# Brief Papers

## Texture Segmentation Using Gaussian–Markov Random Fields and Neural Oscillator Networks

Erdogan Çesmeli and DeLiang Wang

***Abstract*—We propose an image segmentation method based on texture analysis. Our method is composed of two parts. The first part determines a novel set of texture features derived from a Gaussian–Markov random fields (GMRF) model. Unlike a GMRF-based approach, our method does not employ model parameters as features or require the extraction of features for a fixed set of texture types *a priori*. The second part is a two-dimensional (2–D) array of locally excitatory globally inhibitory oscillator networks (LEGION). After being filtered for noise suppression, features are used to determine the local couplings in the network. When LEGION runs, the oscillators corresponding to the same texture tend to synchronize, whereas different texture regions tend to correspond to distinct phases. In simulations, a large system of differential equations is solved for the first time using a recently proposed method for integrating relaxation oscillator networks. We provide results on real texture images to demonstrate the performance of our method.**

***Index Terms*—Dynamical systems, Gaussian Markov random fields, LEGION, neural networks, relaxation oscillators, texture segmentation.**

### I. Introduction

IMAGE segmentation is a fundamental and yet difficult task in machine vision. Texture is one of the visual features playing an important role in scene analysis. Intuitively, it is related to patterned variations of intensity across an image. Texture segmentation in general is composed of two steps, namely, the extraction and the grouping of texture features. Grouping depends on feature correlation, which generally improves with a larger extraction window. However, a large window deteriorates the localization of regions.

There are a number of methods for texture feature extraction. Among them are geometrical, statistical, signal processing, and model-based methods [18]. Markov random field (MRF) models, which fall under the model-based category, have been quite successful for texture modeling and segmentation [4], [7], [3]. They capture local characteristics of an image by assuming a local conditional probability distribution. When this distribution is Gaussian, the model is called Gaussian–Markov random fields (GMRFs) [3]. Model parameters have been proposed to be texture features by assuming that parameter differences among different textures facilitate texture discrimination.

Because of their success in classification, neural networks have been applied to perform texture segmentation. Chellappa *et al.* [3] employed a Hopfield-type network to maximize *aposteriori* probability (MAP) for texture segmentation. Unlike the approaches employing known filter forms, such as Gabor filters [1], [10], [23], [9], Jain and Karu trained a multilayer perceptron (MLP) to obtain texture specific filters and to perform segmentation [11]. The biological relevance of these studies is limited as are the results on real texture. Eom [6] segmented texture images using a neural network whereby a two-dimensional (2-D) moving average model is used to represent textures. Other than the introduction of a new definition for texture features, the study does not suggest a unique approach to the texture segmentation problem. Baldi and Meir [1] proposed a network of coupled oscillators, using Gabor filters as texture feature detectors, for texture discrimination. However, their method is limited by the use of phase oscillators, which cannot synchronize with local coupling, a crucial ability for image segmentation (see [19] for extended discussion on this point).

Temporal correlation [15], [14] has been proposed for binding (grouping) features together, which is important for image segmentation. A special form of temporal correlation is *oscillatory correlation*, whereby the basic unit is a neural oscillator [21], [19]. The discovery of synchronous oscillations in the visual cortex [5], [8] provides neurophysiological support to this representation. LEGION [21], [19], which is based on the idea of oscillatory correlation, has been proposed to deal with image segmentation. When a LEGION network runs, oscillators corresponding to the same region tend to attain the same phase, which is different from those of other regions. Unlike many other image segmentation approaches, LEGION networks represent a dynamical systems approach. Oscillators are active agents that are driven by external stimulation and influence each other through coupling. Segmentation is achieved in a continuous-time process that corresponds to parallel and distributed computation. LEGION has been rigorously shown to be capable of both rapid synchronization and desynchronization, a crucial property for dynamical systems performing image segmentation [19].

In this paper, we describe a texture-based image segmentation method combining GMRF modeling and neural oscillator networks. Although our texture features are not GMRF model

Manuscript received June 23, 2000; revised and November 9, 2000. This work was supported in part by the NSF under Grant IRI-9423312. The work of D. Wang was supported in part by an ONR Young Investigator Award.

E. Çesmeli is with the Biomedical Engineering Center, The Ohio State University, Columbus, OH 43210 USA.

D. Wang is with the Department of Computer and Information Science and Center for Cognitive Science, The Ohio State University, Columbus, OH 43210 USA.

Publisher Item Identifier S 1045-9227(01)02050-1.

parameters, they still have a local dependence structure, a notion essential for texture segmentation (for more details, see Section II). The extraction of features is followed by filtering for noise suppression. Finally, local couplings in LEGION are defined based on filtered features. We emphasize the notion of spatial context by allowing only a dynamic set of neighbors to determine local couplings and reinforce desynchronization by an additional inhibitor. In previous studies, an algorithm was designed to mimic the network behavior for image segmentation [21]. In order to assess the network dynamics more faithfully, our study directly integrates a large number of differential equations governing the behavior of oscillators. Note that such integration is computationally very expensive when dealing with real images typically having $128 \times 128$ pixels or more. Our integration is made feasible by the singular limit method that is recently proposed specifically for integrating relaxation oscillator networks [12], of which our network is an example. The method integrates relaxation oscillator networks in the singular limit (more details in Section II) and gives rise to remarkable speedup compared to the commonly used Runge–Kutta method.

In sum, this paper proposes a set of texture features derived from a GMRF model and, for the first time, demonstrates image segmentation by integrating a large set of differential equations corresponding to a biologically plausible oscillator network.

The rest of this paper is organized as follows. Section II provides the description of our method. We demonstrate its performance on real images in Section III. Some comparisons are drawn in Section IV, and we conclude in Section V.

## II. Proposed Model

We outline our method in Fig. 1. First, we extract texture features at each site. After filtering them for noise suppression, local couplings are determined using a similarity measure defined in feature space. Finally, LEGION encodes segmented regions by different oscillator phases.

### A. Texture Features

The idea of GMRF in image processing is to represent local image characteristics in terms of a local conditional probability distribution which is assumed to be Gaussian [4]. The size and the topology of the local dependence are designated as the order of the model. For example, a second-order GMRF contains eight nearest neighbors. Let $\Omega = \{(i, j), 1 \le i, j \le M\}$ denote the set of grid points in the $M \times M$ lattice corresponding to the pixels in the image, and $\{L_s, s \in \Omega\}$ and $\{Y_s, s \in \Omega\}$ be a set of labels and intensities (with zero mean), respectively. $N_c(s)$ is the symmetric second-order clique neighborhood at site $s$. Assuming that $s$ and its neighbors have the same label, $l$, the conditional probability density over intensity is

$$P(Y_s = y_s | Y_r = y_r, r \in N_c(s), L_s = l) = \frac{\exp(-U'(Y_s = y_s | Y_r = y_r, r \in N_c(s), L_s = l))}{Z'(l | y_r, r \in N_c(s))}. \quad (1)$$

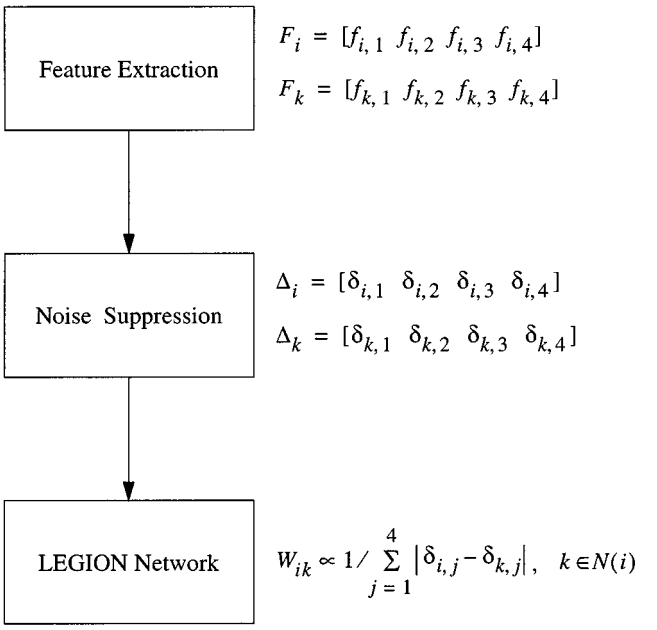


Fig. 1. Flow diagram for texture segmentation, which is composed of three stages: feature extraction, noise suppression, and LEGION network. Next to each stage are shown the related variables.

