cdot P(T_1 | T_0) \cdots P(T_{w+z-2} | T_{w+z-3}, T_{w+z-4}, \dots, T_0) \quad (7)$$

where  $T_i$  denotes the sequence of states for blocks on diagonal  $i$ ,  $\{s_{i,0}, s_{i-1,1}, \dots, s_{0,i}\}$ , and  $w$  and  $z$  are the number of rows and columns respectively, as shown in Fig. 2.

We next show that  $P(T_i | T_{i-1}, \dots, T_0) = P(T_i | T_{i-1})$ . Without loss of generality, suppose  $T_i = \{s_{i,0}, s_{i-1,1}, \dots, s_{0,i}\}$ ; then,  $T_{i-1} = \{s_{i-1,0}, s_{i-2,1}, \dots, s_{0,i-1}\}$ , and

$$\begin{aligned} P(T_i | T_{i-1}, \dots, T_0) &= P(s_{i,0}, s_{i-1,1}, \dots, s_{0,i} | T_{i-1}, T_{i-2}, \dots, T_0) \\ &= P(s_{i,0} | T_{i-1}, \dots, T_0) \cdot P(s_{i-1,1} | s_{i,0}, T_{i-1}, \dots, T_0) \\ &\quad \cdots P(s_{0,i} | s_{1,i-1}, \dots, s_{i,0}, T_{i-1}, \dots, T_0) \\ &= P(s_{i,0} | s_{i-1,0}) \cdot P(s_{i-1,1} | s_{i-2,1}, s_{i-1,0}) \cdots P(s_{0,i} | s_{0,i-1}). \end{aligned}$$

The last equality is obtained from (6). Since all the states  $s_{i,j}$  that appear in the conditions are in  $T_{i-1}$ , it is concluded that

$$P(T_i | T_{i-1}, \dots, T_0) = P(T_i | T_{i-1}).$$

Equation (7) simplifies to

$$P\{s_{i,j} : (i,j) \in \mathbb{N}\} = P(T_0) \cdot P(T_1 | T_0) \cdots P(T_{w+z-2} | T_{w+z-3}). \quad (8)$$

The state sequence  $T_i$  thus serves as an “isolating” element in the expansion of  $P\{s_{i,j} : (i,j) \in \mathbb{N}\}$ , which plays the role of a state at a single unit of time in the case of a 1-D Markov model. As we shall see, this property is essential for developing the algorithm. We may notice that, besides diagonals, there exist other geometric forms that can serve as “isolating” elements, for example, state sequences on rows or columns. State sequences  $T_i$  on diagonals are preferred for computational reasons that will be explained in Section V.

The task of the classifier is to estimate the 2-D HMM from training data and to classify images by finding the combination of states with the maximum *a posteriori* probability given the observed feature vectors.

### III. PARAMETER ESTIMATION

For the assumed HMM, we need to estimate the following parameters: transition probabilities  $a_{m,n,l}$ , where  $m, n, l = 1, \dots, M$ , and  $M$  is the total number of states, the mean vectors  $\mu_m$ , and the covariance matrices  $\Sigma_m$  of the Gaussian distributions,  $m = 1, \dots, M$ . We define the set  $\mathcal{M} = \{1, \dots, M\}$ . The parameters are estimated by the maximum likelihood (ML) criterion using the EM algorithm [6],

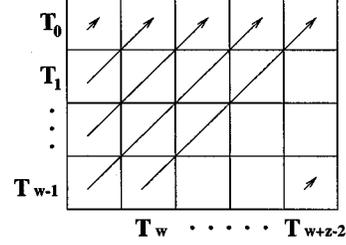


Fig. 2. Blocks on the diagonals of an image.

[13], [50]. First, the EM algorithm as described in Dempster *et al.* [13] is introduced briefly. The algorithm is then applied to the particular case to derive a specific formula.

The EM algorithm provides an iterative computation of maximum likelihood estimation when the observed data are incomplete. The term “incomplete” reflects the fact that we need to estimate the distribution of  $\mathbf{x}$  in sample space  $\mathcal{X}$ , but we can only observe  $\mathbf{x}$  indirectly through  $\mathbf{y}$  in sample space  $\mathcal{Y}$ . In many cases, there is a mapping  $\mathbf{x} \rightarrow \mathbf{y}(\mathbf{x})$  from  $\mathcal{X}$  to  $\mathcal{Y}$ , and  $\mathbf{x}$  is only known to lie in a subset of  $\mathcal{X}$ , denoted by  $\mathcal{X}(\mathbf{y})$ , which is determined by the equation  $\mathbf{y} = \mathbf{y}(\mathbf{x})$ . We postulate a family of distribution  $f(\mathbf{x} | \phi)$ , with parameters  $\phi \in \Omega$ , on  $\mathbf{x}$ . The distribution of  $\mathbf{y}$ ,  $g(\mathbf{y} | \phi)$  can be derived as

$$g(\mathbf{y} | \phi) = \int_{\mathcal{X}(\mathbf{y})} f(\mathbf{x} | \phi) d\mathbf{x}.$$

The EM algorithm aims at finding a  $\phi$  that maximizes  $g(\mathbf{y} | \phi)$  given an observed  $\mathbf{y}$ .

Before describing the algorithm, we introduce a function [13]

$$Q(\phi' | \phi) = E(\log f(\mathbf{x} | \phi') | \mathbf{y}, \phi)$$

that is, the expected value of  $\log f(\mathbf{x} | \phi')$  according to the conditional distribution of  $\mathbf{x}$  given  $\mathbf{y}$  and parameter  $\phi$ . The expectation is assumed to exist for all pairs  $(\phi', \phi)$ . In particular, it is assumed that  $f(\mathbf{x} | \phi) > 0$  for  $\phi \in \Omega$ . The EM iteration  $\phi^{(p)} \rightarrow \phi^{(p+1)}$  is defined in [13] as follows.

- 1) E-step: Compute  $Q(\phi | \phi^{(p)})$ .
- 2) M-step: Choose  $\phi^{(p+1)}$  to be a value of  $\phi \in \Omega$  that maximizes  $Q(\phi | \phi^{(p)})$ .

Define the following notation.

- 1) The set of observed feature vectors for the entire image is  $\mathbf{u} = \{u_{i,j} : (i,j) \in \mathbb{N}\}$ .
- 2) The set of states for the image is  $\mathbf{s} = \{s_{i,j} : (i,j) \in \mathbb{N}\}$ .
- 3) The set of classes for the image is  $\mathbf{c} = \{c_{i,j} : (i,j) \in \mathbb{N}\}$ .
- 4) The mapping from a state  $s_{i,j}$  to its class is  $C(s_{i,j})$ , and the set of classes mapped from states  $\mathbf{s}$  is denoted by  $C(\mathbf{s})$ .

Specific to our case, the complete data  $\mathbf{x}$  are  $\{s_{i,j}, u_{i,j} : (i,j) \in \mathbb{N}\}$ , and the incomplete data  $\mathbf{y}$  are  $\{c_{i,j}, u_{i,j} : (i,j) \in \mathbb{N}\}$ . The function  $f(\mathbf{x} | \phi')$  is

$$\begin{aligned} f(\mathbf{x} | \phi') &= P(\mathbf{s} | \phi') \cdot P(\mathbf{u} | \mathbf{s}, \phi') \\ &= P(\mathbf{s} | a'_{m,n,l} : m, n, l \in \mathcal{M}) \cdot P(\mathbf{u} | \mathbf{s}, \mu'_m, \Sigma'_m : m \in \mathcal{M}) \\ &= \prod_{(i,j) \in \mathbb{N}} a'_{s_{i-1,j}, s_{i,j-1}, s_{i,j}} \cdot \prod_{(i,j) \in \mathbb{N}} P(u_{i,j} | \mu'_{s_{i,j}}, \Sigma'_{s_{i,j}}). \end{aligned}$$

We then have

$$\log f(\mathbf{x} | \phi') = \sum_{(i,j) \in \mathbb{N}} \log a'_{s_{i-1,j}, s_{i,j-1}, s_{i,j}} + \sum_{(i,j) \in \mathbb{N}} \log P(u_{i,j} | \mu'_{s_{i,j}}, \Sigma'_{s_{i,j}}). \quad (9)$$

Given  $\mathbf{y}$ ,  $\mathbf{x}$  can only take finite number of values, corresponding to different sets of states  $\mathbf{s}$  that have classes consistent with  $\mathbf{y}$ . The distribution of  $\mathbf{x}$  is shown as the first expression at the bottom of the page, where  $\alpha$  is a normalization constant, and  $I(\cdot)$  is the obvious indicator function. From this point, we write  $P(\mathbf{x} | \mathbf{y}, \phi^{(p)})$  as  $P(\mathbf{s} | \mathbf{y}, \phi^{(p)})$ , assuming that all the  $u_{i,j}$  in  $\mathbf{x}$  are the same as those in  $\mathbf{y}$ , since otherwise, the conditional probability of  $\mathbf{x}$  given  $\mathbf{y}$  is zero.

In the M-step, we set  $\phi^{(p+1)}$  to the  $\phi'$  that maximizes (10), shown at the bottom of the page. Equation (10) follows directly from (9). The two items in (10) can be maximized separately by choosing corresponding parameters. Consider the first term in (11), shown at the bottom of the page.

