Stochastic and Deterministic Networks for Texture Segmentation

B. S. MANJUNATH, STUDENT MEMBER, IEEE, TAL SIMCHONY, MEMBER, IEEE, AND RAMA CHELLAPPA, SENIOR MEMBER, IEEE

Abstract—**This paper describes several texture segmentation algorithms based on deterministic and stochastic relaxation principles, and their implementation on parallel networks. The segmentation problem is posed as an optimization problem and two different optimality criteria are considered. The first criterion involves maximizing the posterior distribution of the intensity field given the label field (maximum *a posteriori* (MAP) estimate). The posterior distribution of the texture labels is derived by modeling the textures as Gauss Markov random field (GMRF) and characterizing the distribution of different texture labels by a discrete multilevel Markov model. Fast approximate solutions for MAP are obtained using deterministic relaxation techniques implemented on a Hopfield neural network and are compared with those of simulated annealing in obtaining the MAP estimate. A stochastic algorithm which introduces learning into the iterations of the Hopfield network is proposed. This iterated hill-climbing algorithm combines fast convergence of deterministic relaxation with the sustained exploration of the stochastic algorithms, but is guaranteed to find only a local minimum. The second optimality criterion requires minimizing the expected percentage of misclassification per pixel by maximizing the posterior marginal distribution, and the maximum posterior marginal (MPM) algorithm is used to obtain the corresponding solution. All these methods implemented on parallel networks can be easily extended for hierarchical segmentation and we present results of the various schemes in classifying some real textured images.**

I. INTRODUCTION

THIS PAPER describes several algorithms, both deterministic and stochastic, for the segmentation of textured images. Segmentation of image data is an important problem in computer vision, remote sensing, and image analysis. Most objects in the real world have textured surfaces. Segmentation based on texture information is possible even if there are no apparent intensity edges between the different regions. There are many existing methods for texture segmentation and classification, based on different types of statistics that can be obtained from the gray level images. The approach we use stems from the idea of using Markov random field models for textures in an image. We assign two random variables for the observed pixel, one characterizing the underlying intensity and the other for labeling the texture corresponding to the pixel location. We use the Gauss Markov random field (GMRF) model for the conditional density of the intensity field given the label field. Prior information about the texture label field is introduced using a discrete Markov distribution. The segmentation can then be formulated as an optimization problem involving minimization of a Gibbs energy function. Exhaustive search for the optimum solution is not possible because of the large dimensionality of the search space. For example, even for the very simple case of segmenting a 128×128 image into two classes, there are $2^{2^{14}}$ possible label configurations. Derin and Elliott [1] have investigated the use of dynamic programming for obtaining an approximation to the maximum *a posteriori* (MAP) estimate while Cohen and Cooper [2] give a deterministic relaxation algorithm for the same problem. The optimal MAP solution can be obtained by using stochastic relaxation algorithms such as simulated annealing [3]. Recently there has been considerable interest in using neural networks for solving computationally hard problems and the main emphasis in this paper is on developing parallel algorithms which can be implemented on such networks of simple processing elements (neurons).

The inherent parallelism of neural networks provides an interesting architecture for implementing many computer vision algorithms [4]. Some examples are image restoration [5], stereopsis [6], and computing optical flow [7]–[9]. Networks for solving combinatorially hard problems such as the traveling salesman problem have received much attention in the neural network literature [10]. In almost all cases, these networks are designed to minimize an energy function defined by the network architecture. The parameters of the network are obtained in terms of the energy (cost) function it is designed to minimize and it can be shown [10] that for networks having symmetric interconnections, the equilibrium states correspond to the local minima of the energy function. For practical purposes, networks with few interconnections are preferred because of the large number of processing units required in any image processing application. In this context Markov random field (MRF) models for images play a useful role. They are typically characterized by local dependencies and symmetric interconnections which can be expressed in terms of energy functions using Gibbs–Markov equivalence [3].

Manuscript received September 19, 1988; revised August 2, 1989. This work was supported in part by the AFSOR under Grant 86-0196.

B. S. Manjunath and R. Chellappa are with the Department of Electrical Engineering–Systems, University of Southern California, Los Angeles, CA 90089.

T. Simchony was with the Department of Electrical Engineering–Systems, University of Southern California, Los Angeles, CA 90089. He is now with ECI Telecom, Rafael, Israel.

IEEE Log Number 9034989.

We look into two different optimality criteria for segmenting the image. The first corresponds to the label configuration which maximizes the posterior probability of the label array given the intensity array. As noted before, an exhaustive search for the optimal solution is practically impossible. An alternative is to use stochastic relaxation algorithms such as simulated annealing [3], which asymptotically converge to the optimal solution. However the computational burden involved because of the theoretical requirements on the initial temperature and the impractical cooling schedules overweigh their advantages in many cases. Fast approximate solutions can be obtained by such deterministic relaxation algoritms as the iterated conditional mode rule [11]. The energy function corresponding to this optimality criterion can be mapped into a Hopfield-type network in a straightforward manner and it can be shown that the network converges to an equilibrium state, which in general will be a local optimum. The solutions obtained using this method are sensitive to the initial configurations, and in many cases starting with a maximum likelihood estimate is preferred. Stochastic learning can be easily introduced into the network, and the overall system improves the performance by learning while searching. The learning algorithms used are derived from the theory of stochastic learning automata [12] and we believe that this is the first time such a hybrid system has been used in an optimization problem. The stochastic nature of the system helps in preventing the algorithm from being trapped in a local minimum and we observe that this improves the quality of the solutions obtained.

The second optimality criterion minimizes the expected percentage of classification error per pixel. This is equivalent to finding the pixel labels that maximize the marginal posterior probability given the intensity data [13]. Since calculating the marginal posterior probability is very difficult, Marroquin [14] suggested the MPM algorithm, which asymptotically computes the posterior marginal. In [14] the algorithm is used for image restoration, stereo matching, and surface interpolation. Here we use this method to find the texture label that maximizes the marginal posterior probability for each pixel.

The organization of the paper is as follows: Section II describes the image model. A neural network model for the relaxation algorithms is given in Section III along with a deterministic updating rule. Section IV discusses the stochastic algorithms for segmentation and their parallel implementation on the network. A learning algorithm is proposed in Section V and the experimental results are provided in Section VI.

II. Image Model

The use of MRF models for image processing applications has been investigated by many researchers (see e.g., Chellappa [15]). Cross and Jain [16] provide a detailed discussion on the application of MRF in modeling textured images. Geman and Geman [3] discuss the equivalence between MRF and Gibbs distributions. The GMRF model for the texture intensity process has been used in [1], [2], and [17]. The MRF is also used to describe the label process in [1] and [2]. In this paper we use the fourth-order GMRF indicated in Fig. 1 to model the conditional probability density of the image intensity array given its texture labels. The texture labels are assumed to obey a first- or second-order discrete Markov model with a single parameter β, which measures the amount of clustering between adjacent pixels.

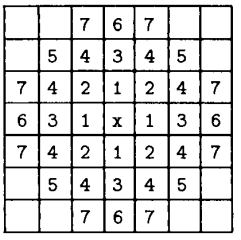


Fig. 1. Structure of the GMRF model. The numbers indicate the order of the model relative to x [16].

