LETTER Communicated by Scott Kirkpatrick

A Fast and Robust Cluster Update Algorithm for Image Segmentation in Spin-Lattice Models Without Annealing—Visual Latencies Revisited

Ralf Opara
Florentin Wörgötter
Department of Neurophysiology, Ruhr-Universität, 44780 Bochum, Germany

Image segmentation in spin-lattice models relies on the fast and reliable assignment of correct labels to those groups of spins that represent the same object. Commonly used local spin-update algorithms are slow because in each iteration only a single spin is flipped and a careful annealing schedule has to be designed in order to avoid local minima and correctly label larger areas. Updating of complete spin clusters is more efficient, but often clusters that should represent different objects will be conjoined. In this study, we propose a cluster update algorithm that, similar to most local update algorithms, calculates an energy function and determines the probability for flipping a whole cluster of spins by the energy gain calculated for a neighborhood of the regarded cluster. The novel algorithm, called *energy-based cluster update* (ECU algorithm) is compared to its predecessors. A convergence proof is derived, and it is shown that the algorithm outperforms local update algorithms by far in speed and reliability. At the same time it is more robust and noise tolerant than other versions of cluster update algorithms, making annealing completely unnecessary. The reduction in computational effort achieved this way allows us to segment real images in about 1–5 sec on a regular workstation. The ECU-algorithm can recover fine details of the images, and it is to a large degree robust with respect to luminance-gradients across objects. In a final step, we introduce luminance dependent visual latencies (Opara & Wörgötter, 1996; Wörgötter, Opara, Funke, & Eysel, 1996) into the spin-lattice model. This step guarantees that only spins representing pixels with similar luminance become activated at the same time. The energy function is then computed only for the interaction of the regarded cluster with the currently active spins. This latency mechanism improves the quality of the image segmentation by another 40%. The results shown are based on the evaluation of gray-level differences. It is important to realize that all algorithmic components can be transferred easily to arbitrary image features, like disparity, texture, and motion.

1 Introduction

Solutions for computer vision problems almost always avoid the use of spiking neural networks because they are computationally expensive. This, on the other hand, makes it usually rather difficult, if not impossible, to implement neuronal algorithms directly. For example, image segmentation by spiking neural networks is commonly achieved by synchronizing the firing patterns of units that represent a common object (von der Malsburg 1981; von der Malsburg & Schneider 1986; Gray, König, Engel, & Singer, 1989; Ernst, Pawelzik, & Geisel, 1994; Nischwitz & Glünder, 1995). The same concept (i.e., "synchronization") can be introduced into spin-lattice models by mapping every neuron of the neural net onto one spin element in the lattice (Geman, Geman, Graffigne, & Dong, 1990; Vorbrüggen, 1995; Eckes & Vorbrüggen, 1996; Blatt, Wiseman, & Domany, 1996). Synchronization then means that different spins will get the same orientation. In principle such a mechanism can be used to label the different objects in a visual scene by trying to ensure that spins that belong to the same object have the same orientation (i.e., the same label).

Several spin-lattice algorithms exist that can be used to segment visual scenes by labeling the objects (Geman et al., 1990; Vorbrüggen, 1995; Eckes & Vorbrüggen, 1996; Blatt et al., 1996). These algorithms differ mainly in the way in which they define the interaction range between spins and how the individual spins are iteratively updated. Local update algorithms (Geman et al., 1990; Vorbrüggen, 1995; Eckes & Vorbrüggen, 1996) modify only one spin per iteration, and the interaction ranges are usually small. With cluster update algorithms (Blatt et al., 1996), on the other hand, larger interaction ranges are introduced due to the treatment of whole clusters and groups of spins—the spin clusters—are updated simultaneously. Like neural nets, spin-lattice models are also confronted with the problem of how to organize spins (synchronize units) over large areas. Local update algorithms can solve this problem in many cases by propagating a certain modification through the whole lattice step by step, which makes them rather slow. Cluster update algorithms are much faster because a change can affect many spins at the same time. This, however, affects their robustness in a negative way because clusters that should get different labels can easily collapse and form one indistinguishable clump.

In this article, we mainly focus on cluster update algorithms and describe a modification of the Potts model (Potts, 1952), which preserves the advantages of the original version—the convergence speed—but is much more robust. We also show that this algorithm can be used to introduce one additional neuronal concept—visual latencies—to improve its performance (Opara & Wörgötter, 1996; Wörgötter et al., 1996).

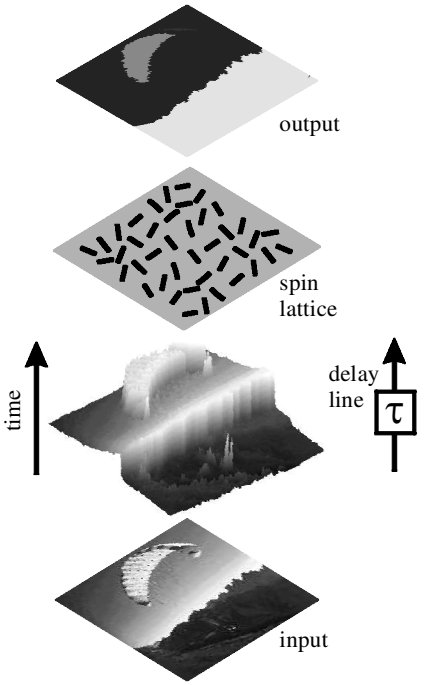

Figure 1: Schematic diagram of the system. The input image is piped into a luminance-dependent delay line. Bright objects are processed before dark objects. The temporal structure of this data stream determines the interaction between spins within the second part of the system, which consists of a spin lattice.