Define

$$H_{m,n,l}^{(p)}(i,j) = \sum_{\mathbf{s}} I(m = s_{i-1,j}, n = s_{i,j-1}, l = s_{i,j}) P(\mathbf{s} | \mathbf{y}, \phi^{(p)})$$

as the probability of being in state  $m$  at block  $(i-1, j)$ , state  $n$  at block  $(i, j-1)$ , and state  $l$  at block  $(i, j)$ , given the ob-

served feature vectors, classes, and model  $\phi^{(p)}$ . Expression (11) becomes

$$\sum_{m,n,l \in \mathcal{M}} \log a'_{m,n,l} \sum_{(i,j) \in \mathbb{N}} H_{m,n,l}^{(p)}(i,j)$$

which is concave in  $a'_{m,n,l}$ . Therefore, to maximize (11) under the linear constraint

$$\sum_{l=1}^M a'_{m,n,l} = 1, \quad \text{for all } m, n \in \mathcal{M}$$

use a Lagrangian multiplier, and take derivatives with respect to  $a'_{m,n,l}$ . The conclusion is

$$a'_{m,n,l} \propto \sum_{(i,j) \in \mathbb{N}} H_{m,n,l}^{(p)}(i,j),$$

which in turn yields

$$a'_{m,n,l} = \frac{\sum_{(i,j) \in \mathbb{N}} H_{m,n,l}^{(p)}(i,j)}{\sum_{l=1}^M \sum_{(i,j) \in \mathbb{N}} H_{m,n,l}^{(p)}(i,j)}.$$

Next, consider the maximization of the second term in (10), shown at the bottom of the next page.

To simplify the above expression, let

$$L_m^{(p)}(i,j) = \sum_{\mathbf{s}} I(m = s_{i,j}) P(\mathbf{s} | \mathbf{y}, \phi^{(p)})$$

$$\begin{aligned} P(\mathbf{x} | \mathbf{y}, \phi^{(p)}) &= \frac{1}{\alpha} I(C(\mathbf{s}) = \mathbf{c}) \cdot P(\mathbf{s} | \phi^{(p)}) \cdot P(\mathbf{u} | \mathbf{s}, \phi^{(p)}) \\ &= \frac{1}{\alpha} I(C(\mathbf{s}) = \mathbf{c}) \cdot \prod_{(i,j) \in \mathbb{N}} a_{s_{i-1,j}, s_{i,j-1}, s_{i,j}}^{(p)} \cdot \prod_{(i,j) \in \mathbb{N}} P(u_{i,j} | \mu_{s_{i,j}}^{(p)}, \Sigma_{s_{i,j}}^{(p)}) \end{aligned}$$

$$\begin{aligned} E(\log f(\mathbf{x} | \phi') | \mathbf{y}, \phi^{(p)}) &= \frac{1}{\alpha} \sum_{\mathbf{s}} P(\mathbf{s} | \mathbf{y}, \phi^{(p)}) \cdot \sum_{(i,j) \in \mathbb{N}} \log a'_{s_{i-1,j}, s_{i,j-1}, s_{i,j}} \\ &\quad + \frac{1}{\alpha} \sum_{\mathbf{s}} P(\mathbf{s} | \mathbf{y}, \phi^{(p)}) \sum_{(i,j) \in \mathbb{N}} \log P(u_{i,j} | \mu'_{s_{i,j}}, \Sigma'_{s_{i,j}}). \end{aligned} \quad (10)$$

$$\begin{aligned} &\sum_{\mathbf{s}} P(\mathbf{s} | \mathbf{y}, \phi^{(p)}) \cdot \sum_{(i,j) \in \mathbb{N}} \log a'_{s_{i-1,j}, s_{i,j-1}, s_{i,j}} \\ &= \sum_{\mathbf{s}} P(\mathbf{s} | \mathbf{y}, \phi^{(p)}) \cdot \sum_{m,n,l \in \mathcal{M}} \sum_{(i,j) \in \mathbb{N}} \log a'_{m,n,l} \cdot I(m = s_{i-1,j}, n = s_{i,j-1}, l = s_{i,j}) \\ &= \sum_{m,n,l \in \mathcal{M}} \log a'_{m,n,l} \sum_{(i,j) \in \mathbb{N}} \sum_{\mathbf{s}} P(\mathbf{s} | \mathbf{y}, \phi^{(p)}) I(m = s_{i-1,j}, n = s_{i,j-1}, l = s_{i,j}). \end{aligned} \quad (11)$$

which is the probability of being in state  $m$  at block  $(i, j)$ , given the observed feature vectors, classes, and model  $\phi^{(p)}$ . The above expression is then

$$\sum_{m=1}^M \sum_{(i,j) \in \mathbb{N}} L_m^{(p)}(i, j) \log P(u_{i,j} | \mu'_m, \Sigma'_m).$$

It is known that for Gaussian distributions, the ML estimate of  $\mu'_m$  is the sample average of the data, and the ML estimate of  $\Sigma'_m$  is the sample covariance matrix of the data [8]. Since, in our case, the data are weighted by  $L_m^{(p)}(i, j)$ , the ML estimate of  $\mu'_m$  and  $\Sigma'_m$  are

$$\begin{aligned} \mu'_m &= \frac{\sum_{i,j} L_m^{(p)}(i, j) u_{i,j}}{\sum_{i,j} L_m^{(p)}(i, j)}, \\ \Sigma'_m &= \frac{\sum_{i,j} L_m^{(p)}(i, j) (u_{i,j} - \mu'_m)(u_{i,j} - \mu'_m)^t}{\sum_{i,j} L_m^{(p)}(i, j)}. \end{aligned}$$

In summary, the estimation algorithm iteratively improves the model estimation by the following two steps.

- 1) Given the current model estimation  $\phi^{(p)}$ , the observed feature vectors  $u_{i,j}$ , and classes  $c_{i,j}$ , the mean vectors and covariance matrices are updated by

$$\mu_m^{(p+1)} = \frac{\sum_{i,j} L_m^{(p)}(i, j) u_{i,j}}{\sum_{i,j} L_m^{(p)}(i, j)} \quad (12)$$

and

$$\Sigma_m^{(p+1)} = \frac{\sum_{i,j} L_m^{(p)}(i, j) (u_{i,j} - \mu_m^{(p+1)})(u_{i,j} - \mu_m^{(p+1)})^t}{\sum_{i,j} L_m^{(p)}(i, j)}. \quad (13)$$

The probability  $L_m^{(p)}(i, j)$  is calculated by

$$\begin{aligned} L_m^{(p)}(i, j) &= \sum_{\mathbf{s}} I(m = s_{i,j}) \cdot \frac{1}{\alpha} I(C(\mathbf{s}) = \mathbf{c}) \\ &\cdot \prod_{(i,j) \in \mathbb{N}} a_{s_{i-1,j}, s_{i,j-1}, s_{i,j}}^{(p)} \cdot \prod_{(i,j) \in \mathbb{N}} P(u_{i,j} | \mu_{s_{i,j}}^{(p)}, \Sigma_{s_{i,j}}^{(p)}). \end{aligned} \quad (14)$$

- 2) The transition probabilities are updated by

$$a_{m,n,l}^{(p+1)} = \frac{\sum_{i,j} H_{m,n,l}^{(p)}(i, j)}{\sum_{l=1}^M \sum_{i,j} H_{m,n,l}^{(p)}(i, j)}$$

where  $H_{m,n,l}^{(p)}(i, j)$  is calculated by

$$\begin{aligned} H_{m,n,l}^{(p)}(i, j) &= \sum_{\mathbf{s}} I(m = s_{i-1,j}, n = s_{i,j-1}, l = s_{i,j}) \cdot \frac{1}{\alpha} I(C(\mathbf{s}) = \mathbf{c}) \\ &\cdot \prod_{(i,j) \in \mathbb{N}} a_{s_{i-1,j}, s_{i,j-1}, s_{i,j}}^{(p)} \\ &\cdot \prod_{(i,j) \in \mathbb{N}} P(u_{i,j} | \mu_{s_{i,j}}^{(p)}, \Sigma_{s_{i,j}}^{(p)}). \end{aligned} \quad (15)$$

The iterative algorithm starts by setting an initial state for each feature vector. For every class, feature vectors labeled as this class are sequenced in a raster order; and the states corresponding to this class are assigned in a round-robin way to those vectors. In the initial step, since the initial states are assumed to be true,  $L_m^{(0)}(i, j)$  and  $H_{m,n,l}^{(0)}(i, j)$  are computed simply by

$$\begin{aligned} L_m^{(0)}(i, j) &= I(m = s_{i,j}^{(0)}) \\ H_{m,n,l}^{(0)}(i, j) &= I(m = s_{i-1,j}^{(0)}, n = s_{i,j-1}^{(0)}, l = s_{i,j}^{(0)}) \end{aligned}$$

where  $s_{i,j}^{(0)}$  denotes the initial states.

In the case of a 1-D HMM as used in speech recognition, the forward-backward algorithm is applied to calculate  $L_m(k)$  and  $H_{m,n,l}(k)$  [52] efficiently. For a 2-D HMM, however, the computation of  $L_m(i, j)$  and  $H_{m,n,l}(i, j)$  is not feasible in view of the 2-D transition probabilities. In the next section, we discuss why this is so and how to reduce the computational complexity.