Let Ω denote the set of grid points in the $M \times M$ lattice, i.e., $\Omega = \{(i, j),\ 1 \le i, j \le M\}$. Following Geman and Graffigne [18], we construct a composite model which accounts for texture labels and gray levels. Let $\{L_s,\ s \in \Omega\}$ and $\{Y_s,\ s \in \Omega\}$ denote the labels and zero mean gray level arrays respectively. The zero mean array is obtained by subtracting the local mean computed in a small window centered at each pixel. Let N_s denote the symmetric fourth-order neighborhood of a site s. Then, assuming that all the neighbors of s also have the same label as that of s, we can write the following expression for the conditional density of the intensity at the pixel site s:

$$P(Y_s = y_s \mid Y_r = y_r,\ r \in N_s,\ L_s = l) = \frac{\exp\left[-U(Y_s = y_s \mid Y_r = y_r,\ r \in N_s,\ L_s = l)\right]}{Z(l \mid y_r,\ r \in N_s)} \tag{1}$$

where $Z(l \mid y_r,\ r \in N_s)$ is the partition function of the conditional Gibbs distribution and

$$U(Y_s = y_s \mid Y_r = y_r,\ r \in N_s,\ L_s = l) = \frac{1}{2\sigma_l^2}\left(y_s^2 - 2\sum_{r \in N_s} \Theta^l_{s,r} y_s y_r\right). \tag{2}$$

In (2), σ_l and Θ^l are the GMRF model parameters of the lth texture class. The model parameters satisfy $\Theta^l_{r,s} = \Theta^l_{r-s} = \Theta^l_{s-r} = \Theta^l_r$.

We view the image intensity array as composed of a set of overlapping $k \times k$ windows W_s, centered at each pixel $s \in \Omega$. In each of these windows we assume that the texture label L_s is homogeneous (all the pixels in the window belonging to the same texture) and compute the joint distribution of the intensity in the window conditioned on L_s. The corresponding Gibbs energy is used in the relaxation process for segmentation. As explained in the previous paragraph, the image intensity in the window is modeled by a fourth-order stationary GMRF. The local mean is

computed by taking the average of the intensities in the window W_s and is subtracted from the original image to get the zero mean image. All our references to the intensity array correspond to the zero mean image. Let Y_s^* denote the 2-D vector representing the intensity array in the window W_s. Using the Gibbs formulation and assuming a free boundary model, the joint probability density in the window W_s can be written as [2]

$$P(Y_s^* \mid L_s = l) = \frac{\exp\left[-U_1(Y_s^* \mid L_s = l)\right]}{Z_1(l)}$$

where $Z_1(l)$ is the partition function and

$$U_1(Y_s^* \mid L_s = l) = \frac{1}{2\sigma_l^2} \sum_{r \in W_s} \left\{ y_r^2 - \sum_{\tau \in N^* \mid r+\tau \in W_s} \cdot \Theta_\tau^l y_r (y_{r+\tau} + y_{r-\tau}) \right\}. \quad (3)$$

N^* is the set of shift vectors corresponding to a fourth-order neighborhood system:

$$\begin{aligned} N^* &= \{\tau_1, \tau_2, \tau_3, \cdots, \tau_{10}\} \\ &= \{(0,1), (1,0), (1,1), (-1,1), (0,2), (2,0), \\ &\quad (1,2), (2,1), (-1,2), (-2,1)\}. \end{aligned}$$

The label field is modeled as a first- or second-order discrete MRF. If $\hat{N}_s$ denotes the appropriate neighborhood for the label field, then we can write the distribution function for the texture label at site s conditioned on the labels of the neighboring sites as

$$P(L_s \mid L_r, r \in \hat{N}_s) = \frac{e^{-U_2(L_s \mid L_r)}}{Z_2}$$

where Z_2 is a normalizing constant and

$$U_2(L_s \mid L_r, r \in \hat{N}_s) = -\beta \sum_{r \in \hat{N}_s} \delta(L_s - L_r), \qquad \beta > 0. \quad (4)$$

In (4), β determines the degree of clustering, and $\delta(i-j)$ is the Kronecker delta. Using the Bayes rule, we can write

$$P(L_s \mid Y_s^*, L_r, r \in \hat{N}_s) = \frac{P(Y_s^* \mid L_s)\, P(L_s \mid L_r, r \in \hat{N}_s)}{P(Y_s^*)}. \quad (5)$$

Since Y_s^* is known, the denominator in (5) is just a constant. The numerator is a product of two exponential functions and can be expressed as

$$P(L_s \mid Y_s^*, L_r, r \in \hat{N}_s) = \frac{1}{Z_p} \exp\left[-U_p(L_s \mid Y_s^*, L_r, r \in \hat{N}_s)\right] \quad (6)$$

where Z_p is the partition function and $U_p(\cdot)$ is the posterior energy corresponding to (5). From (3) and (4) we write

$$U_p(L_s \mid Y_s^*, L_r, r \in \hat{N}_s) = w(L_s) + U_1(Y_s^* \mid L_s) + U_2(L_s \mid L_r, r \in \hat{N}_s). \quad (7)$$

Note that the second term in (7) relates the observed pixel intensities to the texture labels and the last term specifies the label distribution. The bias term $w(L_s) = \log Z_1(L_s)$ is dependent on the texture class and it can be explicitly evaluated for the GMRF model considered here using the toroidal assumption (the computations become very cumbersome if toroidal assumptions are not made). An alternative approach is to estimate the bias from the histogram of the data as suggested by Geman and Graffigne [18]. Finally, the posterior distribution of the texture labels for the entire image given the intensity array is

$$P(L \mid Y^*) = \frac{P(Y^* \mid L)\, P(L)}{P(Y^*)}. \quad (8)$$

Maximizing (8) gives the optimal Bayesian estimate. Though it is possible in principle to compute the right-hand side of (8) and find the global optimum, the computational burden involved is so enormous that it is practically impossible to do so. However, we note that the stochastic relaxation algorithms discussed in Section IV require only the computation of (6) to obtain the optimal solution. The deterministic relaxation algorithm given in the next section also uses these values, but in this case the solution is only an approximation to the MAP estimate.