Here $Z'(l | y_r, r \in N_c(s))$ is the partition function of the conditional probability density, corresponding to the sum of the numerator for all possible realizations of $N_c$ and

$$U'(Y_s = y_s | Y_r = y_r, r \in N_c(s), L_s = l) = \frac{1}{2\sigma_l^2} \left( y_s^2 - 2 \sum_{r \in N_c(s)} \theta_{r,s}^l y_r y_s \right) \quad (2)$$

where $\sigma_l$ is the variance and $\theta_{r,s}^l$s are GMRF model parameters of label $l$ satisfying $\theta_{r,s}^l = \theta_{r-s}^l = \theta_{s-r}^l = \theta_\tau^l$. Assuming conditional independence, the joint probability in a window $R(s) = u \times u$ centered at site $s$ is given by

$$P\left(\vec{Y}_s \middle| L_s = l\right) = \frac{\exp\left(-U\left(\vec{y}_s \middle| L_s = l\right)\right)}{Z(l)} \quad (3)$$

where $Z(l)$ is the partition function, $\vec{y}_s$ represents the intensity array in $R(s)$, and

$$U\left(\vec{y}_s \middle| L_s = l\right) = \sum_{k, k+\tau_j \in R(s)} U'(Y_k | Y_{k+\tau_j}, \tau_j \in N^*, k + \tau_j \in R(s), L_s) = \frac{1}{2\sigma_l^2} \sum_{k, k \pm \tau_j \in R(s)} \left( y_k^2 - \sum_{\tau_j \in N^*} \theta_j^l y_k (y_{k+\tau_j} + y_{k-\tau_j}) \right). \quad (4)$$

Here, $N^* = \{\tau_1, \tau_2, \tau_3, \tau_4\} = \{(0, 1), (1, 0), (1, 1), (-1, 1)\}$ is a set of shift vectors corresponding to the second-order GMRF model.

Among several methods for estimating GMRF model parameters are maximum likelihood (ML) estimation, coding scheme, and least square estimation [3]. Our texture features are derived from GMRF parameters estimated by the least square method, which is easy to compute and has a reasonable efficiency [3].

We assume a second-order GMRF model where $N_c = 3 \times 3$ and consider an estimation window, $R(s) = u \times u$ at site $s$, which contains a single texture. The least square estimate of a GMRF model parameter vector, $\hat{\Theta} = [\hat{\theta}_1\ \hat{\theta}_2\ \hat{\theta}_3\ \hat{\theta}_4]^T$, is given by [3]

$$\hat{\Theta} = \left[\sum_{r,\, r\pm\tau_j \in R(s)} Q(r)Q(r)^T\right]^{-1} \left[\sum_{r,\, r\pm\tau_j \in R(s)} Q(r)y_r\right] \tag{5}$$

where $Q(r) = [y_{r+\tau_1} + y_{r-\tau_1}, \cdots, y_{r+\tau_4} + y_{r-\tau_4}]^T$ and the intensity variance, $\sigma$, within $R$ is

$$\sigma = \frac{1}{u^2} \sum_{r,\, r\pm\tau_j \in R(s)} [y_r - \hat{\Theta}^T Q(r)]^2. \tag{6}$$

A general GMRF-based approach employs GMRF model parameters, $\hat{\theta}_j$s, and the variance, $\sigma$, as its texture feature set (see, e.g., [3]). We assume the same model and employ the same technique to estimate the model parameters, $\hat{\theta}_j$s, and the variance, $\sigma$. However, instead of directly using them as features, we utilize them in the derivation of a different set of texture features, $f_j$s:

$$f_j = \frac{1}{u^2} \sum_{r,\, r\pm\tau_j \in R(s)} [y_r - \hat{\theta}_j Q_j(r)]^2 \tag{7}$$

where $Q_j(r)$ is the $j$th component of $Q(r)$ and $j = 1, 2, 3$, or 4. Due to the similarity between (6) and (7), $f_j$s behave like variances in four directions defined by the shift vectors, $\tau_j$s. Because of directionality, $f_j$s show orientation selectivity, an essential property in texture segregation. Also, note that our purpose is not to synthesize a texture type given its samples, but to assess a discrimination measure among different textures. In Fig. 2, we display both the set of our features and that of the variance and GMRF parameters for three different texture types. These textures are from the image shown in Fig. 2(a). First, we determine both sets at each site using an estimation window, $R = 9 \times 9$. Next, we obtain the distribution of each feature within a square area of the size $21 \times 21$ placed in each textured region. We plot the distributions (histograms) of $f_j$s and those of $\hat{\theta}_j$s and $\sigma$ in Fig. 2(b) and (c), respectively. Each graph consists of the distributions of only one feature for all textures. For example, the top graph in Fig. 2(b) shows the distribution of $f_1$ for all textures simultaneously. Likewise, the second graph in Fig. 2(c) depicts the distribution of $\theta_1$ for all textures.

We observe two major differences between the two sets. First, $f_j$s are more discriminatory; in this example, they result in three nonoverlapping distributions, each corresponding to one texture type [Fig. 2(b)]. $\hat{\theta}_j$s and $\sigma$ of the same textures, however, are similar [Fig. 2(c)]. In particular, the distributions of $\hat{\theta}_j$s almost completely coincide for the texture types in the upper right (solid line) and the left (dotted line) regions of the image, as seen in the lower four graphs in Fig. 2(c). Second, when we compare the top graph with the other four graphs in Fig. 2(c), we see that the range of $\sigma$ is much larger than those of $\hat{\theta}_j$s. As a result, a similarity measure defined on $\hat{\theta}_j$s and $\sigma$ has to consider their range differences. These differences are usually offset by normalizing each parameter by its standard deviation. However, this requires the prior knowledge of standard deviations for $\hat{\theta}_j$s and $\sigma$ for all texture types. Without normalization, $\sigma$ would play too big a role in the discrimination of different textures. Since $f_j$s have about the same range as shown in Fig. 2(b), our method does not require normalization.