#### IV. COMPUTATIONAL COMPLEXITY

As is shown in previous section, the calculation of the probabilities  $H_{m,n,l}^{(p)}(i, j)$  and  $L_m^{(p)}(i, j)$  is the key for the iterative estimation of the model parameters. If we compute  $L_m^{(p)}(i, j)$  and  $H_{m,n,l}^{(p)}(i, j)$  directly according to (14) and (15), we need to consider all the combinations of states that yield the same classes as those in the training set. The large number of such combinations of states results in an infeasible computation. Let us take  $L_m^{(p)}(i, j)$  as an example. Suppose there are  $M_0$  states for each class and that the number of blocks in an image is  $w \times z$ , as previously assumed. Then, the number of admissible combinations of states that satisfy  $C(\mathbf{s}) = \mathbf{c}$  and  $s_{i,j} = m$  is  $M_0^{(w \times z - 1)}$ . When applying the HMM algorithm, although one image is often divided into many sub-images such that  $w$ , or  $z$ , is the number of blocks in one column, or one row, in a

$$\begin{aligned} &\sum_{\mathbf{s}} P(\mathbf{s} | \mathbf{y}, \phi^{(p)}) \cdot \sum_{(i,j) \in \mathbb{N}} \log P(u_{i,j} | \mu'_{s_{i,j}}, \Sigma'_{s_{i,j}}) \\ &= \sum_{\mathbf{s}} P(\mathbf{s} | \mathbf{y}, \phi^{(p)}) \cdot \sum_{m=1}^M \sum_{(i,j) \in \mathbb{N}} \log P(u_{i,j} | \mu'_m, \Sigma'_m) I(m = s_{i,j}) \\ &= \sum_{m=1}^M \sum_{(i,j) \in \mathbb{N}} \sum_{\mathbf{s}} I(m = s_{i,j}) P(\mathbf{s} | \mathbf{y}, \phi^{(p)}) \cdot \log P(u_{i,j} | \mu'_m, \Sigma'_m). \end{aligned}$$

subimage, we need to keep  $w$  and  $z$  sufficiently large to ensure that an adequate amount of context information is incorporated in classification. In the limit, if  $w = z = 1$ , the algorithm is simply a parametric classification algorithm performed independently on each block. It is normal to have  $w = z = 8$ . In this case, if there are four states for each class, the number of the combinations of states is  $M_0^{(w \times z - 1)} = 63^4$ , which is prohibitive for a straightforward calculation of  $L_m^{(p)}(i, j)$ . A similar difficulty occurs when estimating a 1-D HMM. The problem is solved by a recursive calculation of forward and backward probabilities [52].

The idea of using forward and backward probabilities can be extended to the 2-D HMM to simplify the computation. Recall (8) in Section II:

$$P\{s_{i,j} : (i, j) \in \mathbb{N}\} = P(T_0) \cdot P(T_1 | T_0) \cdots P(T_{w+z-2} | T_{w+z-3}).$$

The fact that the state sequence  $T_i$  on a diagonal is an ‘‘isolating’’ element in the expansion of  $P\{s_{i,j} : (i, j) \in \mathbb{N}\}$  enables us to define the forward and backward probabilities and to evaluate them by recursive formulas.

Let us clarify notation first. In addition to the notation provided in the list in Section III, we need the following definitions.

- 1) The diagonal on which block  $(i, j)$  lies is denoted by  $\Delta(i, j)$ .
- 2) The feature vectors on diagonal  $d$ ,  $\{u_{i,j} : \Delta(i, j) = d\}$  is denoted by  $\mathbf{u}(d)$ .
- 3) The state sequence on diagonal  $d$ ,  $\{s_{i,j} : \Delta(i, j) = d\}$  is denoted by  $\mathbf{s}(d)$ .
- 4) For a state sequence  $T$  on diagonal  $d$ , its value at block  $(i, j)$  is  $T(i, j)$ .

The forward probability  $\theta_T(d)$  for some model  $\mathbf{M}$  is defined as

$$\theta_T(d) = P\{\mathbf{s}(d) = T, \mathbf{u}(\tau) : \tau \leq d \mid \mathbf{M}\}.$$

The forward probability  $\theta_T(d)$  is the probability of observing the vectors lying on or above diagonal  $d$  and having state sequence  $T$  for blocks on diagonal  $d$ .

The backward probability  $\beta_T(d)$  is defined as

$$\beta_T(d) = P\{\mathbf{u}(\tau) : \tau > d \mid \mathbf{s}(d) = T, \mathbf{M}\}$$

that is,  $\beta_T(d)$  is the conditional probability of observing the vectors lying below diagonal  $d$  given the state sequence on diagonal  $d$  is  $T$ .

Similar to the case of 1-D HMM, we can derive recursive formulas for calculating  $\theta_T(d)$  and  $\beta_T(d)$ , which are listed below.

$$\theta_{T_d}(d) = \sum_{T_{d-1}} \theta_{T_{d-1}}(d-1)$$

$$\cdot P(T_d | T_{d-1}, \mathbf{M}) \cdot P(\mathbf{u}(d) | T_d, \mathbf{M}) \quad (16)$$

$$\beta_{T_d}(d) = \sum_{T_{d+1}} P(T_{d+1} | T_d, \mathbf{M})$$

$$\cdot P(\mathbf{u}(d+1) | T_{d+1}, \mathbf{M}) \cdot \beta_{T_{d+1}}(d+1). \quad (17)$$

We can then compute  $L_m(i, j)$  given model  $\mathbf{M}$  by

$$L_m(i, j) = P(s_{i,j} = m \mid \mathbf{u}, \mathbf{c}, \mathbf{M}) = \begin{cases} \sum_{T_d: T_d(i,j)=m} P(T_d \mid \mathbf{u}, \mathbf{c}, \mathbf{M}), & C(m) = c_{i,j} \\ 0, & \text{otherwise.} \end{cases}$$

Consider the case  $C(m) = c_{i,j}$ . It is assumed in the derivation below that the summation over  $T_d$  only covers  $T_d$ , which yields consistent classes with the training data.

$$L_m(i, j) = \sum_{T_d: T_d(i,j)=m} \frac{P(T_d, \mathbf{u} \mid \mathbf{M})}{P(\mathbf{u}, \mathbf{c} \mid \mathbf{M})} = \sum_{T_d: T_d(i,j)=m} \frac{\theta_{T_d}(\Delta(i, j)) \cdot \beta_{T_d}(\Delta(i, j))}{P(\mathbf{u}, \mathbf{c} \mid \mathbf{M})}. \quad (18)$$

The subscript ‘‘ $d$ ’’ in  $T_d$  denotes the diagonal  $d$  of block  $(i, j)$ . In the calculation of  $H_{m,n,l}(i, j)$ , shown at the bottom of the page, the summations are always over state sequences with the same classes as those in the training data. We then consider the case  $C(m) = c_{i-1,j}$ ,  $C(n) = c_{i,j-1}$ , and  $C(l) = c_{i,j}$ . In (19), the summations over  $T_d$  and  $T_{d-1}$  are constrained additionally to  $T_d$  satisfying  $T_d(i, j) = l$  and  $T_{d-1}$  satisfying  $T_{d-1}(i-1, j) = m$ ,  $T_{d-1}(i, j-1) = n$ .

$$H_{m,n,l}(i, j) = \sum_{T_d} \sum_{T_{d-1}} \frac{\theta_{T_{d-1}}(\Delta(i, j) - 1)}{P(\mathbf{u}, \mathbf{c} \mid \mathbf{M})} \cdot [P(T_d | T_{d-1}, \mathbf{M}) P(\mathbf{u}(d) | T_d, \mathbf{M}) \cdot \beta_{T_d}(\Delta(i, j))]. \quad (19)$$

Although using the forward and backward probabilities significantly reduces the computation for  $L_m(i, j)$  and  $H_{m,n,l}(i, j)$ , computational complexity is still high due to the 2-D aspects. Equations (16) and (17) for evaluating the forward and backward probabilities are summations over all state sequences on diagonal  $d-1$ , or  $d+1$ , with classes consistent with the training data. With the increase of blocks on a diagonal, the number of state sequences increases exponentially. The same problem occurs with calculating  $L_m(i, j)$  and  $H_{m,n,l}(i, j)$ . Consequently, an approximation is made in the calculation of  $L_m(i, j)$  and  $H_{m,n,l}(i, j)$  to avoid computing the backward and forward probabilities. Recall the definitions in Section III.

$$H_{m,n,l}^{(p)}(i, j) = \sum_{\mathbf{s}} I(m = s_{i-1,j}, n = s_{i,j-1}, l = s_{i,j}) P(\mathbf{s} \mid \mathbf{y}, \phi^{(p)})$$

$$L_m^{(p)}(i, j) = \sum_{\mathbf{s}} I(m = s_{i,j}) P(\mathbf{s} \mid \mathbf{y}, \phi^{(p)}).$$

$$H_{m,n,l}(i, j) = P(s_{i-1,j} = m, s_{i,j-1} = n, s_{i,j} = l \mid \mathbf{u}, \mathbf{c}, \mathbf{M}) = \begin{cases} \sum_{T_d} \sum_{T_{d-1}} P(T_d, T_{d-1} \mid \mathbf{u}, \mathbf{c}, \mathbf{M}), & C(m) = c_{i-1,j}, C(n) = c_{i,j-1}, C(l) = c_{i,j} \\ 0, & \text{otherwise.} \end{cases}$$

To simplify the calculation of  $L_m(i, j)$  and  $H_{m,n,l}(i, j)$ , it is assumed that the single most likely state sequence accounts for virtually all the likelihood of the observations. We thus aim at finding the optimal state sequence that maximizes  $P(\mathbf{s} \mid \mathbf{y}, \phi^{(P)})$ , which is accomplished by the Viterbi training algorithm.

### V. VARIABLE-STATE VITERBI ALGORITHM

Using the Viterbi algorithm to maximize  $P(\mathbf{s} \mid \mathbf{y})$  is equivalent to maximizing  $P\{s_{i,j}, u_{i,j} : (i, j) \in \mathbb{N}\}$  constrained to  $C(s_{i,j}) = c_{i,j}$  during training. When we apply the trained model to classify images (testing process), we also aim at finding states  $\{s_{i,j} : (i, j) \in \mathbb{N}\}$  maximizing  $P\{s_{i,j}, u_{i,j} : (i, j) \in \mathbb{N}\}$  (MAP rule). The states are then mapped into classes. In testing, since  $c_{i,j}$  is to be decided, the previous constraint that  $C(s_{i,j}) = c_{i,j}$  is removed.

In the discussion, the unconstrained (testing) case is considered since in the constrained case, the only difference is to shrink the search range of  $s_{i,j}$  to states corresponding to class  $c_{i,j}$ . Expand  $P\{s_{i,j}, u_{i,j} : (i, j) \in \mathbb{N}\}$  as in (20), shown at the bottom of the page, where  $T_d$  denotes the sequence of states for blocks lying on diagonal  $d$ . The last equality comes from (7).