III. A Neural Network for Texture Classification

We describe the network architecture used for segmentation and the implementation of deterministic relaxation algorithms. The energy function which the network minimizes is obtained from the image model discussed in the previous section. For convenience of notation let $U_1(i, j, l) = U_1(Y_s^*, L_s = l) + w(l)$, where $s = (i, j)$ denotes a pixel site and $U_1(\cdot)$ and $w(l)$ are as defined in (7). The network consists of K layers, each layer arranged as an $M \times M$ array, where K is the number of texture classes in the image and M is the dimension of the image. The elements (neurons) in the network are assumed to be binary and are indexed by (i, j, l) where $(i, j) = s$ refers to their position in the image and l refers to the layer. The (i, j, l)th neuron is said to be ON if its output V_{ijl} is 1, indicating that the corresponding site $s = (i, j)$ in the image has the texture label l. Let $T_{ijl;i'j'l'}$ be the connection strength between neurons (i, j, l) and (i', j', l') and I_{ijl} be the input bias current. Then a general form for the energy of the network is [10]

$$E = -\tfrac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} \sum_{l=1}^{K} \sum_{i'=1}^{M} \sum_{j'=1}^{M} \sum_{l'=1}^{K} T_{ijl;i'j'l'} V_{ijl} V_{i'j'l'} - \tfrac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} \sum_{l=1}^{K} I_{ijl} V_{ijl}. \quad (9)$$

From our discussion in Section II, we note that a solution for the MAP estimate can be obtained by minimizing (8). Here we approximate the posterior energy by

$$U(\boldsymbol{L} \mid \boldsymbol{Y}^*) = \sum_s \left\{ U_1(Y_s^* \mid L_s) + W(L_s) + U_2(L_s) \right\} \tag{10}$$

and the corresponding Gibbs energy to be minimized can be written as

$$E = \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} \sum_{l=1}^{K} U_1(i, j, l)\, V_{ijl} - \frac{\beta}{2} \sum_{l=1}^{K} \sum_{i=1}^{M} \sum_{j=1}^{M} \sum_{(i',j') \in \hat{N}_{ij}} V_{i'j'l} V_{ijl} \tag{11}$$

where $\hat{N}_{ij}$ is the neighborhood of site (i, j) (same as the $\hat{N}_s$ in Section II). In (11), it is implicitly assumed that each pixel site has a unique label; i.e., only one neuron is active in each column of the network. This constraint can be implemented in different ways. For the deterministic relaxation algorithm described below, a simple method is to use a *winner-takes-all* circuit for each column so that the neuron receiving the maximum input is turned on and the others are turned off. Alternatively, a penalty term can be introduced in (11) to represent the constraint as in [10]. From (9) and (11) we can identify the parameters for the network:

$$T_{ijl;i'j'l'} = \begin{cases} \beta & \text{if } (i', j') \in \hat{N}_{ij},\ \forall l = l' \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

and the bias current

$$I_{ijl} = -U_1(i, j, l). \tag{13}$$

A. Deterministic Relaxation

The above equations (12) and (13) relate the parameters of the network to that of the image model. The connection matrix for the above network is symmetric and there is no self-feedback; i.e., $T_{ijl;ijl} = 0$, $\forall i, j, l$. Let u_{ijl} be the potential of neuron (i, j, l). With l the layer number corresponding to texture class l, we have

$$u_{ijl} = \sum_{i'=1}^{M} \sum_{j'=1}^{M} \sum_{l'=1}^{K} T_{ijl;i'j'l'} V_{i'j'l'} + I_{ijl}. \tag{14}$$

In order to minimize (11), we use the following updating rule:

$$V_{ijl} = \begin{cases} 1 & \text{if } u_{ijl} = \min_{l'} \{u_{ijl'}\} \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

This updating scheme ensures that at each stage the energy decreases. Since the energy is bounded, the convergence of the above system is ensured but the stable state will in general be a local optimum.

This network model is a version of the iterated conditional mode (ICM) algorithm of Besag [11]. This algorithm maximizes the conditional probability $P(L_s = l \mid \boldsymbol{Y}_s^*, L_{s'}, s' \in \hat{N}_s)$ during each iteration. It is a local deterministic relaxation algorithm that is very easy to implement. We observe that in general an algorithm based on MRF models can be easily mapped onto neural networks with local interconnections. The main advantage of this deterministic relaxation algorithm is its simplicity. Often the solutions are reasonably good and the algorithm usually converges within 20–30 iterations. In the next section we study two stochastic schemes which asymptotically converge to the global optimum of the respective criterion functions.

IV. Stochastic Algorithms for Texture Segmentation

We look at two optimal solutions corresponding to different decision rules for determining the labels. The first one uses simulated annealing to obtain the optimum MAP estimate of the label configuration. The second algorithm minimizes the expected misclassification per pixel. The parallel network implementation of these algorithms is discussed in Section IV-C.

A. Searching for MAP Solution

The MAP rule [18] searches for the configuration L that maximizes the posterior probability distribution. This is equivalent to maximizing $P(\boldsymbol{Y}^* \mid \boldsymbol{L})\, P(\boldsymbol{L})$ as $P(\boldsymbol{Y}^*)$ is independent of the labels and $\boldsymbol{Y}^*$ is known. The right-hand side of (8) is a Gibbs distribution. To maximize (8) we use simulated annealing [3], a combinatorial optimization method which is based on sampling from varying Gibbs distribution functions:

$$\frac{\exp\left[-\frac{1}{T_k} U_p(L_s \mid \boldsymbol{Y}_s^*, L_r, r \in \hat{N}_s) \right]}{Z_{T_k}}$$

in order to maximize

$$\frac{e^{-U_p(\boldsymbol{L} \mid \boldsymbol{Y}^*)}}{Z}$$

T_k being the time-varying parameter, referred to as the temperature. We used the following cooling schedule:

$$T_k = \frac{T_0}{1 + \log_2 k} \tag{16}$$

where k is the iteration number. When the temperature is high, the bond between adjacent pixels is loose, and the distribution tends to behave like a uniform distribution over the possible texture labels. As T_k decreases, the distribution concentrates on the lower values of the energy function, which correspond to points with higher probability. The process is bound to converge to a uniform distribution over the label configuration that corresponds to the MAP solution. Since the numer of texture labels is finite, convergence of this algorithm follows from [3]. In our experiment, we realized that starting the iterations with $T_0 = 2$ did not guarantee convergence to the MAP solution. Since starting at a much higher temperature will slow the convergence of the algorithm significantly, we use an alternative approach, viz., cycling the temperature

[13]. We follow the annealing schedule until T_k reaches a lower bound; then we reheat the system and start a new cooling process. By using only a few cycles, we obtained results better than those with a single cooling cycle. Parallel implementation of simulated annealing on the network is discussed in Section IV-C. The results we present in Section VI were obtained with two cycles.

B. Maximizing the Posterior Marginal Distribution

The choice of the objective function for optimal segmentation can significantly affect its result. The choice should be made depending on the purpose of the classification. In many implementations the most reasonable objective function is the one that minimizes the expected percentage misclassification per pixel. The solution to the above objective function is also the one that maximizes the marginal posterior distribution of L_s given the observation $\boldsymbol{Y}^*$ for each pixel s:

$$P\{L_s = l_s \mid \boldsymbol{Y}^* = \boldsymbol{y}^*\} \propto \sum_{l \mid L_s = l_s} P(\boldsymbol{Y}^* = \boldsymbol{y}^* \mid \boldsymbol{L} = \boldsymbol{l})\, P(\boldsymbol{L} = \boldsymbol{l}).$$

The summation above extends over all possible label configurations keeping the label at site s constant. This concept was thoroughly investigated in [14]. Marroquin [19] discusses this formulation in the context of image restoration and illustrates the performance on images with few gray levels. He also mentions the possibility of using this objective function for texture segmentation. In [11] the same objective function is mentioned in the context of image estimation.