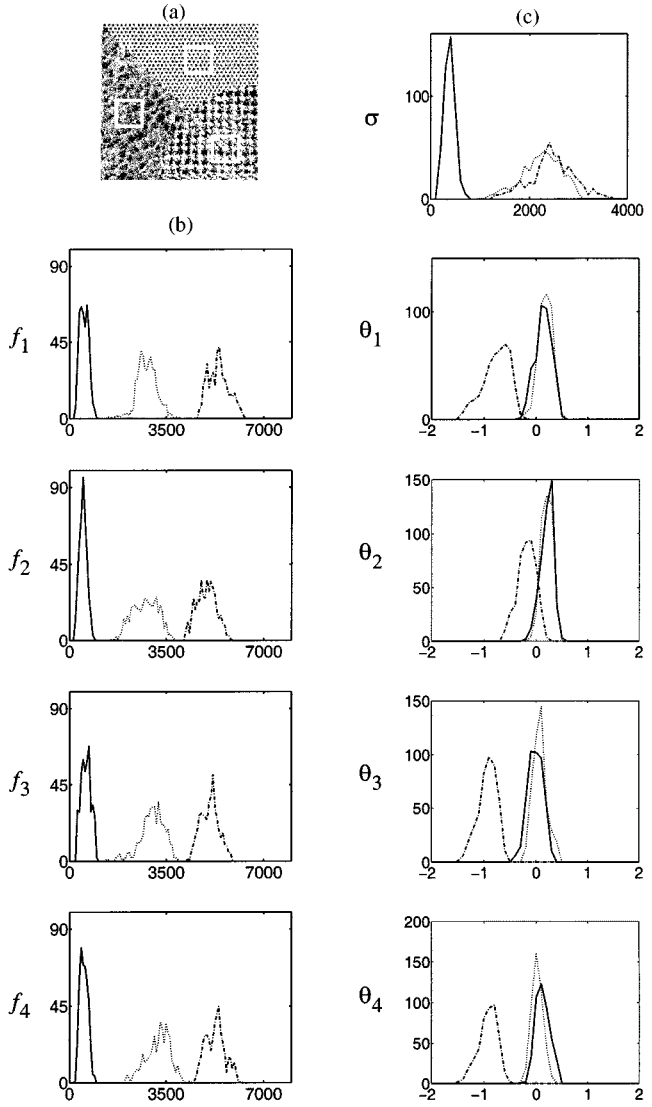


Fig. 2. (a) Square windows, each sampling a different texture type, show the areas where the distributions (histograms) of features are obtained. Each graph corresponds to the distribution of one feature for different texture types. Solid, dot, and dash-dot lines correspond to the distributions in the squares located in the upper right, left, and lower right regions, respectively. (b) Distributions of $f_j$s. (c) Distributions of the variance, $\sigma$, and GMRF model parameters, $\hat{\theta}_j$s.

### B. Noise Suppression

As outlined in Fig. 1, texture feature vectors, $F = [f_1\ f_2\ f_3\ f_4]$, are filtered for noise suppression, following their

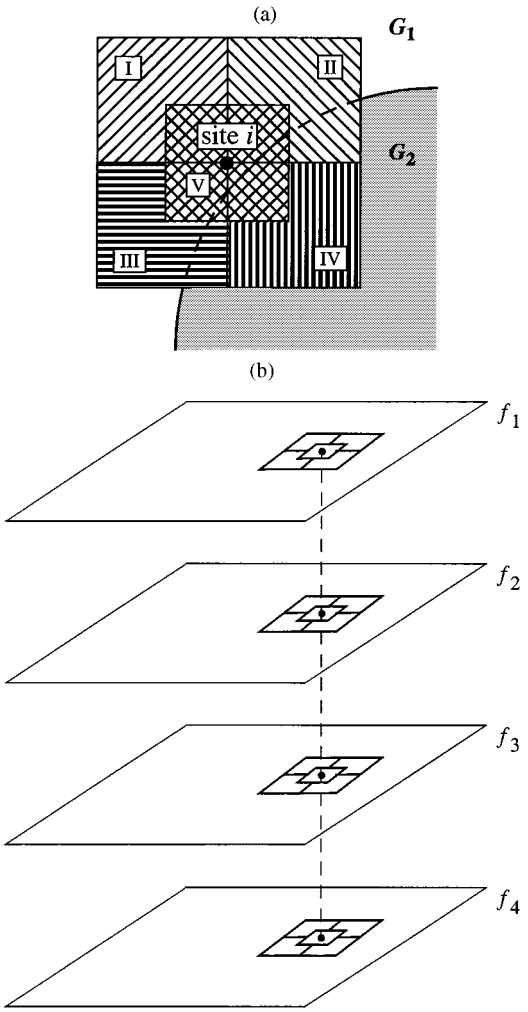


Fig. 3. (a) Filtering a single image using an edge preserving noise suppressing quadrant filter (EPNSQF). Five square subwindows of the same size are formed within each filtering window which is centered at the site $i$ shown as a black dot. In this case, the center location is in region $G_1$ and close to the boundary of region $G_2$. When processing is performed, region information is not known. (b) Filtering multiple images with EPNSQF. Four filtering windows, each represented by a square, are formed at the same location on four feature images.

extraction in an estimation window, $R$. Since the feature vector has four components, we can view them as a set of four feature images. For filtering, we employ an extended version of an edge preserving noise suppressing quadrant filter (EPNSQF), which was originally proposed to process one image at a time [20]. Filtering starts by forming five square subwindows, $N_{s,j}$ on feature image $f_j$, within a filtering window centered at each site in an image, as shown in Fig. 3(a). The mean and the variance of intensities are calculated within each subwindow on each feature image [Fig. 3(b)]. Note that variance calculations in filtering are different from the variance $\sigma$ estimated in (6). The variances of subwindows covering the same locations are summated to obtain their total variance. The set of subwindows having the minimum total variance is selected as the winners. As a result, the means of $f_i$s within the winner subwindow set are assigned to the filtered feature vector, $\Delta_i = [\delta_{i,1}\ \delta_{i,2}\ \delta_{i,3}\ \delta_{i,4}]$ where $\delta_{i,j} = (1/|N_{s,j}|)(\sum_{k\in N_{s,j}} f_{j,k})$, $N_{s,j}$ is the winner subwindow on feature image $f_j$, and $|N_{s,j}|$ is the number of pixels in it. Similar to feature images, each component of filtered feature vectors for all sites forms one filtered feature image. In EPNSQF, blurring due to averaging operation across the region boundaries is minimized by choosing the filter outputs as the mean intensities of only those subwindows with the smallest variance.

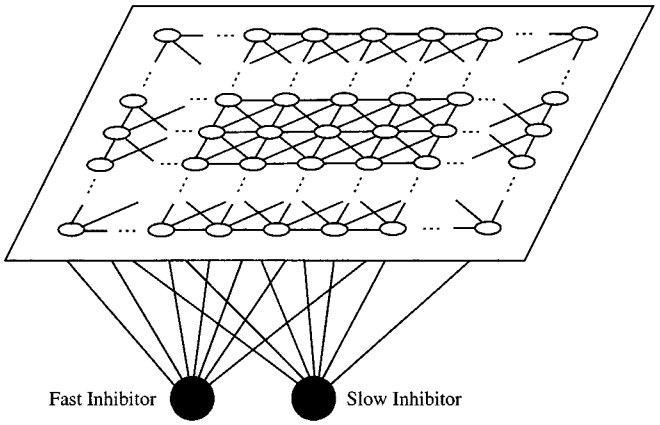


Fig. 4. Architecture of a LEGION network with eight-nearest neighbor coupling. The fast and the slow inhibitors are indicated by two black circles.

### C. Oscillator Networks

With filtered features, we briefly describe our LEGION network for texture segmentation. The part of our network that has been previously published is omitted here; the reader is referred to [22] for a complete description. The network is a 2-D array of relaxation oscillators with two global inhibitors, as shown in Fig. 4. The limit cycle trajectory of a single relaxation oscillator in phase space is composed of four pieces: two pieces on a slow time scale and two pieces on a fast time scale. The two slow pieces are called the left branch (LB) and the right branch (RB). The lower end of LB, where an oscillator jumps from LB to RB on the fast time scale is called the left knee (LK). Similarly, the upper end of RB is the right knee (RK) where an oscillator jumps from RB to LB on the fast time scale.