Since  $T_d$  serves as an ‘‘isolating’’ element in the expansion of  $P\{s_{i,j} : (i, j) \in \mathbb{N}\}$ , the Viterbi algorithm can be applied straightforwardly to find the combination of states maximizing the likelihood  $P\{s_{i,j}, u_{i,j} : (i, j) \in \mathbb{N}\}$ . The difference from the normal Viterbi algorithm is that the number of possible sequences of states at every position in the Viterbi transition diagram increases exponentially with the increase of blocks in  $T_d$ . If there are  $M$  states, the amount of computation and memory are both in the order of  $M^\nu$ , where  $\nu$  is the number of states in  $T_d$ . Fig. 3 shows an example. Hence, this version of the Viterbi algorithm is referred to as a variable-state Viterbi algorithm.

The fact that in the 2-D case, only a sequence of states on a diagonal, rather than a single block, can serve as an ‘‘isolating’’ element in the expansion of  $P\{s_{i,j} : (i, j) \in \mathbb{N}\}$  causes computational infeasibility for the variable-state Viterbi algorithm. To reduce computation, at every position of the Viterbi transition diagram, the algorithm only uses  $N$  out of all the  $M^\nu$  sequences of states, as shown in Fig. 4. The paths are constrained to pass one of these  $N$  nodes. To choose the  $N$  sequences of states, the algorithm separates the blocks in the diagonal from the other blocks by ignoring their statistical dependency. Consequently, the posterior probability of a sequence of states on the diagonal is evaluated as a product of the posterior proba-

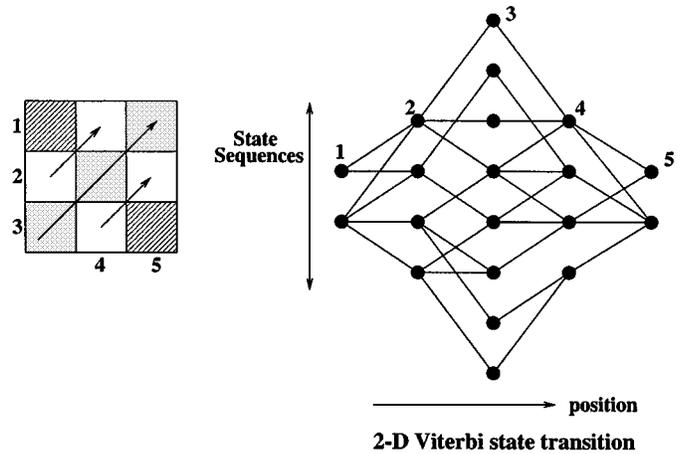


Fig. 3. Variable-state Viterbi algorithm.

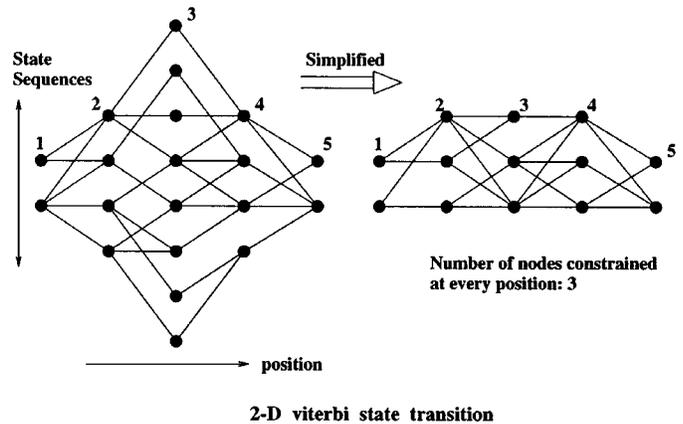


Fig. 4. Path-constrained Viterbi algorithm.

bility of every block. Then, the  $N$  sequences with the largest posterior probabilities are chosen as the  $N$  nodes allowed in the Viterbi transition diagram. The implicit assumption in doing this is that the optimal state sequence (the node in the optimal path of the Viterbi transition diagram) yields high likelihood when the blocks are treated independently. It is also expected that when the optimal state sequence is not among the  $N$  nodes, the chosen suboptimal state sequence coincides with the optimal sequence at most of the blocks. The suboptimal version of the algorithm is referred to as the path-constrained variable-state Viterbi algorithm. This algorithm is different from the  $M$ -algorithm introduced for source coding by Jelinek and Anderson [24] since

$$\begin{aligned}
 & P\{s_{i,j}, u_{i,j} : (i, j) \in \mathbb{N}\} \\
 &= P\{s_{i,j} : (i, j) \in \mathbb{N}\} \cdot P\{u_{i,j} : (i, j) \in \mathbb{N} \mid s_{i,j} : (i, j) \in \mathbb{N}\} \\
 &= P\{s_{i,j} : (i, j) \in \mathbb{N}\} \cdot \prod_{(i,j) \in \mathbb{N}} P(u_{i,j} \mid s_{i,j}) \\
 &= P(T_0) \cdot P(T_1 \mid T_0) \cdot P(T_2 \mid T_1) \cdots P(T_{w+z-2} \mid T_{w+z-3}) \prod_{(i,j) \in \mathbb{N}} P(u_{i,j} \mid s_{i,j}), \tag{20}
 \end{aligned}$$

the  $N$  nodes are preselected to avoid calculating the posterior probabilities of all the  $M^\nu$  state sequences.

As mentioned in Section II, state sequences on rows or columns can also serve as “isolating” elements in the expansion of  $P\{s_{i,j} : (i,j) \in \mathbb{N}\}$ . Diagonals are chosen for the expansion because intuition suggests that the preselection of  $N$  nodes by ignoring dependence among states on a diagonal degrades performance less than would doing the same for a row or a column. Remember that blocks on a diagonal are not geometrically as close as blocks on a row or a column.

A fast algorithm is developed for choosing such  $N$  sequences of states. It is not necessary to calculate the posterior probabilities of all the  $M^\nu$  sequences in order to choose the largest  $N$  from them. In the following discussion, we consider the maximization of the joint log likelihood of states and feature vectors since maximizing the posterior probability of the states given the feature vectors is equivalent to maximizing the joint log likelihood. In addition, note that the log likelihood of a sequence of states is equal to the sum of the log likelihoods of the individual states because we ignore context information in the preselection of nodes. Suppose there are  $\nu$  blocks on a diagonal, and each block exists in one of  $M$  states. The log likelihood of block  $i$  being in state  $m$  is  $\gamma_{i,m}$ . The preselection of the  $N$  nodes is simply to find  $N$  state sequences  $\{s_i : i = 1, \dots, \nu\}$  with the largest  $\sum_{i=1}^{\nu} \gamma_{i,s_i}$ . Suppose we want to find the state sequence  $\max_{s_i: i=1, \dots, \nu} \sum_{i=1}^{\nu} \gamma_{i,s_i}$ ; it is unnecessary to calculate  $\sum_{i=1}^{\nu} \gamma_{i,s_i}$  for all the  $M^\nu$  state sequences. We need only to find  $\max_{s_i}^{-1} \gamma_{i,s_i}$  for each  $i$ . Then, the optimal state sequence is  $\{\max_{s_i}^{-1} \gamma_{i,s_i} : i = 1, \dots, \nu\}$ . The idea can be extended for finding the  $N$  sequences with the largest log likelihood.

To ensure that the path-constrained variable-state Viterbi algorithm yields results sufficiently close to the variable-state Viterbi algorithm, the parameter  $N$  should be larger when there are more blocks in the 2-D Markov chain. As a result, an image is usually divided into subimages to avoid too many blocks in one chain. Every subimage is assumed to be a 2-D Markov chain, but the dependence between subimages is ignored. On the other hand, to incorporate any preassigned amount of context information for classification, the subimages must contain sufficiently many blocks. The selection of the parameters will be discussed in the section on experiments.

## VI. APPLICATIONS

### A. Intra- and Inter-block Features

Choosing features is a critical issue in classification because features often set the limits of classification performance. For a classifier based on the 2-D HMM, both intra-block features and inter-block features are used. The intra-block features are defined according to the pixel intensities in a block. They aim at describing the statistical properties of the block. Features selected vary greatly for different applications. Widely used examples include moments in the spatial domain or frequency domain and coefficients of transformations, e.g., the discrete cosine transform (DCT).

The inter-block features are defined to represent relations between two blocks, for example, the difference between the average intensities of the two blocks. The use of the inter-block

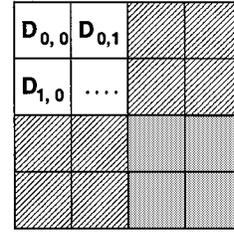


Fig. 5. DCT coefficients of a  $4 \times 4$  image block.

TABLE I  
COMPARISON OF CLASSIFICATION  
PERFORMANCE.

Algorithm	sensitivity	specificity	PVP	$P_c$
2-D HMM	0.7795	0.8203	0.8381	0.1880
CART 1	0.8528	0.7126	0.7530	0.2158
CART 2	0.8097	0.7340	0.7505	0.2408
LVQ1	0.8187	0.7419	0.7691	0.2183

features is similar to that of delta and acceleration coefficients in speech recognition, in which there is ample empirical justification for the inclusion of these features [52]. The motivation for us to use inter-block features is to compensate for the strictness of the 2-D HMM. The 2-D HMM assumes constant state transition probabilities. In practice, however, we expect that a transition to a state may depend on some mutual properties of two blocks. For instance, if the two blocks have close intensities, then they may be more likely to be in the same state. Since it is too complicated to estimate models with transition probabilities being functions, we preserve the constant transition probabilities and offset this assumption somewhat by incorporating the mutual properties into feature vectors in such a way that they can influence the determination of states through posterior probabilities. In the 2-D HMM, since the states of adjacent blocks right above or to the left of a block determine the transition probability to a new state, mutual properties between the current block and these two neighboring blocks are used as inter-block features.