To find the optimal solution we use the stochastic algorithm suggested in [14]. The algorithm samples out of the posterior distribution of the texture labels given the intensity. Unlike the stochastic relaxation algorithm, samples are taken with a fixed temperature $T = 1$. The Markov chain associated with the sampling algorithm converges with probability 1 to the posterior distribution. We define new random variables $g_s^{l,t}$ for each pixel ($s \in \Omega$):

$$g_s^{l,t}\{L_s^t\} = \begin{cases} 1 & L_s^t = l \\ 0 & \text{otherwise} \end{cases}$$

where L_s^t is the class of the s pixel, at time t, in the state vector of the Markov chain associated with the Gibbs sampler. We use the ergodic property of the Markov chain [20] to calculate the expectations for these random variables using time averaging:

$$E\{g_s^{l,t}\} = \lim_{N \to \infty} \frac{1}{N} g_s^{l,t} = P_s\{L_s = l \mid Y^*\}$$

where N is the number of iterations performed. To obtain the optimal class for each pixel, we simply chose the class that occurred more often than the others.

The MPM algorithm was implemented using the Gibbs sampler [3]. A much wider set of sampling algorithms, such as Metropolis, can be used for this purpose. The algorithms can be implemented sequentially or in parallel, with a deterministic or stochastic decision rule for the order of visiting the pixels. In order to avoid the dependence on the initial state of the Markov chain, we can ignore the first few iterations. In the experiments conducted we obtained good results after 500 iterations. The algorithm does not suffer from the drawbacks of simulated annealing. For instance we do not have to start the iterations with a high temperature to avoid local minima, and the performance is not badly affected by enlarging the state space.

C. Network Implementation of the Sampling Algorithms

All the stochastic algorithms described in the Gibbs formulation are based on sampling from a probability distribution. The probability distribution is constant in the MPM algorithm [14] and is time varying in the case of annealing. The need for parallel implementation is due to the heavy computational load associated with their use.

The issue of parallel implementation in stochastic algorithms was first addressed by Geman and Geman [3]. They show that the Gibbs sampler can be implemented by any deterministic or stochastic rule for choosing the order in which pixels are updated, as long as each pixel is visited infinitely often. An iteration is the time required to visit each pixel at least once (a full sweep). Note that the stochastic rules have a random period and allow us to visit a pixel more than once in a period. They consider the new Markov chain one obtains from the original by viewing it only after each iteration. Their proof is based on two essential elements. The first is the fact that the embedded Markov chain has a strictly positive transition probability p_{ij} for any possible states i, j, which proves that the chain will converge to a unique probability measure regardless of the initial state. The second is that the Gibbs measure is an invariant measure for the Gibbs sampler, so that the embedded chain converges to the Gibbs measure. The proof introduced in [3] can be applied to a much larger family of sampling algorithms satisfying the following properties [20]:

1) The sampler produces a Markov chain with a postive transition probability p_{ij} for any choice of states i, j.
2) The Gibbs measure is invariant under the sampling algorithm.

The Metropolis and heat bath algorithms are two such sampling methods. To see that the Metropolis algorithm satisfies property 2, we look at the following equation for updating a single pixel:

$$P^{n+1}(i) = \frac{1}{m} \sum_{\pi(j) < \pi(i)} P^n(j) + \frac{1}{m} \sum_{\pi(j) < \pi(i)} P^n(i) \frac{\pi(i) - \pi(j)}{\pi(i)} + \frac{1}{m} \sum_{\pi(j) \geq \pi(i)} P^n(j) \frac{\pi(i)}{\pi(j)}$$

where m is the number of values each pixel can take. The first term corresponds to the cases when the system was in state j and the new state i has higher probability. The second term corresponds to a system in state i and a new state j that has lower probability. The given probability is for staying in state i. The third term corresponds to a system in state j and a new state i with lower probability. If we now replace $P^{n+1}(i)$ and $P^n(i)$ with $\pi(i)$ and $P^n(j)$, we see that the equality holds, implying that the Gibbs measure is invariant under the Metropolis algorithm. The first property is also satisfied. Note that the states now correspond to the global configuration. To implement the algorithm in parallel, one can update pixels in parallel as long as neighboring pixels are not updated at the same time. A very clear discussion on this issue can be found in [14].

We now describe how these stochastic algorithms can be implemented on the network discussed in Section III. The only modification required for the simulated annealing rule is that the neurons in the network fire according to a time-dependent probabilistic rule. Using the same notation as in Section III, the probability that neuron (i, j, l) will fire during iteration k is

$$P(V_{ijl} = 1) = \frac{e^{-(1/T_k)u_{ijl}}}{Z_{T_k}} \tag{17}$$

where u_{ijl} is as defined in (14) and T_k follows the cooling schedule (16).

The MPM algorithm uses the above selection rule with $T_k = 1$. In addition, each neuron in the network has a counter which is incremented every time the neuron fires. When the iterations are terminated the neuron in each column of the network having the maximum count is selected to represent the label for the corresponding pixel site in the image.

We have noted before that for parallel implementation of the sampling algorithms, neighboring sites should not be updated simultaneously. Some additional observations are made in Section VI.

V. Stochastic Learning and Neural Networks

In the previous sections purely deterministic and stochastic relaxation algorithms were discussed. Each has its own advantages and disadvantages. Here we consider the possibility of combining the two methods using stochastic learning automata and we compare the results obtained by this new scheme with those of previous algorithms.

We begin with a brief introduction to the stochastic learning automaton [12]. An automation is a decision maker operating in a random environment. A stochastic automation can be defined by a quadruple (α, Q, T, R), where $\boldsymbol{\alpha} = \{\alpha_1, \cdots, \alpha_N\}$ is the set of available actions to the automaton. The action selected at time t is denoted by $\alpha(t)$. $Q(t)$ is the state of the automaton at time t and consists of the action probability vector $\boldsymbol{p}(t) = [p_1(t), \cdots, p_N(t)]$, where $p_i(t) = \text{prob}(\alpha(t) = \alpha_i)$ and $\Sigma_i p_i(t) = 1 \ \forall t$. The environment responds to the action $\alpha(t)$ with a $\lambda(t) \in R$, R being the set of the environment's responses. The state transitions of the automaton are governed by the learning algorithm T, $Q(t+1) = T(Q(t), \alpha(t), \lambda(t))$. Without loss of generality, it can be assumed that $R = [0, 1]$; i.e., the responses are normalized to lie in the interval $[0, 1]$, 1 indicating a complete *success* and 0 total *failure*. The goal of the automaton is to converge to the optimal action, i.e., the action which results in the maximum expected reward. Again without loss of generality let α_1 be the optimal action and $d_1 = E[\lambda(t) \mid \alpha_1] = \max_i \{E[\lambda(t) \mid \alpha_i]\}$. At present no learning algorithms exist which are optimal in the above sense. However we can choose the parameters of certain learning algorithms so as to realize a response as close to the optimum as desired. This condition is called ϵ optimality. If $M(t) \triangleq E[\lambda(t) \mid p(t)]$, then a learning algorithm is said to be ϵ optimal if it results in an $M(t)$ such that

$$\lim_{t \to \infty} E[M(t)] > d_1 - \epsilon \tag{18}$$

for a suitable choice of parameters and for any $\epsilon > 0$. One of the simplest learning schemes is the linear reward-inaction rule, L_{R-I}. Suppose at time t we have $\alpha(t) = \alpha_i$; if $\lambda(t)$ is the response received, then according to the L_{R-I} rule

$$\begin{aligned} p_i(t+1) &= p_i(t) + a\lambda(t)\left[1 - p_i(t)\right] \\ p_j(t+1) &= p_j(t)\left[1 - a\lambda(t)\right], \qquad \forall j \neq i \end{aligned} \tag{19}$$

where a is a parameter of the algorithm controlling the learning rate. Typical values for a are in the range 0.01–0.1. It can be shown that this L_{R-I} rule is ϵ optimal in all stationary environments; i.e., there exists a value for the parameter a so that condition (18) is satisfied.