The coupling to each oscillator $i$, $S_i$, is a sum of four terms: 1) local excitatory coupling with the neighboring oscillators; 2) a potential term encoding the differences between noisy and homogeneous parts; 3) fast global inhibition; and 4) slow global inhibition

$$S_i = WH\left(\sum_{k\in N(i)} W_{ik}H(x_k) - \theta\right) + W_pH(p_i - 0.5) - W_fH(z_f) - W_sH(z_s). \tag{8}$$

$H$ represents the Heaviside step function.

Local excitatory coupling between a filtered feature vector at site $i$ and those at neighboring sites $k \in N(i)$ takes place via dynamic connection weights

$$W_{ik} = \hat{F} \Big/ \left(\sum_{j=1}^{4} |\delta_{i,j} - \delta_{k,j}|\right) \tag{9}$$

where $\delta_{i,j}$ and $\delta_{k,j}$ are the $j$th component of filtered feature vectors, $\Delta_i$ and $\Delta_k$, corresponding to oscillators $i$ and $k$, respectively. Note that a small value is always added to a denominator to avoid division by zero. $\hat{F}$ is a normalization factor, determined by

$$\hat{F} = F_a(F_i \cdot F_N)^{1/4} \tag{10}$$

where

$F_a = \sum_{k \in N(i)} H(x_k)$ number of active neighbors;
$F_i = \sum_{j=1}^{4} \delta_{i,j}^2$ magnitude of $\Delta_i$;
$F_N$ magnitude of the average feature vector of the active oscillators in $N(i)$.

$$F_N = \sum_{j=1}^{4} \left[ \sum_{k \in N(i)} \delta_{k,j} H(x_k) \Big/ F_a \right]^2. \tag{11}$$

Note that $\hat{F}$ assesses the local feature changes independent of their magnitudes and considers feature vectors of only active neighbors.

The motivation behind (9)–(11) is to have a large dynamic weight $W_{ik}$ when $\Delta_i$ is similar to $\Delta_k$.[1] Large $W_{ik}$ tends to recruit oscillator $i$ to synchronize with oscillator $k$. To further illustrate local interactions, let us consider an oscillator at site $i$ and assume that, at a particular time, only a subset of its neighbors $N(i)$ belong to an active region as depicted in Fig. 5(a). Note that only these active neighbors contribute to the total similarity, making the activity of oscillator $i$ depend on a neighborhood that evolves dynamically. Accordingly, oscillator $i$ is recruited by the active region only if its similarity to its dynamic neighborhood inside $N(i)$ is larger than $\theta$.

A main rationale for using dynamic neighborhood is to deal with large variations across region boundaries. With dynamic neighborhood, only the oscillators belonging to the currently active region are used in the grouping process. Thus, within a neighborhood, similarity is measured separately within each region, producing better localized boundaries than does fixed neighborhood. Another benefit of using dynamic neighborhood occurs when a feature-defined boundary between two regions is not sharp. In this case, dynamic neighborhood plays an important role in grouping as illustrated in Fig. 5(b). Take, for example, oscillator $i$, which is located in the left region but close to the boundary between the two distinct regions. Let us divide $N(i)$ into $R_1$ and $R_2$, as shown in the figure. Assume that $\Delta_i$ is similar to the average feature vector of its neighbors in $R_1$, but not that of $R_2$. At the same time, $\Delta_i$ is also similar to that of a small group of neighbors $R_3$ within $R_2$. A possible scenario is that $R_3$ is active simultaneously and hence recruits $i$ to join $R_3$. But, when the oscillators in $R_1$ are synchronized, $i$ is grouped (correctly) with $R_1$ and hence, with the left region. When $i$s neighbors in $R_3$ get synchronized with the rest of $R_2$, $R_2$ as a whole will not recruit $i$ because $\Delta_i$ is not similar to the average feature vector of $R_2$. Thus, $i$ will not be recruited by the right region anymore, and its neighbors in $R_1$ win the competition for the oscillator. Without dynamic neighborhood, however, oscillator $i$ groups wrongly with the right region; even worse, it likely causes the merging of the left and the right regions. Thus, dynamic neighborhood makes segmentation more robust to local variations.

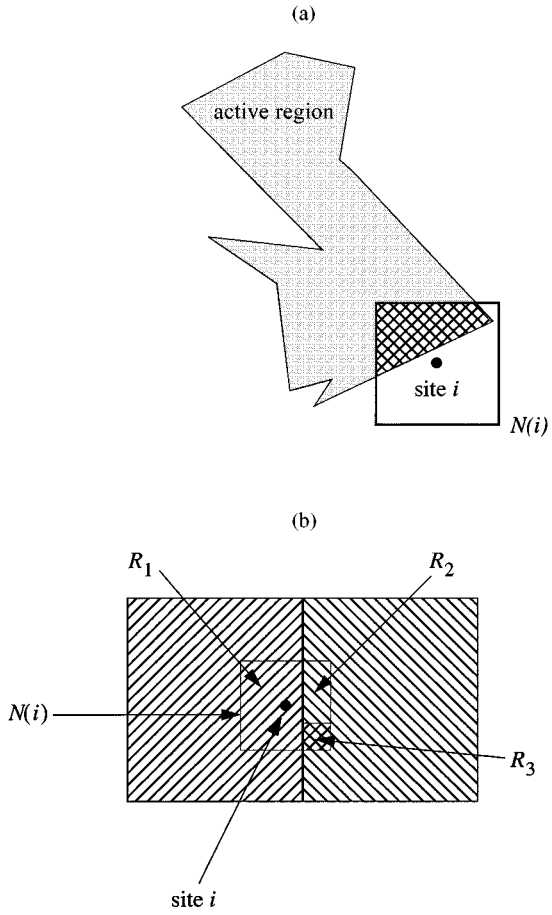


Fig. 5. (a) The calculation of the total similarity for an oscillator at site $i$. Gray region represents a group of oscillators that are currently active. The square region around the site $i$ is the neighborhood $N(i)$, and the intersection of $N(i)$ with the active region is the set of oscillators forming the dynamic neighborhood of oscillator $i$. Feature vectors of only this dynamic neighborhood are included in the calculation of the similarity. (b) The effect of dynamic neighborhood on synchronization when texture boundaries are not sharply defined. $\Delta_i$ is similar to the average feature vector of its neighbors in $R_1$ and $R_3$ but not to that of those in $R_2$, where $N(i) = R_1 \cup R_2$ and $R_3 \subset R_2$.

The second term of (8), where $p_i$ is referred to as potential, is introduced to distinguish between relatively homogeneous regions and noisy fragments [22]. Oscillators with high potentials are called *leaders*. Oscillators corresponding to a noisy fragment will stop oscillation after a brief initial period. These oscillators together form the background.

In addition to the synchronization within a group of oscillators that have similar features, different groups need to be desynchronized to complete the segmentation. We extend the notion of global inhibition by introducing a slow inhibitor $z_s$ [see (8)] along with a fast inhibitor $z_f$ introduced by Terman and Wang [19]. Slow inhibition is needed due to dynamic neighborhood. When the oscillators in a currently active region are jumping down to LB, it may cause other oscillators near the region to

[1]Other formulations may also achieve a similar effect; our particular choice is mainly motivated by simplicity.

jump up before the entire region has jumped to LB. Again, consider Fig. 5(b). It is possible that, during the process for $R_2$ to jump down to LB, $R_3$ is the last part of $R_2$ to jump down. When this happens, the similarity between $R_3$ and oscillator $i$ can recruit $i$ to jump up to RB before the right region completes its jumping down. This may lead to the merging of the left and the right regions. To prevent this from happening, the slow inhibitor rises slowly but decays quickly. Due to its slow rise, it does not influence the process of grouping when oscillators recruit each other to jump up. However, when an active region tracks RB on the slow time scale, slow inhibition increases and makes it impossible for other oscillators that are not active to jump to RB. This continues until the entire active region has jumped down, and after that slow inhibition decays quickly. Formally, $z_s$ is defined by

$$\frac{dz_s}{dt} = \varepsilon\kappa \left[ \sum_k H(x_k) - z_s \right]^+ - z_s. \tag{12}$$

Here $[x]^+ = x$ if $x \geq 0$ and $[x]^+ = 0$ otherwise. The parameter $\varepsilon$ is a small positive number that characterizes relaxation oscillations [19], $\kappa$ is a parameter, and $\varepsilon\kappa$ together causes slow rise. But the decay part of the inhibitor is in fast time.