### B. Aerial Image Segmentation

1) *Features*: The first application of the 2-D HMM algorithm is the segmentation into man-made and natural regions of aerial images. The images are  $512 \times 512$  gray-scale images with 8 bits per-pixel (b/pixel). They are the aerial images of the San Francisco Bay area provided by TRW (formerly ESL, Inc.) [35]. The data set used contains six images, whose hand-labeled segmented images are used as the truth set of classes. The six images and their hand-labeled classes are shown in Fig. 6.

The images were divided into  $4 \times 4$  blocks, and DCT coefficients or averages over some of them were used as features. There are six such features. The reason to use DCT coefficients is that the different energy distributions in the frequency domain distinguish the two classes better. Denote the DCT coefficients for a  $4 \times 4$  block by  $\{D_{i,j} : i, j \in (0, 1, 2, 3)\}$ , shown by Fig. 5. The definitions of the six features are the following.

- 1)  $f_1 = D_{0,0}$ ;  $f_2 = |D_{1,0}|$ ;  $f_3 = |D_{0,1}|$ ;
- 2)  $f_4 = \sum_{i=2}^3 \sum_{j=0}^1 |D_{i,j}|/4$ ;

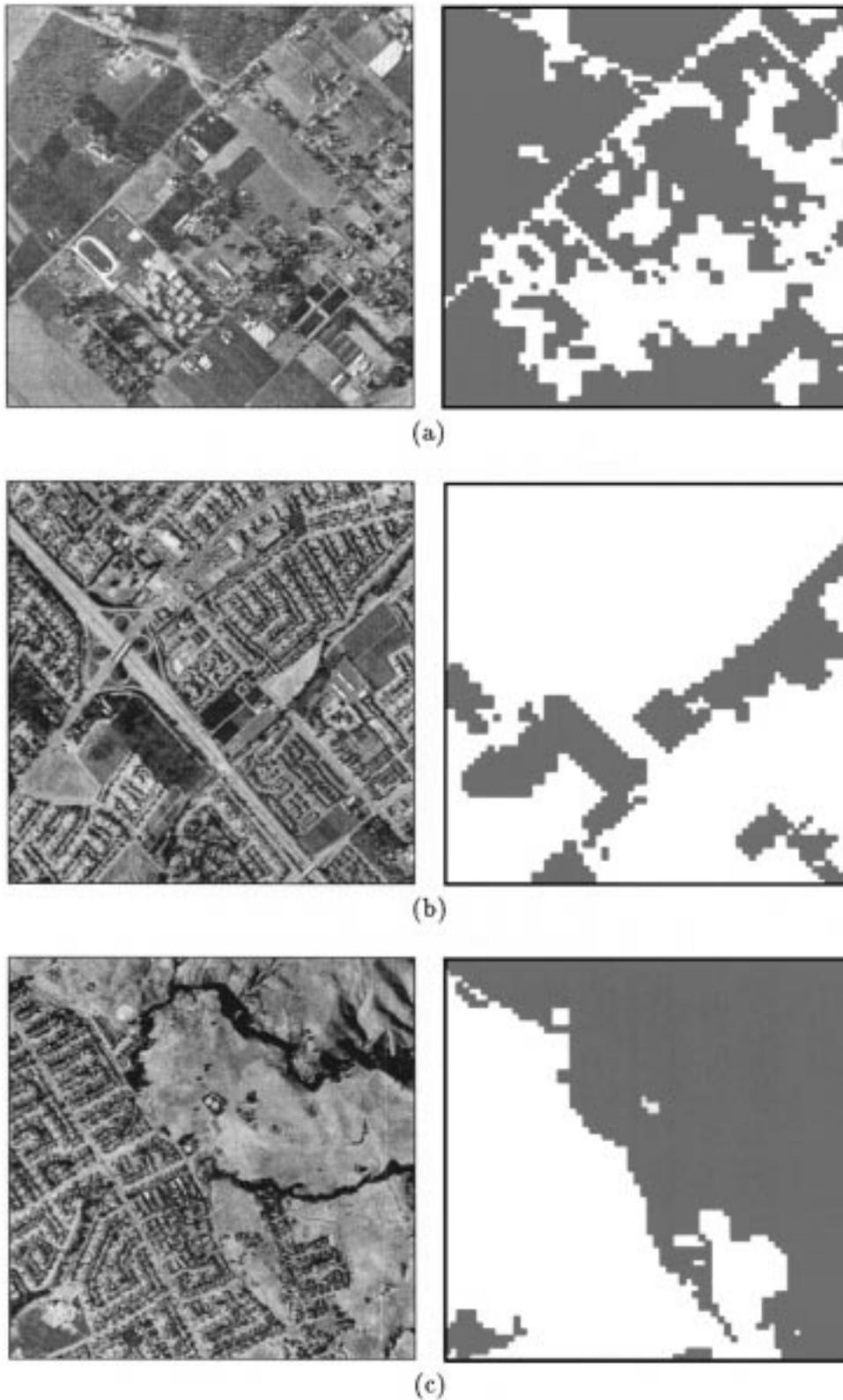


Fig. 6. Aerial images: (a)–(f) Image 1–6. Left: Original 8 b/pixel images. Right: Hand-labeled classified images. White: man-made. Gray: natural.

$$3) f_5 = \sum_{i=0}^1 \sum_{j=2}^3 |D_{i,j}|/4;$$

$$4) f_6 = \sum_{i=2}^3 \sum_{j=2}^3 |D_{i,j}|/4.$$

In addition, the spatial derivatives of the average intensity values of blocks were used as inter-block features. In particular, the spatial derivative refers to the difference between the average

intensity of a block and that of the block's upper neighbor or left neighbor.

2) *Results:* Six-fold cross-validation [47] was used to evaluate algorithms. For each iteration, one image was used as test data, and the other five were used as training data. Performance

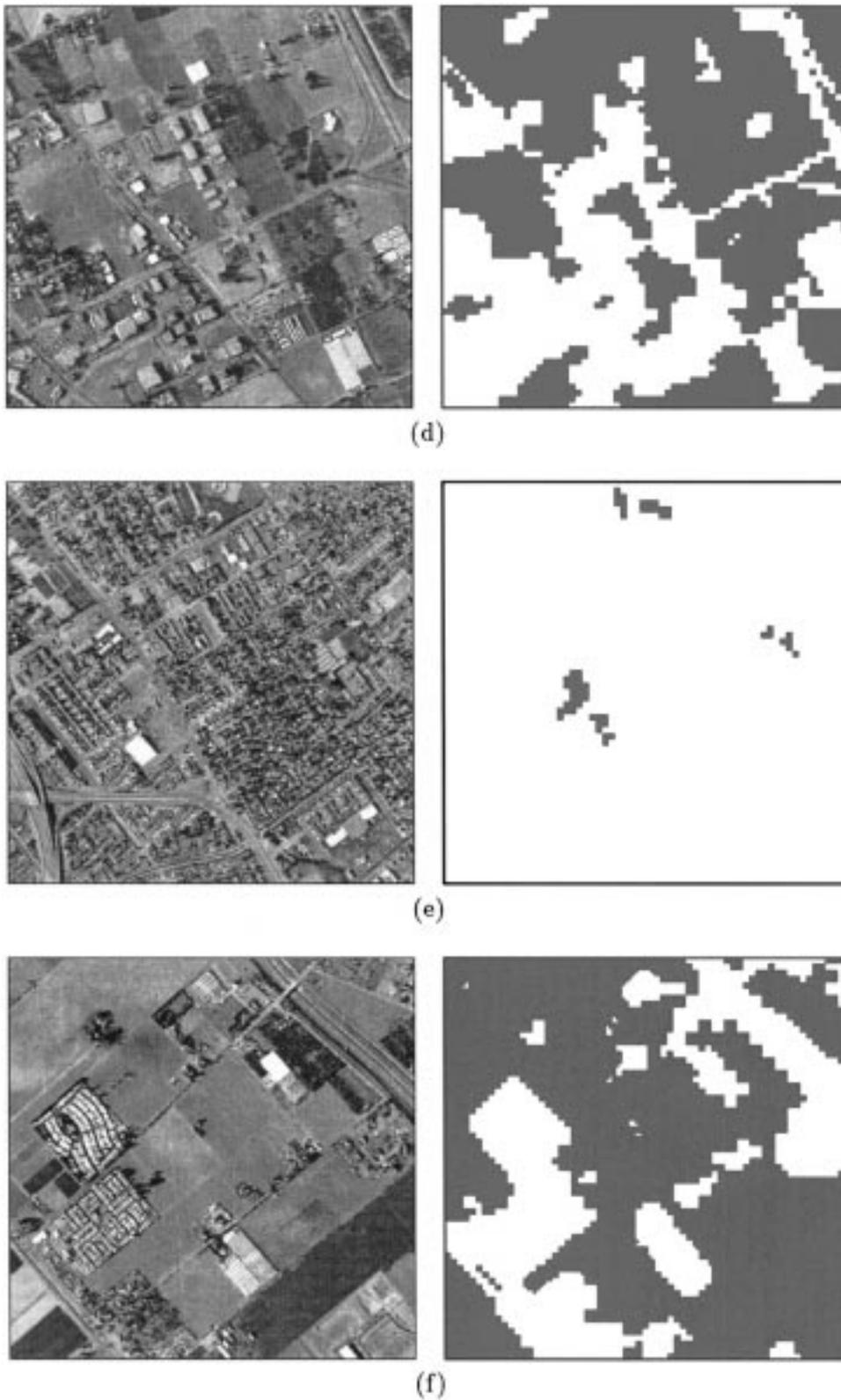


Fig. 6. (Continued.) Aerial images: (a)–(f) Image 1–6. Left: Original 8 b/pixel images. Right: Hand-labeled classified images. White: man-made. Gray: natural.