Collective behavior of a group of automata has also been studied. Consider a team of N automata $A_i (i = 1, \cdots, N)$, each having r_i actions $\boldsymbol{\alpha}^i = \{\alpha_1^i \cdots \alpha_{r_i}^i\}$. At any instant t each member of the team makes a decision $\alpha^i(t)$. The environment responds to this by sending a reinforcement signal $\lambda(t)$ to all the automata in the group. This situation represents a cooperative game among a team of automata with an identical payoff. All the automata update their action probability vectors according to (19) using the same learning rate, and the process repeats. Local convergence results can be obtained for the case of stationary random environments. Variations of this rule have been applied to complex problems such as decentralized control of Markov chains [21] and relaxation labeling [22].

The texture classification discussed in the previous sections can be treated as a relaxation labeling problem and stochastic automata can be used to learn the labels (texture class) for the pixels. A learning automaton is assigned to each of the pixel sites in the image. The actions of the automata correspond to selecting a label for the pixel site to which it is assigned. Thus for a K-class problem each automaton has K actions and a probability dis-

tribution over this action set. Initially the labels are assigned randomly with equal probability. Since the number of automata involved is very large, it is not practical to update the action probability vector at each iteration. Instead we combine the iterations of the neural network described in the previous section with the stochastic learning algorithm. This results in an iterative hill-climbing-type algorithm which combines the fast convergence of deterministic relaxation with the sustained exploration of the stochastic algorithm. The stochastic part prevents the algorithm from getting trapped in local minima and at the same time "learns" from the search by updating the state probabilities. However, in contrast to simulated annealing, we cannot guarantee convergence to the global optimum. Each cycle now has two phases: the first consists of the deterministic relaxation network converging to a solution; the second consists of the learning network updating its state, the new state being determined by the equilibrium state of the relaxation network. A new initial state is generated by the learning network depending on its current state and the cycle repeats. Thus relaxation and learning alternate with each other. After each iteration the probability of the more stable states increases and because of the stochastic nature of the algorithm the possibility of getting trapped in bad local minima is reduced. The algorithm is summarized below.

A. Learning Algorithm

Let the pixel site be denoted by $s \in \Omega$ and the number of texture classes be K. Let A_s be the automaton assigned to site s and the action probability vector of A_s be $\boldsymbol{p}_s(t) = [p_{s,1}(t), \cdots, p_{s,k}(t)]$ and $\Sigma_i p_{s,i}(t) = 1 \; \forall s, t$, where $p_{s,l}(t) =$ prob (label of site $s = l$). The steps in the algorithm are as follows:

1) Initialize the action probability vectors of all the automata

$$p_{s,l}(0) = 1/K, \qquad \forall s, l.$$

Initialize the iteration counter to 0.

2) Choose an initial label configuration sampled from the distribution of these probability vectors.

3) Start the neural network of Section III with this configuration.

4) Let l_s denote the label for site s at equilibrium. Let the current time (iteration number) be t. Then the action probabilities are updated as follows:

$$p_{s,l_s}(t+1) = p_{s,l_s}(t) + a\lambda(t)\left[1 - p_{s,l_s}(t)\right]$$

$$p_{s,j}(t+1) = p_{s,j}(t)\left[1 - a\lambda(t)\right],$$

$$\forall j \neq l_s \text{ and } \forall s. \qquad (20)$$

The response $\lambda(t)$ is derived as follow: Suppose the present label configuration resulted in a lower energy state than the previous one. Then it results in $\lambda(t) = \lambda_1$, and if the energy increases we have $\lambda(t) = \lambda_2$ with $\lambda_1 > \lambda_2$. In our simulations we used $\lambda_1 = 1$ and $\lambda_2 = 0.25$.

5) Generate a new configuration from this updated label probabilities, increment the iteration counter, and go to step 3.

Thus the system consists of two layers, one for relaxation and the other for learning. The relaxation network is similar to the one considered in Section III, the only difference being that the initial state is decided by the learning network. The learning network consists of a team of automata and learning takes place at a much lower speed than the relaxation, with fewer updatings. The probabilities of the labels corresponding to the final state of the relaxation network are increased according to (20). Using these new probabilities a new configuation is generated. Since the response does not depend on time, this corresponds to a stationary environment, and as we have noted before this L_{R-I} algorithm can be shown to converge to a stationary point, not necessarily to the global optimum.

VI. Experimental Results and Conclusions

The segmentation results using the above algorithms are given on two examples. The parameters σ_l and Θ_l corresponding to the fourth-order GMRF for each texture class were precomputed from 64 × 64 images of the textures. The local mean (in an 11 × 11 window) was first subtracted to obtain the zero mean texture, and the least-square estimates [17] of the parameters were then computed from the interior of the image. The first step in the segmentation process involves computing the Gibbs energies $U_l(\boldsymbol{Y}_s^* \mid L_s)$ in (3). This is done for each texture class and the results are stored. For computational convenience these $U_l(\cdot)$ values are normalized by dividing by k^2, where k is the size of the window. To ignore the boundary effects, we set $U_l = 0$ at the boundaries. We have experimented with different window sizes; larger windows result in more homogeneous texture patches but the boundaries between the textures are distorted. The results reported here are based on windows of size 11 × 11 pixels. The bias term $w(l_s)$ can be estimated using the histogram of the image data [18] but we obtained these values by trial and error.

In Section IV we observed that neighboring pixel sites should not be updated simultaneously. This problem occurs only if digital implementations of the networks are considered, as the probability of this happening in an analog network is zero. When this simultaneous updating was tested for the deterministic case, it always converged to limit cycles of length 2. (In fact it can be shown that the system converges to limit cycles of length at most 2.)

The choice of β plays an important role in the segmentation process and its value depends on the magnitude of the energy function $U_l(\cdot)$. Various values of β ranging from 0.2–3.0 were used in the experiments. In the deterministic algorithm it is preferable to start with a small β and increase it gradually. Large values of beta usually degrade the performance. We also observed that slowly increasing β during the iterations improves the results for

the stochastic algorithms. It should be noted that using a larger value of β for the deterministic algorithm (compared to those used in the stochastic algorithms) does not improve the performance.