One of the challenges of simulating a LEGION network for segmenting real images is the amount of numerical computation needed. To reduce it, an algorithm was extracted and applied to intensity image segmentation in a previous study [22], [24]. Although the algorithm exhibits the essential properties of the network, it does not completely capture network dynamics, and does not, for example, exhibit the segmentation capacity which refers to the property that oscillatory dynamics can separate only a few segments due to finite temporal resolution [22], [24].

To faithfully exhibit oscillatory dynamics in segmentation, we employ the singular limit method [13] for integrating large relaxation oscillator networks. This recently proposed method is based on the observation that a single relaxation oscillator travels along either LB or RB slowly, but jumps quickly between them. It is this motion structure that is exploited in the singular limit method. In the singular limit $\varepsilon \to 0$, the limit cycle reduces to two analytically solvable intervals of LB and RB [13], and two instantaneous jumps. In a network of such oscillators, time can be divided into consecutive intervals that are separated by jumps. During each interval, analytic solutions are used to march the system in large time steps. At each jumping instance, iterative operations are executed so that appropriate phase shifts are made and the nullclines of oscillators are updated. As analyzed in [12], the singular limit method produces remarkable reduction of computing time compared to commonly used integration methods such as that of Runge–Kutta.

We summarize the singular limit method employed here into the algorithm below.

```
0. Determine the filtered feature vectors
   at all locations and form the LEGION
   network accordingly.
1. Randomize the location of each oscil-
   lator in phase space; both fast and slow
   inhibitors are initialized accordingly.
2. Determine the time that it takes the
   closest oscillator to reach either LK or
   RK.
3. March oscillators, namely update their
   locations in the phase space, by the
   time determined in Step 2 and update the
   inhibitors and potentials of the oscil-
   lators accordingly.
4. Jump the oscillator(s) at the knee,
   which is a leader when there is no ac-
   tive oscillator, to the opposite branch
   and update the inhibitors.
5. Iterate until no further jumping oc-
   curs in one iteration: update each os-
   cillator's location in phase space due
   to jumping in the previous iteration; if
   the updated location is beyond either LK
   or RK jump the oscillator to the oppo-
   site branch.
6. Update the lateral potential of each
   oscillator.
7. Repeat Steps 2–6 until a predetermined
   simulation time is reached. Note that
   the oscillators that are on RB together
   form the segment of the textured region.
```

A similar but simpler system has been rigorously analyzed by Terman and Wang [19], [22], [24], and the analysis ensures both the stability of limit cycle solutions and fast rates of synchronization and desynchronization. Though we have not carried out a rigorous analysis of our extended LEGION system, our extensive numerical simulations show that its behavior is consistent with the Terman–Wang analysis.

In our LEGION network, the weights except for $W_{ik}$s are set to $W = 1.0$, $W_p = 0.4$, $W_f = 0.5$ and $W_s = 0.6$, respectively, in all simulations. The corresponding neighborhoods for the first two terms of (8) are $N = 7 \times 7$ and $N_p = 11 \times 11$. We use $N_c = 3 \times 3$ and $R = 9 \times 9$ for the texture model, a subwindow of the size $N_{s,j} = 21 \times 21$ in the filtering window. The values for other network parameters include $\theta_p = 11{\cdot}11{\cdot}0.75 = 90.75$, $\kappa = 1.3$ and $\theta = 2.0$. The threshold, $\theta$, is the only parameter that is adjusted for different images. In our simulations, it is always within 2.0 and 3.6.

## III. Results

We demonstrate our method using a set of nine images of the size $128 \times 128$. The set includes an indoor image from Hofmann *et al.* [9], textures from Brodatz Photography Album [2] and the MIT Vision Texture Image Archive [16].

We illustrate the segmentation process using the image shown in Fig. 6(a), which is composed of four quadrant regions each with a different texture. Assuming a second-order GMRF model, we extract texture features in an estimation window, $R = 9 \times 9$, forming a texture feature vector at each site.

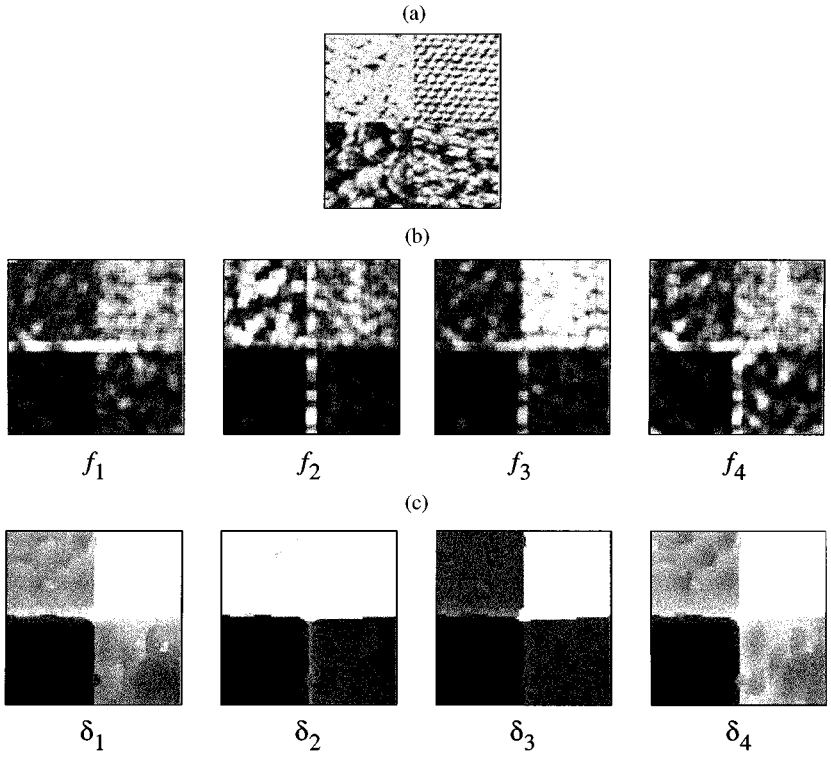


Fig. 6. Texture feature extraction and noise suppression. (a) Input image composed of four quadrant regions each with a different texture. (b) Four feature images. Each one is formed by one component of the texture feature vector, $F = [f_1\ f_2\ f_3\ f_4]$, for all sites. (c) Four filtered feature images, each corresponding to one component of $\Delta = [\delta_1\ \delta_2\ \delta_3\ \delta_4]$, are obtained after filtering the feature images in (b) for noise suppression.