is evaluated by averaging over all the iterations. Hidden Markov models with different number of states were trained and tested. Experiments show that models with four to six states for the natural class, and seven to ten states for the man-made class yield very similar results. For the result to be given in this section, a

model with five states for the natural class and nine states for the man-made class was used. Setting too many states for each class results in worse classification for two reasons: the model closest to the truth may not be so sophisticated; and more complicated models require a larger training set. With a fixed training set, the

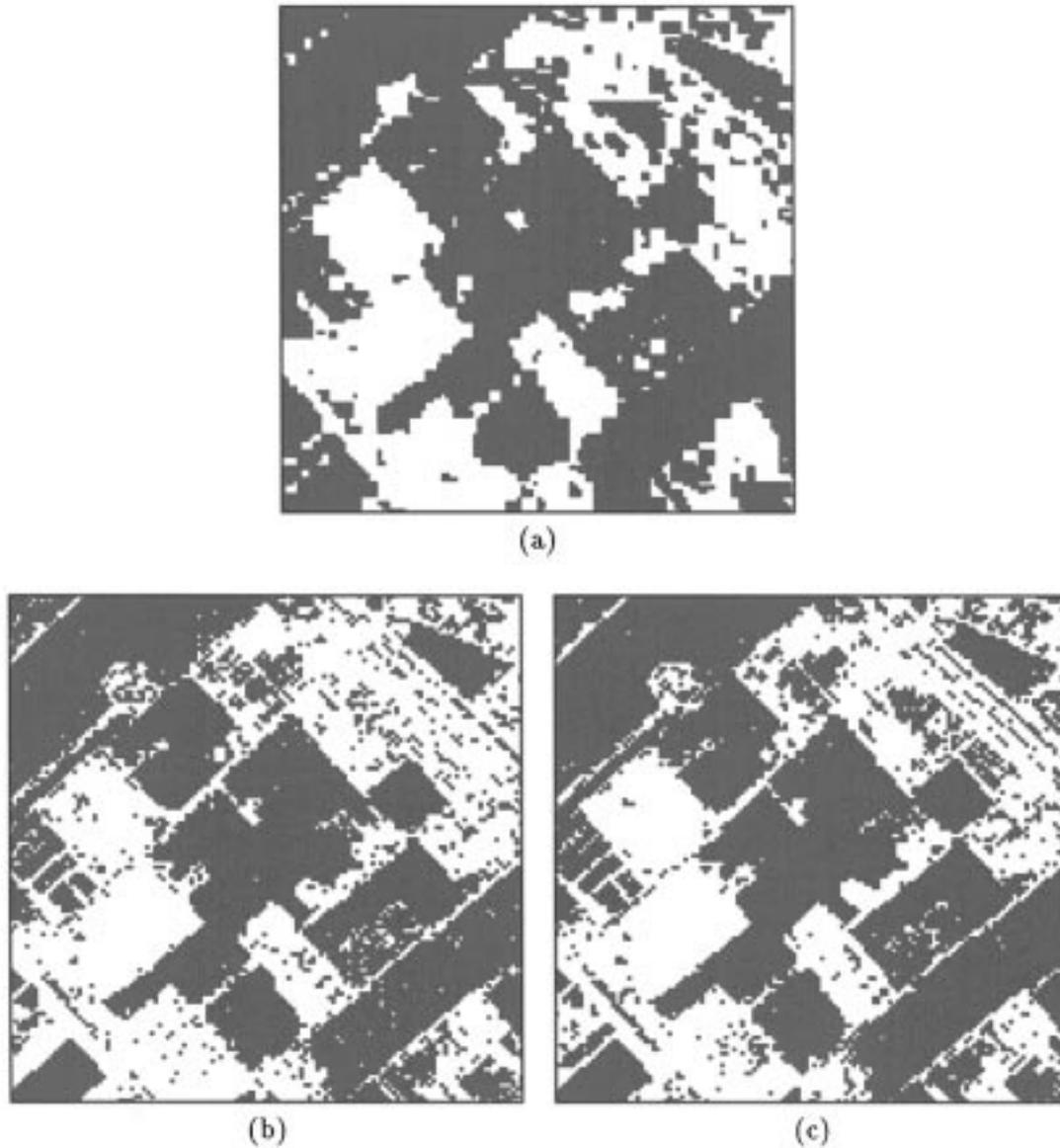


Fig. 7. Comparison of the classification results of 2-D HMM, CART, and LVQ1 for an aerial image. (a) HMM with classification error rate 13.39%. (b) CART using both inter- and intra-block features with classification error rate 20.29%. (c) LVQ1 using both inter- and intra-block features with classification error rate 18.13%. White: man-made. Gray: natural.

accuracy of estimation becomes less with the increase of parameters.

When training and applying the HMM using the path-constrained 2-D Viterbi algorithm, an image was divided into square subimages, each containing 16 blocks. The subimages were considered separate Markov chains. The number of nodes constrained at each position in the Viterbi transition diagram  $N$  was chosen as 32 for the result provided in this section. We experimented with several  $N$ 's. For  $N$  from 2 to 16, the performance is gradually enhanced. For  $N$  greater than 16, the results, with minor differences, start showing a convergence trend. The classification error rate with  $N = 16$  is about 0.26% higher than that with  $N = 32$ . As classification time is spent mainly on the Viterbi searching process, and the Viterbi searching time increases at the order of the second power of the number of nodes at every transition step, the classification time is roughly proportional to  $N^2$ . Experiments were performed

on a Pentium Pro 230 MHz PC with LINUX operating system. The average user CPU time to classify an aerial image is 18 s for  $N = 8$ , 59 s for  $N = 16$ , and 200 s for  $N = 32$ .

The 2-D HMM result was compared with those obtained from two popular block-based statistical classifiers: CART [10] and the first version of Kohonen's learning vector quantization (LVQ) algorithm [27], [28]. The basic idea of CART is to partition a feature space by a tree structure and assign a class to every cell of the partition. Feature vectors landing in a cell are classified as the class of the cell. Since CART is developed for general purposes of decision tree design, we can apply it in the scenario of context dependent classification. As the goal here is to explore how much context improves classification by the 2-D HMM algorithm, CART was applied in a context independent manner to set a benchmark for comparison. In the training process, CART was used to partition feature vectors formed for each image

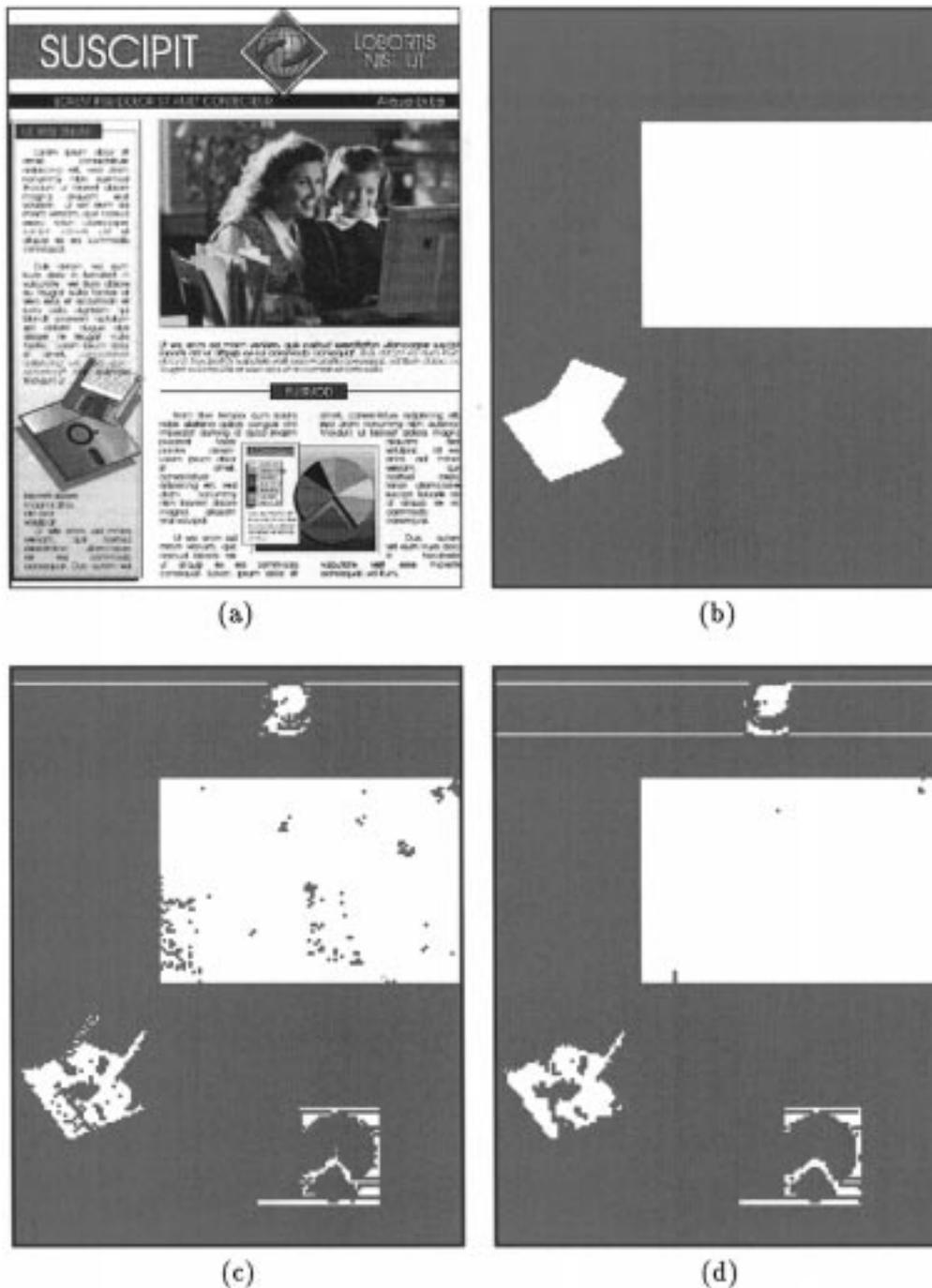


Fig. 8. Test document image 1. (a) Original image. (b) Hand-labeled classified image. (c) CART classification result. (d) 2-D HMM classification result. White: photograph. Gray: text.

block. Images were then classified by tracing their feature vectors independently through the decision tree. Two types of decision trees were trained with CART. One was trained on both inter- and intra-block features; the other was trained on only intra-block features. These two classifiers are referred to as CART 1 and CART 2, respectively. CART 1 incorporates context information implicitly through inter-block features but not as directly and extensively as does the 2-D HMM algorithm.