The nature of the segmentation results depends on the order of the label model. It is preferable to choose the first-order model for the stochastic algorithms if we know *a priori* that the boundaries are either horizontal or vertical. However, for the deterministic rule and the learning scheme the second-order model results in more homogeneous classification.

The MPM algorithm requires the statistics obtained from the invariant measure of the Markov chain corresponding to the sampling algorithm. Hence it is preferable to ignore the first few hundred trials before starting to gather the statistics. The performance of the deterministic relaxation rule of Section III also depends on the initial state and we have looked into two different initial conditions. The first one starts with a label configuration $\boldsymbol{L}$ such that $L_s = l_s$ if $U_1(\boldsymbol{Y}_s^* \mid l_s) = \min_{l_k} \{U_1(\boldsymbol{Y}_s^* \mid l_k)\}$. This corresponds to maximizing the probability $P(\boldsymbol{Y}^* \mid \boldsymbol{L})$ [23]. The second choice for the initial configuration is a randomly generated label set. Results for both cases are provided and we observe that the random choice often leads to better results.

In the examples below the following learning parameters were used: learning rate $a = 0.05$ and reward/penalty parameters $\lambda_1 = 1.0$ and $\lambda_2 = 0.25$.

Example 1: This is a two-class problem consisting of grass and calf textures. The image is of size 128×128 and is shown in Fig. 2(a). In Fig. 2(b) the classification obtained by the deterministic algorithm discussed in Section III is shown. The maximum likelihood estimate was the initial state for the network, and Fig. 2(c) gives the result with random initial configuration. Notice that in this case the final result has fewer misclassified regions than in Fig. 2(b) and this was observed to be true in general. Parts (d) and (e) of the figure give the MAP solution using simulated annealing and the MPM solution respectively. The result of the learning algorithm is shown in Fig. 2(f) and there are no misclassifications within the homogeneous regions. However the boundary is not as good as those of the MAP or MPM solutions. In all the cases we used $\beta = 0.6$.

Example 2: This is a 256×256 image (Fig. 2(a)) having six textures: calf, grass, wool, wood, pig skin, and sand. This is a difficult problem in the sense that three of the textures (wool, pig skin, and sand) have almost identical characteristics and are not easily distinguishable, even by the human eye. The maximum likelihood solution is shown in Fig. 3(b), and part (c) of the figure is the solution obtained by the deterministic relaxation network with the result in part (b) as the initial condition. Fig. 3(d) gives the result with random initial configuration. The MAP solution using simulated annealing is shown in part (e). As was mentioned in Section IV-A, cycling of temperature improves the performance of simulated annealing. The segmentation result ws obtained by starting with an initial temperature $T_0 = 2.0$ and cooling according to the schedule (16) for 300 iterations. Then the system was reset to $T_0 = 1.5$ and the process was repeated for 300 more iterations. In the case of the MPM rule the first 500 iterations were ignored and Fig. 3(f) shows the result obtained using the last 200 iterations. As in the previous example, the best results were obained by the simulated annealing and MPM algorithms. For the MPM case there were no misclassifications within homogeneous regions but the boundaries were not accurate. In fact, as indicated in Table I, simulated annealing has the lowest percentage error in classification. Introducing learning in deterministic relaxation considerably improves the performance (Fig. 3(g)). Table I gives the percentage classification error for the different cases.

It is noted from the table that although learning improves the performance of the deterministic network algorithm, the best results were obtained by the simulated annealing technique, which is to be expected.

A. Hierarchical Segmentation

The various segmentation algorithms described in the previous sections can be easily extended to hierarchical structures wherein the segmentation is carried out at different levels—from coarse to fine. The energy functions are modified to take care of the coupling between the adjacent resolutions of the system. Consider a K-stage hierarchical system, with stage 0 representing the maximum resolution level and stage $K-1$ being the coarsest level. The energy corresponding to the kth stage is denoted by $U_1^k(s, l)$ and $U_2^k(s)$ (eqs. (3) and (4)). The size of the window used in computing the joint energy potential $U_1^k(\cdot)$ increases with the index k. The potential U_2 is modified to take care of the coupling as follows:

$$U_2^k(s) = -\beta \sum_{t' \in D_s^k} \delta\big(L^k(s) - L^k(t')\big) + \beta_k\big(\delta\big(L^k(s) - L^{k+1}(s)\big)\big) + w(L(s)), \quad 0 \le k < K-1 \tag{21}$$

where $L^k(s)$ is the label for the site s in the kth stage, and β_k is the coupling coefficient between the stages $k+1$ and k. $D^k(s)$ is the appropriate neighborhood set for the kth stage. The result of segmentation on the six-class problem with $K = 2$ and using the learning algorithm is shown in Fig. 3(h).

B. Conclusion

In this paper we have looked into different texture segmentation algorithms based on modeling the texture intensities as a GMRF. It is observed that a large class of natural textures can be modeled in this way. The perfor-