Because of the image border effects, the resulting feature images have the size of 120 $\times$ 120, as shown in Fig. 6(b). Following their extraction, features are filtered in a window of the size of 41 $\times$ 41 with a subwindow of the size 21 $\times$ 21 resulting in four filtered feature images of the size 80 $\times$ 80, as depicted in Fig. 6(c). Note that while small variations within different regions are removed, region boundaries are mostly preserved. After defining the couplings in our LEGION network, the segmentation is performed in a network of 80 $\times$ 80 oscillators. Snapshots of the network activities at different stages of the dynamic evolution are provided in Fig. 7(a). All oscillators start on LB along which their initial locations are determined by their initial phases. For computational efficiency, we preset initial phases of the oscillators in accordance with their filtered feature vectors; oscillators having similar filtered features start at similar phases. After a few cycles following the initialization, a group of oscillators becomes highly active while the others keep low activity levels, as depicted in the leftmost snapshot in Fig. 7(a), indicating the segregation of the lower right region from the other regions in terms of phase. A short time later, the oscillators stimulated by the lower left region become active and segregate from the rest as shown in the second image of Fig. 7(a). Subsequently, the oscillators of the other two regions become active in turn, completing the segmentation.

Once synchronized within each group and desynchronized among different groups, oscillators in our LEGION network keep their relative phase relations unless the input image is changed. To provide a complete picture of dynamic evolution, Fig. 7(b) depicts the temporal activity of each oscillator throughout the simulation. The activities of the two inhibitors are plotted in addition to those of the oscillators in the four regions and in the background, which is composed of those oscillators that become silent shortly after the network starts running. Aligned activities of the oscillators indicate the synchronization of the corresponding regions. When comparing the top four traces, it is easy to see that the oscillators of different regions achieve high activities at different times, namely, they are desynchronized. Notice that the dynamic evolution of the inhibitors is consistent with the activation of different groups of oscillators, as depicted in the lower two traces. The fast inhibitor has sequences of four consecutive activation traces, each of which corresponds to the synchronization of one group of oscillators. Similarly, the slow inhibitor shows four gradually rising traces in concert with the activity of the fast inhibitor. The inhibitors together assure proper desynchronization among different groups of oscillators.

$\delta_j$s in Fig. 6(c) do not equally distinguish different textures. $\delta_2$, for instance, can discriminate the upper right texture and the lower right texture but not the upper left texture and the upper right one. On the other hand, $\delta_3$ can distinguish the latter pair. Since $\delta_j$s have comparable ranges, distinct discrimination abilities from $\delta_j$s are readily integrated in our feature detection method.

Five additional input images and their segmentation results are provided in Fig. 8. Except for the threshold, $\theta$, all results are obtained with the same set of parameters as used for the image in Fig. 6(a). Input images are composed of four quadrant regions each with a different texture. Next to each image is its segmentation result which is displayed as a gray level image,

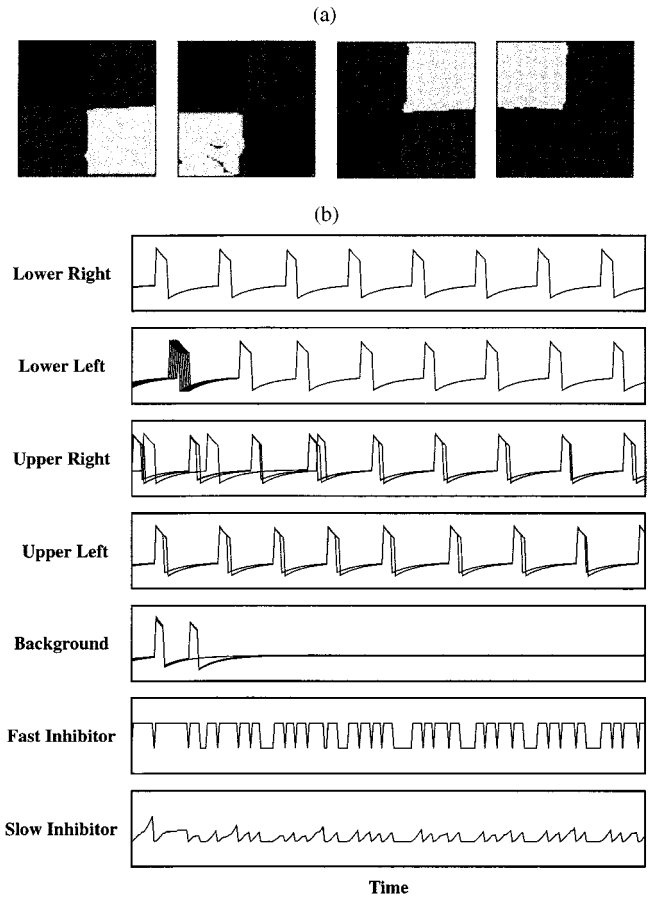


Fig. 7. The segmentation result for the input image shown in Fig. 6(a). (a) Snapshots of the network activity. Oscillators corresponding to the lower right region reach high activity level while the others have low levels of activity. A short time later, the oscillators stimulated by the lower left region reach high values. Subsequently, the oscillators representing the other two regions become highly active in turn. (b) Temporal activities of the oscillators in the four regions and in the background, and those of the fast and the slow inhibitors throughout the simulation. $\theta = 2.0$.

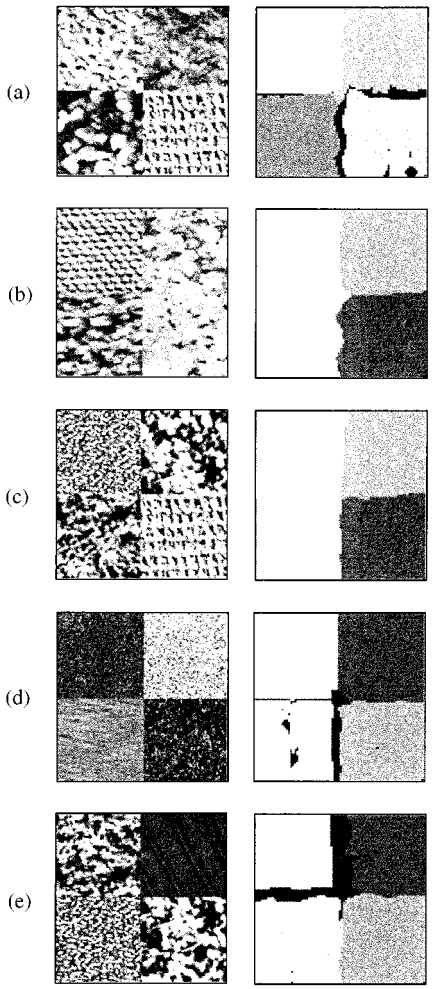


Fig. 8. (a)–(e) Segmentation results. The left image in each case is an input image of the size $128 \times 128$. Next to each input image is the corresponding segmentation result, where segmented regions are assigned different gray levels. $\theta = 3.0, 2.9, 2.8, 2.8,$ and $3.2$ for (a)–(e), respectively.

instead of temporal activity of the network. Sites corresponding to the same group of synchronized oscillators are colored with the same gray level while different groups are assigned different gray levels. Finally, sites corresponding to the oscillators in the background are indicated as black.

In all cases of Fig. 8, our method is able to segment four regions successfully except for some errors along the boundaries. In Fig. 8(a), except for the lower right region and where all regions meet, the segmentation result is very close to the ground truth. The fact that GMRF modeling is not very good at regular textures reveals itself for the lower right region which is relatively more regular than the other regions. The center area is prone to misclassification partially due to filtering. When the filtering window for noise suppression is in this area, it is more likely to choose a wrong subwindow set for the output assignment than elsewhere in the image. Another source of error is the texture extraction process. Our method makes errors usually along the boundaries because the estimation windows contain two different texture types, and therefore the estimated features are different from that of either of the two textures. Fig. 8(b) and (c) have relatively better segmentation results than the other examples in Fig. 8. Aside from the errors along the boundaries, segmented regions do not have holes except for the lower right region in Fig. 8(b). Note that even though the texture in the lower right regions of Fig. 8(a) and (c) is the same, the segmentation results are not the same for this texture type. This is because of the difference in the values of $\theta$. Holes in the lower right region of Fig. 8(a), which are part of the background, do not appear in the result of Fig. 8(c), since the threshold for the latter image tolerates more dissimilarity. In Fig. 8(d) and (e), similar to the previous results, errors occur along the boundaries and the central region. In Fig. 8(d), even though all regions seem to be uniform to the human eye, our method identifies inhomegeneities in the lower left region. This is an example illustrating that GMRF-based texture features do not always agree with perception. Finally, Fig. 8(e) illustrates another problem due to the size of the texture elements. The upper right region in the image has diagonal line structures. Since these structures have larger sizes than that of the estimation window, our texture features do not capture them well, resulting in a large undecided region (background) between the upper left and the upper right regions. This does not present a problem

along the boundary between the upper right and the lower right regions since the two regions have a relatively larger intensity difference as compared to that between the upper left and the upper right regions.