To compare with LVQ1, we used programs provided by the LVQ\_PAK software package [28]. As with CART 1, classifica-

tion was based on both inter- and intra-block features. The total number of centroids for the two classes is 1024, and the number for each class is proportional to the empirical *a priori* probabilities of the classes. Other parameters were set by default.

The classification results obtained by six-fold cross-validation for 2-D HMM, CART 1, CART 2, and LVQ1 are shown in Table I. Suppose the man-made class is the target class, or positive class. Sensitivity is the true positive ratio, i.e., the probability of detecting positive given the truth is positive. Specificity is the true negative ratio, i.e., the probability of accepting negative given the truth is negative. Predictive value positive (PVP)

is the probability of being truly positive given a positive detection of the classifier. The average percentage of classification error with CART 2 is 24.08%. CART 1 improves the error rate to 21.58%. LVQ1 achieves an error rate of 21.83%, which is close to the result of CART 1. The 2-D HMM algorithm further decreases the error rate to 18.80%. The classification results for Image 6, which is the image shown in Fig. 6(f), are given in Fig. 7. A visual difference to note is that the results of CART 1 and LVQ1 appear “noisy” due to scattered errors caused by classifying blocks independently. Although *ad hoc* postprocessing can eliminate isolated errors, it may increase the error rate if clustered errors occur. Note that at the lower-left corners of Fig. 7(b) and (c), a large continuous region is classified mistakenly as man-made. If postprocessing techniques, such as closing, were applied, the mistakenly classified region would be enlarged. Similar clusters of errors can be found in other parts of the image. On the other hand, if we apply postprocessing after all the three algorithms, the result of the 2-D HMM algorithm provides a better starting point and is less likely to have error propagation.

The segmentation of aerial images was also studied by Oehler [35] and Perlmutter [41]. In both cases, the Bayes vector quantizer (BVQ) [35]–[37], [41] is used as a classifier. With the same set of images and six-fold cross-validation, the best result of simulations with different parameters provides an average classification error rate of roughly 21.5% [41], which is comparable to CART 1.

### C. Document Image Segmentation

The second application of the 2-D HMM algorithm is to segmentation of document images into text and photograph. Photograph refers to continuous-tone images such as scanned pictures, and text refers to normal text, tables, and artificial graphs generated by computer software [32]. We refer to the normal text as text for simplicity if the meaning is clear from context. Images experimented with are 8 bits/pixel gray-scale images. An example image and its segmented image are shown in Fig. 8. This type of classification is useful in a printing process for separately rendering different local image types. It is also a tool for efficient extraction of data from image databases.

Previous work on gray-scale document image segmentation includes Chaddha [11], Williams [49], Perlmutter [41], [42], and Ohuchi [38]. Thresholding is used to distinguish image types in [11]. In [49], a modified quadratic neural network [34] is used for classifying features. In [41] and [42], the Bayes VQ algorithm is applied. As those algorithms were developed particularly for different types of document images, direct comparison with our algorithm is not provided.

The features we use contain the two features described in detail in [32]. The first feature is a measure of the goodness of match between the empirical distribution of wavelet coefficients in high-frequency bands and the Laplacian distribution. It is defined as a  $\chi^2$  statistics normalized by the sample size. The second feature measures the likelihood of wavelet coefficients in high-frequency bands being composed by highly concentrated values. We also use the spatial derivatives of the average intensity values of blocks as features, which is the same as in the previous application. The block size used is  $8 \times 8$ . The HMM has

five states for each class. Experiments show that models with two to five states for each class yield similar results.

The result of HMM is compared with that of a classification tree generated by CART with both inter- and intra-block features. The image set was provided by Hewlett Packard, Inc. [41], [42]. They are RGB color images with size around  $1600 \times 1300$ . Each color component is 8 bits/pixel. In the experiments, only the luminance component (i.e., gray-scale images) was used. For most images tested, both algorithms achieve very low classification error rates: about 2% on average. More differences between the two algorithms appear with one sample image shown in Fig. 8 because the photograph region in this image is very smooth at many places; therefore, it resembles text. The classification results of both CART and the 2-D HMM algorithm are shown in Fig. 8. We see that the result using the HMM is much cleaner than the result using CART, especially in the photograph regions. This is expected since the classification based on the HMM takes context into consideration. As a result, some smooth blocks in the photograph regions, which locally resemble text blocks, can be identified correctly as a photograph.

## VII. CONCLUSION

We have proposed a 2-D hidden Markov model for image classification. The 2-D model provides a structured way to incorporate context information into classification. Using the EM algorithm, we have derived a specific iterative algorithm to estimate the model. As the model is 2-D, computational complexity is an important issue. Fast algorithms are developed to efficiently estimate the model and to perform classification based on the model. The application of the algorithm to several problems shows better performance than that of several popular block-based statistical classification algorithms.

## ACKNOWLEDGMENT

The authors gratefully acknowledge the helpful comments of R. A. Olshen for improving the clarity of the paper. They also wish to thank the reviewers for giving useful suggestions.

## REFERENCES

- [1] K. Abend, T. J. Harley, and L. N. Kanal, “Classification of binary random patterns,” *IEEE Trans. Inform. Theory*, vol. IT-11, pp. 538–544, Oct. 1965.
- [2] J. K. Baker, “The dragon system—An overview,” in *Proc. Int. Conf. Acoust., Speech, Signal Process.*, Feb. 1975, pp. 24–29.
- [3] L. E. Baum, “An inequality and associated maximization technique in statistical estimation for probabilistic functions of finite state Markov chains,” in *Inequalities III*. New York: Academic, 1972, pp. 1–8.
- [4] L. E. Baum and J. A. Eagon, “An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology,” *Bull. Amer. Math. Stat.*, vol. 37, pp. 360–363, 1967.
- [5] L. E. Baum and T. Petrie, “Statistical inference for probabilistic functions of finite state Markov chains,” *Ann. Math. Stat.*, vol. 37, pp. 1554–1563, 1966.
- [6] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, “A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains,” *Ann. Math. Stat.*, vol. 41, no. 1, pp. 164–171, 1970.
- [7] J. Besag, “Spatial interaction and the statistical analysis of lattice systems (with discussion),” *J. R. Stat. Soc.*, ser. B, vol. 34, pp. 75–83, 1972.
- [8] P. J. Bickel and K. A. Doksum, *Mathematical Statistics: Basic Ideas and Selected Topics*. Englewood Cliffs, NJ: Prentice-Hall, 1977.
- [9] J. M. Boyett, “Random  $R \times C$  tables with given row and column totals,” *Appl. Stat.*, vol. 28, pp. 329–332, 1979.