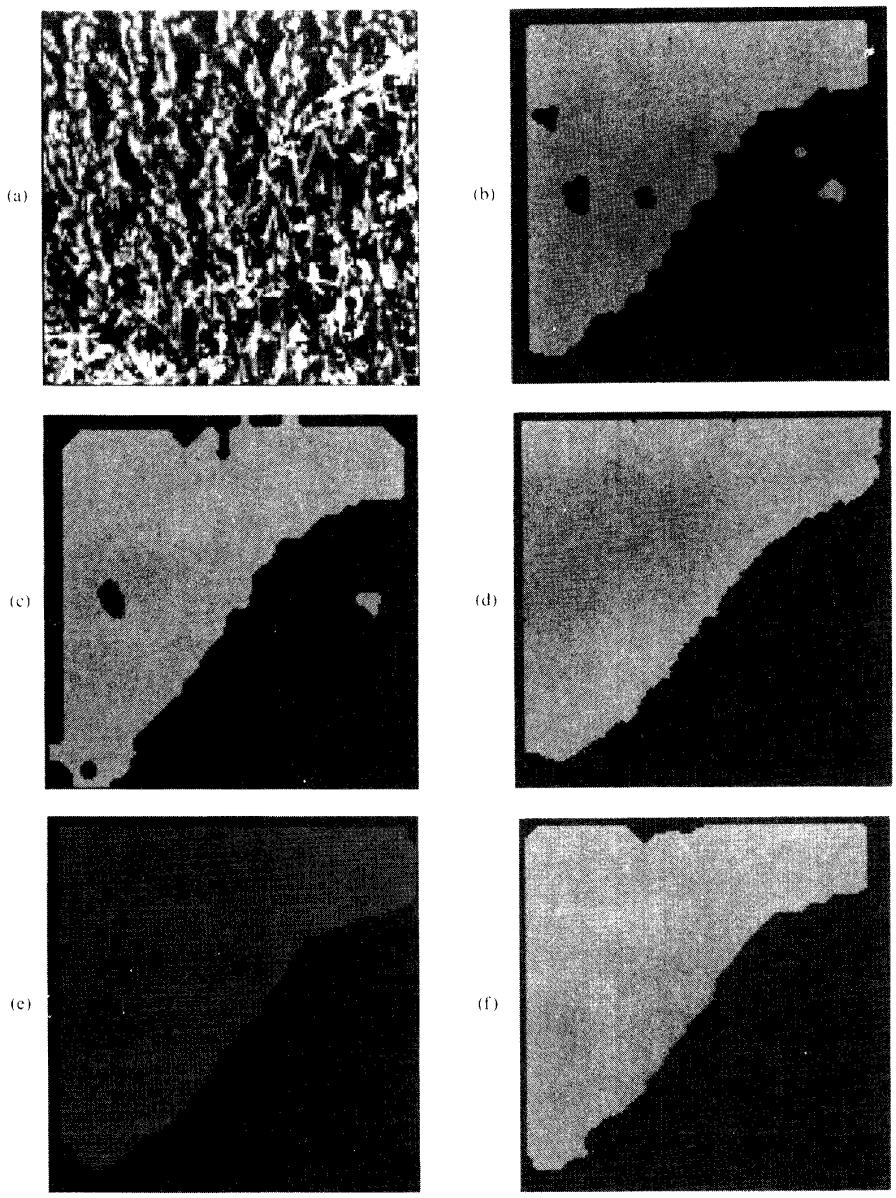


Fig. 2. (a) Original image consisting of two textures. The classification using different algorithms is shown in (b)–(f). (The textures are coded by gray levels.) (b) Deterministic relaxation with maximum likelihood solution as initial condition and (c) with random initial condition. (d) MAP estimate using simulated annealing. (e) MPM solution. (f) Network with stochastic learning.

mance of several algorithms for texture segmentation is studied. The stochastic algorithms obtain nearly optimal results, as can be seen from the examples. We noted that the MRF model helps us to trivially map the optimization problem onto a Hopfield-type neural network. This deterministic relaxation network converges extremely fast to a solution, typically in 20–30 iterations for the 256×256 image. Its performance, however, is sensitive to the initial state of the system and often is not very satisfactory. To overcome the disadvantages of the network, a new algorithm, which introduces stochastic learning into the iterations of the network, was proposed. This helps to maintain a sustained search of the solution space while learning from the past experience. This algorithm combines the advantages of deterministic and stochastic relaxation schemes and it would be interesting to explore its performance in solving other computationally hard problems in computer vision.

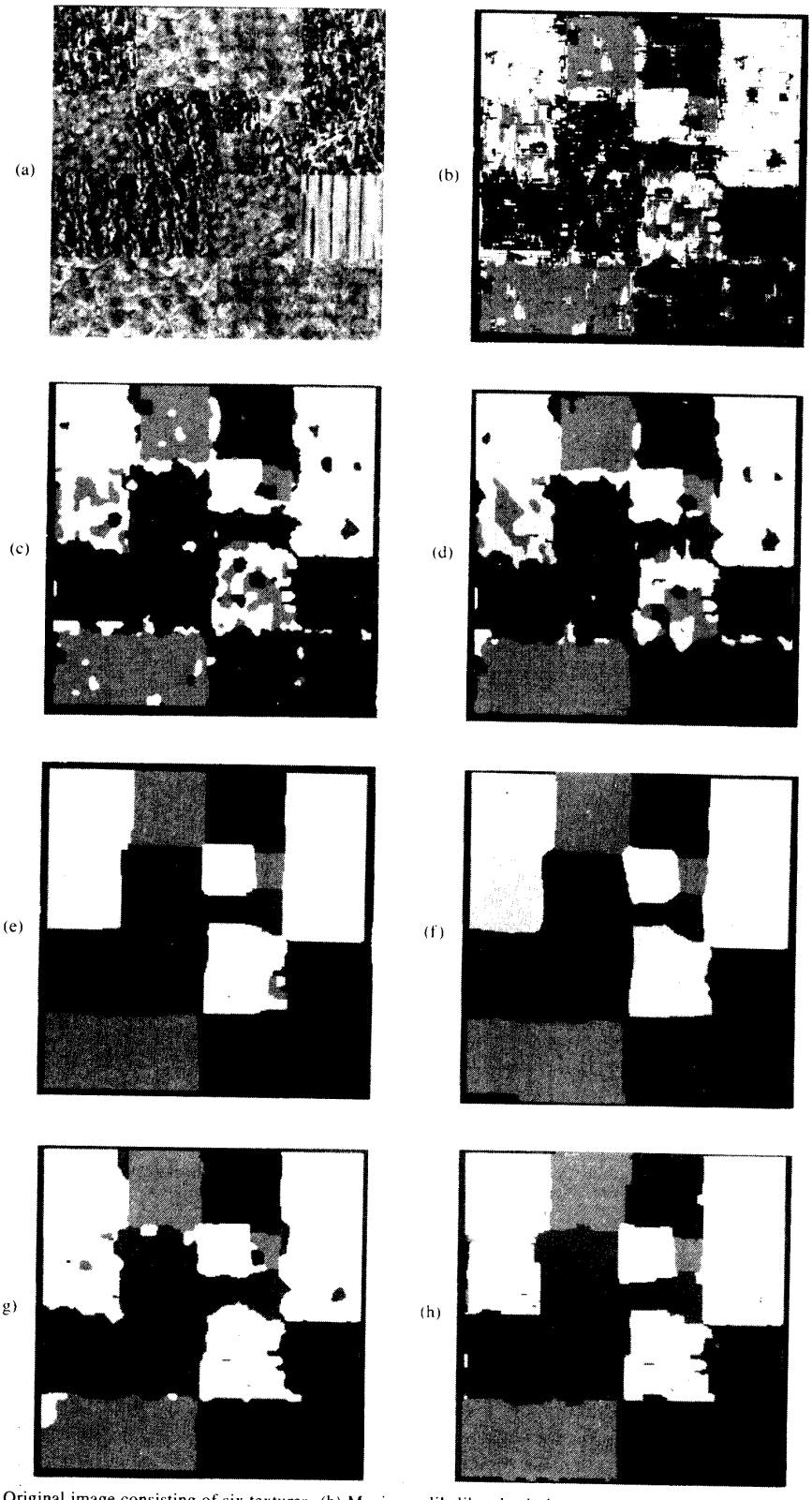


Fig. 3. (a) Original image consisting of six textures. (b) Maximum likelihood solution. (c) Deterministic relaxation with (b) as initial condition and (d) with random initial condition. (e) MAP estimate using simulated annealing. (f) MPM solution. (g) Network with stochastic learning. (h) Hierarchical network solution

TABLE I
PERCENTAGE MISCLASSIFICATION FOR EXAMPLE 2 (SIX CLASS PROBLEM)

Algorithm	Percentage Error
Maximum Likelihood Estimate	22.17
Neural network (MLE as initial state)	16.25
Neural network (Random initial state)	14.74
Simulated annealing (MAP)	6.72
MPM algorithm	7.05
Neural network with learning	8.7
Hierarchical network	8.21

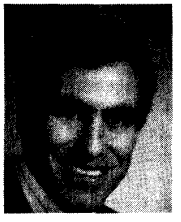
ACKNOWLEDGMENT

The authors wish to thank the reviewers for the many useful comments and suggestions which helped to improve the presentation of this paper.