An additional image having different boundary shape and orientations among three textured regions is segmented and the result is shown in Fig. 9(a). One of the three regions is a quadrant of a circular region (lower right corner of the image). The other two regions are separated by a diagonal border from the upper left corner to the lower right corner. The result is summarized in Fig. 9(b). As compared to the ground truth, the segmentation is successful except for the errors located nearby the area where the three regions meet. Although our filtering windows have intrinsic square shapes, they are still able to locate the diagonal and the circular borders satisfactorily.

To demonstrate the performance of our method for input images with more than four regions, an input image similar to the ones in Figs. 6 and 8 but with an additional square region in the center of the image is used as shown in Fig. 9(c). Unlike the networks used for the previous images, $\kappa = 3.0$ and an additional condition is incorporated: An oscillator can respond to an excitation only if it has more than two neighboring oscillators active simultaneously. This condition is to avoid one pixel wide region extensions. As it is depicted in Fig. 9(d), the five regions are successfully segregated along with the background region. There is an interesting "error" related to the boundary between the upper left and the upper right regions in the segmentation result. Even though the boundary should vertically bisect the top edge of the image according to the ground truth, our segmentation result shifts the boundary to the left. In fact, one can observe that the segmentation result is more consistent with that of perception in this particular case.

Finally, an indoor image that was used in Hofmann *et al.* [9] and the resulting segmentation are shown to demonstrate the performance of our method in the presence of both textured and untextured regions (Fig. 10). For this purpose, a filtering window of the size $21 \times 21$ and $\kappa = 2.5$ are used. The condition where an oscillator can respond only if it has more than two simultaneously active neighboring oscillators is also employed to eliminate possible narrow extensions of the regions. Our method is able to segregate the couch with a rich texture, the floor, the couch with approximately uniform luminance, the drawer part of the bookshelf, and the books on top of it in the upper right corner of the image. Fuzzy regions along the borders of the different objects constitute the background. The result is summarized in Fig. 10(b).

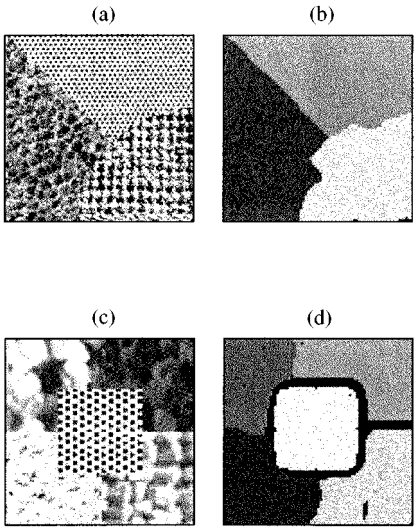


Fig. 9. The performance of our method in the presence of different boundary shape and orientations. (a) An input image of the size $128 \times 128$ with circular and diagonal border orientations. (b) Its segmentation result of the size $80 \times 80$, with $\theta = 3.0$. (c) An input image of the size $128 \times 128$ with five different textured regions. (d) Its segmentation result of the size $80 \times 80$, with $\theta = 3.6$.

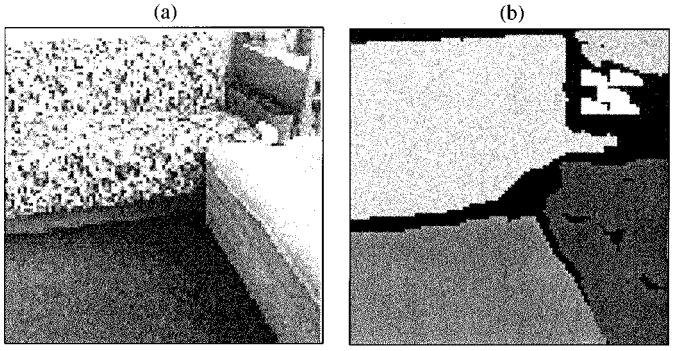


Fig. 10. The performance of our method in the presence of both textured and untextured regions. (a) An indoor image of the size $128 \times 128$ (from [9]). (b) Its segmentation result of the size $90 \times 90$, with $\theta = 3.0$.

## IV. Comparison with Other Methods

### A. Methodology

In their pioneering study, Geman *et al.* [7] incorporated prior knowledge about regions and boundaries into a probabilistic framework, and segmentation is achieved by searching for MAP solutions using simulated annealing. Hofmann *et al.* [9] defined texture segmentation as a pairwise clustering problem based on a similarity measure using Gabor filter outputs and employed a deterministic annealing technique to solve the problem. Jain and Farrokhnia [10] used Gabor filters as texture feature detectors and a variation of the $k$-means algorithm for clustering. However, the selection and/or the integration of filter outputs still remain to be an unsolved problem [18]. Malik *et al.* [13] addressed a similar problem requiring the combination of texture and brightness-defined contours and employed a normalized cut algorithm for grouping.

A core issue in combining multiple features of different kind, as in the case of incorporating a large set of filters as well as spatial proximity, is that different features generally have different ranges of values. Thus, a set of features with comparable ranges is highly desired to eliminate at least one variability in the problem. Unlike most other approaches, our features, $f_j$s, have similar ranges. Using $f_j$s not only facilitates combining features but also strengthens the orientation selectivity in feature detection which is an important property in texture segregation. In fact, our early attempts to use Gabor filters produced unsatisfactory results partly due to large magnitude variations in the filter outputs.

Most methods require prior training either in feature extraction or grouping. GMRF-based approaches, for instance, require prior feature estimation for a selected set of textures and the

tuning of additional parameters. Neural networks, such as the one proposed by Jain and Karu [11], usually involve training either in the filter selection stage or in the grouping process. Our method, on the other hand, does not involve texture dependent training in feature extraction or grouping and thus, lends itself as an unsupervised segmentation system.

Different from other methods, which usually require prior information about the number of labels (regions), LEGION uses a flexible way of expressing labels as phases of oscillators. The interactions between oscillators are in terms of phase only, and operations are formulated in terms of just one phase variable, which implicitly expresses multiple labels. With implicit labeling LEGION does not entail more local operations when more regions need to be segmented. On the other hand, more labels increase local operations in methods employing explicit labeling, such as deterministic and stochastic relaxation methods, because operations are formulated in terms of different labels. Implicit labeling in LEGION comes from the fact that it performs segmentation in *time*.

The present study differs from previous LEGION networks for image segmentation in three main aspects. First, the inclusion of dynamic neighborhood and slow inhibition significantly expands the ability of LEGION for image segmentation in general, and texture segmentation in particular. Second, the present study, for the first time, integrates a large system of differential equations rather than relying on algorithmic abstractions. Third, the extraction of texture features is a significant topic in its own right, which has not been addressed in previous LEGION studies.