- [10] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. London, U.K.: Chapman & Hall, 1984.
- [11] N. Chaddha, R. Sharma, A. Agrawal, and A. Gupta, "Text segmentation in mixed-mode images," in *Proc. Asilomar Conf. Signals, Syst., Comput.*, vol. 2, Nov. 1994, pp. 1356–1361.
- [12] R. Cole, L. Hirschman, L. Atlas, and M. Beckman *et al.*, "The challenge of spoken language systems: Research directions for the nineties," *IEEE Trans. Speech Audio Processing*, vol. 3, pp. 1–21, Jan. 1995.
- [13] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. R. Stat. Soc.*, vol. 39, no. 1, pp. 1–21, 1977.
- [14] P. A. Devijver, "Probabilistic labeling in a hidden second order Markov mesh," *Pattern Recognition in Practice II*, pp. 113–123, 1985.
- [15] P. A. Devijver, "Segmentation of binary images using third order Markov mesh image models," in *Proc. 8th Int. Conf. Pattern Recogn.*, Paris, France, Oct. 1986, pp. 259–261.
- [16] P. A. Devijver, "Modeling of digital images using hidden Markov mesh random fields," *Signal Processing IV: Theories and Applications (Proc. EUSIPCO-88)*, pp. 23–28, 1988.
- [17] P. A. Devijver, "Real-time modeling of image sequences based on hidden Markov mesh random field models," in *Proc. 10th Int. Conf. Pattern Recogn.*, vol. 2, Los Alamitos, CA, 1990, pp. 194–199.
- [18] P. A. Devijver and M. M. Dekesel, "Experiments with an adaptive hidden Markov mesh image model," *Philips J. Res.*, vol. 43, no. 3/4, pp. 375–392, 1988.
- [19] C. H. Fosgate, H. Krim, W. W. Irving, W. C. Karl, and A. S. Willsky, "Multiscale segmentation and anomaly enhancement of SAR imagery," *IEEE Trans. Image Processing*, vol. 6, pp. 7–20, Jan. 1997.
- [20] R. G. Gallager, *Information Theory and Reliable Communication*. New York, NY: Wiley, 1968.
- [21] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-6, pp. 721–741, Nov. 1984.
- [22] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Boston, MA: Kluwer, 1992.
- [23] X. D. Huang, Y. Ariki, and M. A. Jack, *Hidden Markov Models for Speech Recognition*. Edinburgh, U.K.: Edinburgh Univ. Press, 1990.
- [24] F. Jelinek and J. B. Anderson, "Instrumentable tree encoding of information sources," *IEEE Trans. Inform. Theory*, vol. IT-17, pp. 118–119, Jan. 1971.
- [25] L. N. Kanal, "Markov mesh models," in *Image Modeling*. New York: Academic, 1980, pp. 239–243.
- [26] N.Y.R. Kindermann and J. L. Snell, *Markov Random Fields and Their Applications*. New York: Amer. Math. Soc., 1980.
- [27] T. Kohonen, G. Barna, and R. Chrisley, "Statistical pattern recognition with Neural Networks: Benchmarking studies," in *Proc. IEEE Int. Conf. Neural Networks*, July 1988, pp. 1-61–1-68.
- [28] T. Kohonen, J. Hynninen, J. Kangas, J. Laaksonen, and K. Torkkola, "LVQ\_PAK: The learning vector quantization program package (version 3.1)," Helsinki Univ. Technol., Lab. Comput. Inform. Sci., Helsinki, Finland, Tech. Rep., Apr. 1995. Available via anonymous ftp to cochlea.hut.fi.
- [29] S. S. Kuo and O. E. Agazzi, "Machine vision for keyword spotting using pseudo 2D hidden Markov models," in *Proc. Int. Conf. Acoust., Speech Signal Process.*, vol. 5, 1993, pp. 81–84.
- [30] E. Levin and R. Pieraccini, "Dynamic planar warping for optical character recognition," in *Proc. Int. Conf. Acoust., Speech Signal Process.*, vol. 3, San Francisco, CA, Mar. 1992, pp. 149–152.
- [31] J. Li and R. M. Gray, "Context based multiscale classification of images," presented at the Int. Conf. Image Process., Chicago, IL, Oct. 1998.
- [32] —, "Text and picture segmentation by the distribution analysis of wavelet coefficients," presented at the Int. Conf. Image Processing, Chicago, IL, Oct. 1998.
- [33] A. A. Markov, "An example of statistical investigation in the text of 'Eugene Onyegin' illustrating coupling of 'tests' in chains," *Proc. Acad. Sci.*, ser. 7, p. 153, 1913.
- [34] N. J. Nilsson, *Learning Machines: Foundations of Trainable Pattern-Classifying Systems*. New York: McGraw-Hill, 1965.
- [35] K. L. Oehler, "Image compression and classification using vector quantization," Ph.D. dissertation, Stanford Univ., Stanford, CA, 1993.
- [36] K. L. Oehler and R. M. Gray, "Combining image classification and image compression using vector quantization," in *Proc. Data Compression Conf.*, Snowbird, UT, Mar. 1993, pp. 2–11.
- [37] —, "Combining image compression and classification using vector quantization," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 17, pp. 461–473, 1995.
- [38] S. Ohuchi, K. Imao, and W. Yamada, "Segmentation method for documents containing text/picture (screened halftone, continuous tone)," *Trans. Inst. Electron., Inform., Commun. Eng. D-II*, vol. J75D-II, no. 1, pp. 39–47, Jan. 1992.
- [39] M. Park and D. J. Miller, "Image decoding over noisy channels using minimum mean-squared estimation and a Markov mesh," in *Proc. Int. Conf. Image Process.*, vol. 3, Santa Barbara, CA, Oct. 1997, pp. 594–597.
- [40] D. B. Paul, "Speech recognition using hidden Markov models," *Lincoln Lab. J.*, vol. 3, no. 1, pp. 41–62, 1990.
- [41] K. O. Perlmutter, "Compression and classification of images using vector quantization and decision trees," Ph.D. dissertation, Stanford Univ., Stanford, CA, 1995.
- [42] K. O. Perlmutter, N. Chaddha, J. B. Buckheit, R. M. Gray, and R. A. Olshen, "Text segmentation in mixed-mode images using classification trees and transform tree-structured vector quantization," in *Proc. Int. Conf. Acoust., Speech Signal Process.*, vol. 4, Atlanta, GA, May 1996, pp. 2231–2234.
- [43] K. O. Perlmutter, S. M. Perlmutter, R. M. Gray, R. A. Olshen, and K. L. Oehler, "Bayes risk weighted vector quantization with posterior estimation for image compression and classification," *IEEE Trans. Image Processing*, vol. 5, pp. 347–360, Feb. 1996.
- [44] D. K. Pickard, "A curious binary lattice process," *J. Appl. Prob.*, vol. 14, pp. 717–731, 1977.
- [45] L. Rabiner and B. H. Juang, *Fundamentals of Speech Recognition*. Englewood Cliffs, NJ: Prentice-Hall, 1993.
- [46] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379–423, July 1948.
- [47] M. Stone, "Cross-validation: A review," *Math. Oper. Statist.*, no. 9, pp. 127–139, 1978.
- [48] A. J. Viterbi and J. K. Omura, "Trellis encoding of memoryless discrete-time sources with a fidelity criterion," *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 325–332, May 1974.
- [49] P. S. Williams and M. D. Alder, "Generic texture analysis applied to newspaper segmentation," in *Proc. Int. Conf. Neural Networks*, vol. 3, Washington, DC, June 1996, pp. 1664–1669.
- [50] C. F. J. Wu, "On the convergence properties of the EM algorithm," *Ann. Stat.*, vol. 11, no. 1, pp. 95–103, 1983.
- [51] C. C. Yen and S. S. Kuo, "Degraded documents recognition using pseudo 2D hidden Markov models in Gray-scale images," *Proc. SPIE*, vol. 2277, pp. 180–191, 1994.
- [52] S. Young, J. Jansen, J. Odell, D. Ollason, and P. Woodland, *HTK—Hidden Markov Model Toolkit*. Cambridge, U.K.: Cambridge Univ. Press, 1995.



**Jia Li** was born in Hunan, China, in 1974. She received the B.S. degree in electrical engineering from Xi'an JiaoTong University, Xi'an, China in 1993, the M.S. degree in statistics from Stanford University, Stanford, CA, in 1998, and the M.S. and Ph.D. degrees in electrical engineering from Stanford University in 1999.

She worked as a Research Assistant on image compression and classification in the Electrical Engineering Department and as a Post-Doctoral Resercher on content-based image database retrieval in the Computer Science Department at Stanford University. She is currently a Researcher at the Xerox Palo Alto Research Center, Palo Alto, CA. Her research interests include statistical classification and modeling with applications to image processing and information retrieval.



**Amir Najmi** was born in Karachi, Pakistan, in 1965. He received his B.S. degree in 1986 and the M.S. degree in 1993, both in electrical engineering, from Stanford University, Stanford, CA.

He has worked in industry for 10 years in the areas of digital imaging, speech recognition, and data mining. His current research focus is on applying the theory and practice of data compression to statistical modeling and inference.



**Robert M. Gray** (S'68–M'69–SM'77–F'80) was born in San Diego, CA, on November 1, 1943. He received the B.S. and M.S. degrees from the Massachusetts Institute of Technology, Cambridge, in 1966 and the Ph.D. degree from the University of Southern California, Los Angeles, in 1969, all in electrical engineering.

Since 1969, he has been with Stanford University, Stanford, CA, where he is currently a Professor and Vice Chair of the Department of Electrical Engineering. His research interests are the theory and design of signal compression and classification systems. He is the coauthor, with L. D. Davisson, of *Random Processes* (Englewood Cliffs, NJ: Prentice—Hall, 1986), with A. Gersho, of *An Introduction to Statistical Signal Processing* (<http://www-isl.stanford.edu/~gray/sp.html>) and also with A. Gersho *Vector Quantization and Signal Compression* (Boston, MA: Kluwer, 1992) and, with J. W. Goodman, of *Fourier Transforms* (Boston, MA: Kluwer, 1995). He is the author of *Probability, Random Processes, and Ergodic Properties* (New York: Springer-Verlag, 1988), *Source Coding Theory* (Boston, MA: Kluwer, 1990), and *Entropy and Information Theory* (New York: Springer-Verlag, 1990).

Dr. Gray was a member of the Board of Governors of the IEEE Information Theory Group from 1974 to 1980 and from 1985 to 1988, as well as an Associate Editor (from 1977 to 1980) and Editor-in-Chief (from 1980 to 1983) of the IEEE TRANSACTIONS ON INFORMATION THEORY. He is currently a member of the Board of Governors of the IEEE Signal Processing Society. He was Co-Chair of the 1993 IEEE International Symposium on Information Theory and Program Co-Chair of the 1997 IEEE International Conference on Image Processing. He is an Elected Member of the Image and Multidimensional Signal Processing Technical Committee of the IEEE Signal Processing Society, an Appointed Member of the Multimedia Signal Processing Technical Committee, and a Member of the Editorial Board of the IEEE SIGNAL PROCESSING MAGAZINE. He was co-recipient, with L. D. Davisson, of the 1976 IEEE Information Theory Group Paper Award and corecipient, with A. Buzo, A. H. Gray, and J. D. Markel, of the 1983 IEEE ASSP Senior Award. He was awarded an IEEE Centennial medal in 1984, the IEEE Signal Processing 1993 Society Award in 1994, and the IEEE Signal Processing Society Technical Achievement Award and a Golden Jubilee Award for Technological Innovation from the IEEE Information Theory Society in 1998. He was elected Fellow of the Institute of Mathematical Statistics in 1992 and has held fellowships from Japan Society for the Promotion of Science at the University of Osaka (in 1981), the Guggenheim Foundation at the University of Paris XI (in 1982), and NATO/Consiglio Nazionale delle Ricerche at the University of Naples (in 1990). During spring 1995, he was a Vinton Hayes Visiting Scholar at the Division of Applied Sciences of Harvard University, Cambridge, MA. He is a member of Sigma Xi, Eta Kappa Nu, AAAS, AMS, and the Société des Ingénieurs et Scientifiques de France.