REFERENCES

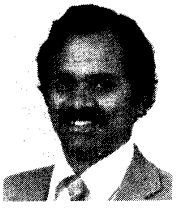
[1] H. Derin and H. Elliott, "Modeling and segmentation of noisy and textured images using Gibbs random fields," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-9, pp. 39-55, Jan. 1987.
[2] F. S. Cohen and D. B. Cooper, "Simple parallel hierarchical and relaxation algorithms for segmenting noncausal Markovian fields," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-9, pp. 195-219, Mar. 1987.
[3] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and Bayesian restoration of images," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-6, pp. 721-741, Nov. 1984.
[4] T. Poggio, V. Torre, and C. Koch, "Computational vision and regularization theory," *Nature*, vol. 317, pp. 314-319, Sept. 1985.
[5] Y. T. Zhou, R. Chellappa, A. Vaid, and B. K. Jenkins, "Image restoration using a neural network," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 36, pp. 1141-1151, July 1988.
[6] Y. T. Zhou and R. Chellappa, "Stereo matching using a neural network," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.* (New York, NY), Apr. 1988, pp. 940-943.
[7] C. Koch, J. Luo, C. Mead, and J. Hutchinson, "Computation motion using resistive networks," in *Proc. Neural Inform. Process. Syst.* (Denver, CO), 1987.
[8] Y. T. Zhou and R. Chellappa, "Computation of optical flow using a neural network," in *Proc. IEEE Int. Conf. Neural Networks*, vol. 2 (San Diego, CA), pp. 71-78.
[9] H. Bulthoff, J. Little, and T. Poggio, "A parallel algorithm for real-time computation of optical flow," *Nature*, vol. 337, pp. 549-553, Feb. 1989.
[10] J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems," *Biolog. Cybernet.*, vol. 52, pp. 114-152, 1985.
[11] J. Besag, "On the statistical analysis of dirty pictures," *J. Roy. Statist. Soc. B*, vol. 48, pp. 259-302, 1986.
[12] K. S. Narendra and M. A. L. Thathachar, "Learning automata—A survey," *IEEE Trans. Syst., Man, Cybern.*, pp. 323-334, July 1974.
[13] U. Grenander, *Lectures in Pattern Theory*, vols. I-III. New York: Springer-Verlag, 1981.
[14] J. L. Marroquin, "Probabilistic solution of inverse problems," Ph.D. thesis, M.I.T., Artificial Intelligence Laboratory, Sept. 1985.
[15] R. Chellappa, "Two-dimensional discrete Gaussian Markov random field models for image processing," in *Progress in Pattern Recognition 2*, L. N. Kanal and A. Rosenfeld, Eds. New York: Elsevier, 1985, pp. 79-112.
[16] G. R. Cross and A. K. Jain, "Markov random field texture models," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-5, pp. 25-39, Jan. 1983.
[17] R. Chellappa and S. Chatterjee, "Classification of textures using Gaussian-Markov random fields," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-33, pp. 959-963, Aug. 1985.
[18] S. Geman and C. Graffigne, "Markov random fields image models and their application to computer vision," in *Proc. Int. Congress of Mathematicans 1986* (Providence).
[19] J. Marroquin, S. Mitter, and T. Poggio, "Probabilistic solution of ill-posed problems in computer vision," in *Proc. Image Understanding Workshop* (Miami Beach, FL), Dec. 1985, pp. 293-309.
[20] B. Gidas, "Non-stationary Markov chains and convergence of the annealing algorithm," *J. Statist. Phys.*, vol. 39, pp. 73-131, 1985.
[21] R. M. Wheeler, Jr., and K. S. Narendra, "Decentralized learning in finite Markov chains," *IEEE Trans. Automat. Contr.*, vol. AC-31, pp. 519-526, June 1986.
[22] M. A. L. Thathachar and P. S. Sastry, "Relaxation labeling with learning automata," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-8, pp. 526-268, Mar. 1986.
[23] S. Chatterjee and R. Chellappa, "Maximum likelihood texture segmentation using Gaussian Markov random field models," in *Proc. Computer Vision and Pattern Recognition Conf.* (San Francisco, CA), June 1985.

B. S. Manjunath (S'88) received the bachelor of engineering degree in electronics from Bangalore University in 1985, and the master of engineering degree in systems science and automation from the Indian Institute of Science in 1987. Since 1987 he has been a Research Assistant at the Signal and Image Processing Institute, University of Southern California, Los Angeles, where he is currently working toward the Ph.D. degree in electrical engineering. His research interests include stochastic learning, self-organization, neural networks, and computer vision.



Tal Simchony (S'86-M'89) was born in Tel Aviv, Israel, on January 18, 1956. He received the B.S. degree in mathematics and computer science and the M.S. degree in applied mathematics from Tel Aviv University in 1982 and 1985, respectively. He then received the Ph.D. degree in electrical engineering from the University of Southern California, Los Angeles, in 1988.

In 1982 he joined ECI Telecom as a Software and Systems Engineer. During the years 1985-1988 he was a Research Assistant at the Signal and Image Processing Institute, USC. He is currently at ECI Telecom as Deputy Chief Engineer working on speech compression algorithms on digital networks. His research interests include optimization, learning, and computer vision.



Rama Chellappa (S'79-M'81-SM'83) was born in Madras, India. He received the B.S. degree (honors) in electronics and communications engineering from the University of Madras in 1975 and the M.S. degree (with distinction) in electrical communication engineering from the Indian Institute of Science in 1977. He then received the M.S. and Ph.D. degrees in electrical engineering from Purdue University, West Lafayette, IN, in 1978 and 1981, respectively.

During the years 1979-1981, he was a Faculty Research Assistant at the Computer Vision Laboratory, University of Maryland, College Park. Since 1986, he has been an Associate Professor in the Electrical Engineering-Systems, University of Southern California, Los Angeles, and in September 1988 he became the Director of the Signal and Image Institute there. His current research interests are in signal and image processing, computer vision, and pattern recognition.

Dr. Chellappa is a member of Tau Beta Pi and Eta Kappa Nu. He is a coeditor of two volumes of selected papers on image analysis and processing, published in the autumn of 1985. He was an Associate Editor for the IEEE TRANSACTIONS ON ACOUSTICS, SPEECH, AND SIGNAL PROCESSING and he is a coeditor of *Computer Vision, Graphics, and Image Processing: Graphic Models and Image Processing*, published by Academic Press. He was a recipient of a National Scholarship from the Government of India during the period 1969-1975. He was the recipient of the 1975 Jawaharlal Nehru Memorial Award from the Department of Education, Government of India, the 1985 Presidential Young Investigator Award, and the 1985 IBM Faculty Development Award. He served as the General Chairman of the 1989 IEEE Computer Society Conference on Computer Vision and Pattern Recognition and the IEEE Computer Society Workshop on Artificial Intelligence for Computer Vision. He was also Program Cochairman of the NSF-sponsored Workshop on Markov Random Fields for Image Processing Analysis and Computer Vision.