### B. Performance

For a more comprehensive evaluation of our method, we first consider results of other methods that are not based on neural networks. For this purpose, we compare our results with those of Geman *et al.* [7], Jain and Farrokhnia [10], and Chellappa *et al.* [3], because of their novel approaches and relatively successful results on a common set of real textures, namely, Brodatz textures [3]. In addition, we include two more recent studies in our comparison, since their images are quite similar to Brodatz textures [11], [9]. Note that these studies include both supervised [7], [4], [11] and unsupervised [10], [9] segmentation methods.

The methods by Geman *et al.* [7] and Hofmann *et al.* [9] generate segmentation results of lower resolution than what we obtain. Because of the large resolution reduction, the two methods would lose boundary accuracy. Recently, Puzicha and Buhmann [17] improved the work of Hofmann *et al.* in terms of computation and resolution. After prior training and elaborate adjustment of several parameters, the methods by Geman *et al.* and Chellappa *et al.* [3] obtain similar results compared to ours. The results of Jain and Farrokhnia [10] would be similar to ours, too, if the correct number of regions is provided to their method and the set of Gabor filters used is able to capture differences in textures forming the test images. In the study by Jain and Karu [11], after successful training on MLP their neural network can generate similar results compared to ours.

Among the five methods selected for comparison, Jain and Farrokhnia [10] and Hofmann *et al.* [9] considered the problem of segmenting images composed of both textured and untextured regions. Jain and Farrokhnia pointed out that their method can segregate textured regions from untextured regions but cannot discriminate one untextured region from another. Hofmann *et al.* assumed a preprocessing step to segregate untextured regions from textured regions in order to process the indoor image in Fig. 10.

In general, we observe that our results are at least as good as those of the above methods. Like ours, they all have classification errors around the boundaries, where decision windows include two or more regions simultaneously during segmentation. The main advantage of the methods based on training using a fixed set of textures is that they can produce good results on images with a large number of different textures. Since our method starts with a weaker assumption about texture types, its performance for such images is generally not as good. On the other hand, our method would perform better for novel texture types. Also, unlike other methods, our method generates a background that is composed of unclassified sites. We claim that the inclusion of a background makes our method more amenable to a real environment, where textured and untextured regions mingle together as do uninteresting regions and interesting ones.

When only neural network methods are considered, our method presents clearly better results. The only other method based on coupled oscillators [1] does not have results on real texture images. We believe that our method has substantially expanded the applicability of neural networks in texture image analysis.

In terms of computational complexity, for a network of the size $80 \times 80$, our network simulation takes typically a minute of CPU time on an HP workstation. This computational performance is comparable with the steepest descent and Monte Carlo algorithms used in those MAP-based approaches that reported computing times [23].

To summarize, after carefully evaluating our method against existing methods in terms of both methodology and performance, we are led to the conclusion that our method offers unique computational advantages, in addition to the fact that our approach has a strong biological link.

## V. Conclusion

Our method is composed of two parts, the extraction of texture features and a neural oscillator network. Both parts are based on the notion of local computations, which is important for parallel and distributed processing. Our method, with only one parameter to adjust and no prior training, is able to obtain comparable segmentation results with other successful studies. Four main factors give rise to the performance of our method. First, a second-order GMRF can describe the dependency of a pixel intensity on that of its neighboring pixels. Second, our texture features seem more discriminatory than GMRF model parameters. Third, our computational framework is a biologically plausible neural network, where a measure of similarity incorporates a dynamic neighborhood. Finally, a large set of differential equations are solved efficiently using the singular limit method.

## ACKNOWLEDGMENT

The authors thank A. K. Jain, S. C. Zhu, and four anonymous referees for their useful comments.

## REFERENCES

[1] P. Baldi and R. Meir, "Computing with arrays of coupled oscillators: An application to preattentive texture discrimination," *Neural Comput.*, vol. 2, pp. 458–471, 1990.

[2] P. Brodatz, *Textures: A Photographic Album for Artists and Designers*. New York: Dover, 1966.

[3] R. Chellappa, R. Kashyap, and B. S. Manjunath, "Model-based texture segmentation and classification," in *Handbook of Pattern Recognition and Computer Vision*, C. H. Cohen, L. F. Pau, and P. S. P. Wang, Eds. Singapore: World Scientific, 1993, pp. 277–310.

[4] G. R. Cross and A. K. Jain, "Markov random field texture models," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-5, pp. 25–39, 1983.

[5] Eckhorn *et al.*, "Coherent oscillations: A mechanism of feature linking in the visual cortex?," *Biol. Cybern.*, vol. 60, pp. 121–130, 1988.

[6] K. B. Eom, "Segmentation of monochrome and color textures using moving average modeling approach," *Image Vision Comput.*, vol. 17, pp. 233–244, 1999.

[7] D. Geman, S. Geman, C. Graffigne, and P. Dong, "Boundary detection by constrained optimization," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, pp. 609–628, 1990.

[8] C. M. Gray, P. König, A. K. Engel, and W. Singer, "Oscillatory responses in cat visual cortex exhibit inter-columnar synchronization which reflects global stimulus properties," *Nature*, vol. 338, pp. 334–337, 1989.

[9] T. Hofmann, J. Puzicha, and J. M. Buhmann, "Unsupervised texture segmentation in a deterministic annealing framework," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 20, pp. 803–818, 1998.

[10] A. K. Jain and F. Farrokhnia, "Unsupervised texture segmentation using Gabor filters," *Pattern Recognition*, vol. 24, pp. 1167–1186, 1991.

[11] A. K. Jain and K. Karu, "Learning texture discrimination masks," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 18, pp. 195–205, 1996.

[12] P. S. Linsay and D. L. Wang, "Fast numerical integration of relaxation oscillator networks based on singular limit solutions," *IEEE Trans. Neural Networks*, vol. 9, pp. 523–532, 1998.

[13] J. Malik, S. Belongie, J. Shi, and T. Leung, "Textons, contours and regions: Cue integration in image segmentation," in *Proc. Int. Conf. Comput. Vision*, 1999, pp. 918–925.

[14] C. V. D. Malsburg, "The correlation theory of brain functions," Max-Planck-Inst. Biophys. Chem., Göttingen, Germany, Int. Rep. 81-2, 1981.

[15] P. M. Milner, "A model for visual shape recognition," *Psychol. Rev.*, vol. 81, pp. 521–535, 1974.

[16] "MIT Vision Texture Image Archive,", Cambridge, MA.

[17] J. Puzicha and J. Buhmann, "Multiscale annealing for grouping and unsupervised texture segmentation," *Computer Vision and Image Understanding*, vol. 76, pp. 213–230, 1999.

[18] T. R. Reed and J. M. H. du Buf, "A review of texture segmentation and feature extraction techniques," *CVGIP: Image Understanding*, vol. 57, pp. 359–372, 1993.

[19] D. Terman and D. L. Wang, "Global competition and local cooperation in a network of neural oscillators," *Phys. D*, vol. 81, pp. 148–176, 1995.

[20] F. Tomita and S. Tsuji, "Extraction of multiple regions by smoothing on selected neighborhoods," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-7, pp. 107–109, 1977.

[21] D. L. Wang and D. Terman, "Locally excitatory globally inhibitory oscillator networks," *IEEE Trans. Neural Networks*, vol. 6, pp. 283–286, 1995.

[22] D. L. Wang and D. Terman, "Image segmentation based on oscillatory correlation," *Neural Comp.*, vol. 9, pp. 805–836, 1997.

[23] J.-P. Wang, "Stochastic relaxation on partitions with connected components and its applications to image segmentation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 20, pp. 619–636, 1998.

[24] D. L. Wang and D. Terman, "Image segmentation based on oscillatory correlation," *Neural Comput.*, vol. 9, pp. 1623–1626, 1997.