2 Overview of the Model

The model we propose consists of two parts, shown in Figure 1. The first part of the system contains a luminance-dependent delay line ("visual latencies"; see also Opara & Wörgötter, 1996). Pixels with a high luminance are processed before dark pixels. The temporal structure of this data stream determines the interaction between spins in the second part of the system. The second part consists of Potts spins (Potts, 1952), which are arranged on a two-dimensional lattice. Within this spin lattice, the algorithm tends to assign the same label only to those spatially adjacent pixels that reach the spin lattice with high temporal coherence (same latency). Pixels with strong delays between them will get different labels.

2.1 Spin-Lattice Model. First we will describe the model *without* visual latencies. The spin-lattice model we use is similar to the model that Potts proposed in 1952. In the Potts model, each spin can take q different values

($2 \leq q$), which is a generalization of the Ising model ($q = 2$) (Ising, 1925). In the two-dimensional case, the spins are arranged on a lattice of size $N = L_x L_y$.

We define a label using the symbol w and a label configuration by $W = \{w_1, \ldots, w_N\} \in \Omega$, where Ω is the space of all configurations. A subconfiguration, where we consider fewer than N spins, is denoted by a superscript (e.g., W^c).

The global energy function of our specific Potts configuration $W \in \Omega$ is given by:

$$E(W) = \sum_{i=1}^{N} \left[K_i + \sum_{\langle i,j \rangle} -J_{i,j}\delta_{w_i,w_j} \right] = \sum_{i=1}^{N} \left[K_i + \sum_{<i,j>} E_{ij} \right], \tag{2.1}$$

where w_i,w_j is labels of spin i and j; $J_{i,j}$ is the interaction strength between spins at locations i and j; δ_{w_i,w_j} is a kronecker function, being 1 if $w_i = w_j$ and 0 else; $\langle i, j \rangle$ is a neighborhood of spin i with $||i, j|| \leq \Delta$, where Δ is a constant that needs to be set; and K_i is a global "inhibition."

If the term K_i (which will be defined and discussed below) were set to zero, one would get the global energy function of the generic Potts model in its usual form. In the homogeneous Potts model ($J_{ij} = const$), all spins are interacting with the same strength. In the inhomogeneous Potts model, the interaction strength is changing over space ($J_{i,j} \neq const$).

To apply the Potts model to an image segmentation task, the similarity of the input image has to be represented in the interaction strength J_{ij} of the spins (Geman et al., 1990; Vorbrüggen, 1995; Eckes & Vorbrüggen, 1996; Blatt et al., 1996). Spins that are representing similar image parts (same object) have to interact strongly, while nonsimilar image parts will not interact or will interact with a negative strength.

2.2 The Energy-Based Cluster Update (ECU) Algorithm. Several different algorithms exist in the literature to order the spins in a Pott-model according to a predefined goal, like determining phase transitions in ferromagnetic systems or, as in our case, in order to segment an image. The local-update "Metropolis" algorithm (Metropolis, Rosenbluth, Rosenbluth, Teller, & Teller, 1953), which rotates single spins per iteration and tries to minimize a global energy function using simulated annealing (Kirkpatrick, Gelatt, & Vecci, 1983), is well known. Cluster update algorithms (Swendsen & Wang, 1987; Wolff, 1989) treat groups of spins simultaneously and are therefore much faster, but often an undesired fusion of clusters is observed.

The approach we propose seeks to combine the advantages of both types of algorithms: local and cluster update. We will demonstrate that our algorithm overcomes the critical slowing down at phase transitions and in the low-temperature phase observed in local update algorithms, so simulated

annealing is not necessary for our approach. Moreover, it does not produce unwanted cluster fusions.

The structure of the ECU algorithm is as follows. We start with a random label configuration, and individual spins are combined as clusters according to equation 2.2. Then we calculate an energy function. Following this, the label configuration is modified such that the probability of a new configuration depends on the energy of this new configuration and the temperature. From this new label configuration, a new cluster configuration is formed, and the next iteration starts.

First we show how clusters are formed. Similar to the definitions for labels above, we define c as a cluster; $C = \{c_1, \ldots, c_M\}$, $M \leq N$ as a configuration of clusters; and $C \in \Gamma$, where Γ is the space of all possible cluster configurations.

The formation of clusters is the same as in other cluster update algorithms. Clusters must contain only spins that are in the same state (same label). The binding probability P_B that two nearest-neighbor spins are conjoined in order to form a cluster depends on the temperature T and the coupling strength $J_{i,j}$ (see below) of the two nearest-neighbor spins and is given by:

$$P_{B,\langle ij\rangle_c}(i, j) = (1 - \exp[-0.5 J_{i,j}\delta_{w_i,w_j}/T]) = (1 - \exp[0.5 E_{ij}/T]). \qquad (2.2)$$

The factor 0.5 is a normalization constant that is necessary because the clusters are not flipped independently of each other, as in the Swendsen and Wang algorithm. We show later that this normalization ensures that in the thermodynamic equilibrium, configurations are taken according to the Boltzmann distribution. Furthermore, only a certain subset of spins denoted as $\langle ij\rangle_c$ are taken into account when computing P_B—namely, those that are nearest neighbors and have a positive $J_{i,j}$, because otherwise negative probabilities would obtain.

At high temperatures, the average cluster size is small ($J_{i,j}/T \to 0$ and $P_B(i, j) \to 0$), while at lower temperatures the possible cluster size increases.

In our approach we consider the energy gain of a cluster flip. Therefore, we define the energy of the subconfiguration W_k^c, assuming that all spins in the cluster c would take the label w_k. This is done by considering the interactions of all spins in the cluster with those outside the cluster but within a neighborhood $\langle ij\rangle$, $c_k \neq c_j$,

$$E(W_k^c) = \sum_{i \in c_i} \left[K_i + \sum_{\substack{\langle ij\rangle \\ c_k \neq c_j}} \eta_{ij} E_{ij} \right] \qquad (2.3)$$

where $\langle ij\rangle$, $c_k \neq c_j$ is the noncluster neighborhood of spin i, which is that set of spins j outside the cluster c_k but within the interaction range Δ of

spin i given by $||i, j|| \leq \Delta$ The constant η_{ij} can be set to 1.0 for all practical purposes. In order to arrive at the correct proof of the detailed equilibrium, however, a more complicated definition is required.[1]

In classical cluster update algorithms, each cluster is simultaneously and independently assigned a spin label. In the ECU algorithm, we instead update one cluster at a time given the label of neighboring clusters. This uses the relative energies of different label assignments computed by equation 2.3.

K_i acts as a global inhibition and is commonly used in other neural networks. Its value is rather noncritical, and it can be set within wide ranges. It ensures that far-away objects will get different labels. We define it as:

$$K_i = \frac{\alpha}{N} \sum_{j=1}^{N} \delta_{w_i, w_j}, \tag{2.4}$$

where α is a control parameter that adjusts the strength of the global inhibition ($\alpha \geq 0$).

$J_{i,j}$, which occurs in the definition of E_{ij} (see, e.g., equation 2.1), is the interaction strength of two spins i and j. We define it such that regions with similar gray values will get positive weights, whereas dissimilar regions get negative weights (Vorbrüggen, 1995; Eckes & Vorbrüggen, 1996):

$$J_{i,j} = 1 - \frac{|g_i - g_j|}{\Theta}, \tag{2.5}$$

where g_i is the gray value of pixel i of the input image and Θ is the average difference of the gray values within all interaction neighborhoods N_i. It thereby represents the intrinsic (short-range) similarity within the whole input image. It is given by:[2]

$$\Theta = \frac{1}{N} \frac{1}{(2\Delta + 1)^2 - 1} \sum_{i=1}^{N} \sum_{\langle ij \rangle} |g_i - g_j|, \tag{2.6}$$

where $(2\Delta + 1)^2 - 1$ is the number of neighbors of a spin.

[1] The correct definition of η is given by redefining equation 2.3:

$$E(W_k^c) = \sum_{i \in c_k} \left[K_i + \sum_{\substack{\langle ij \rangle c_i \\ c_k \neq c_j}} 0.5 E_{ij} + \sum_{\substack{\langle ij \rangle \setminus \langle ij \rangle c_i \\ c_k \neq c_j}} E_{ij} \right],$$

where $\langle ij \rangle_{c_k}$ is that set of spins used to compute P_B and $\langle ij \rangle \setminus \langle ij \rangle_{c_k}$ is its opposite.

[2] In the case of $\Theta = 0$, equation 2.5 is ill defined, but in this case only a single uniform surface exists, and segmentation is not necessary.

Similar to a Gibbs sampler, the selection probability P_S of selecting a label subconfiguration W_k^c where all spins in the cluster c take the label w_k is given by:

$$P_S(W_k^c) = \frac{\exp(E(W_k^c)/T)}{\sum_{j=1}^{q} \exp(E(W_j^c)/T)}. \tag{2.7}$$

The number of possible labels that a cluster can take is given by q. In the following simulations, we use $q = 10$ unless otherwise noted.

3 Detailed Balance

We have two sets of variables: the label configuration $W \in \Omega$ and (similar to the Swendsen and Wang algorithm) the cluster configuration $C \in \Gamma$. The complete system assumes configurations in the shared configuration space $\Gamma \times \Omega$.

The goal of the ECU algorithm is to label an image according to the energy function on the labels $E(W)$ (see equation 2.1), which leads to an equilibrium probability distribution[3] $P(W) = \frac{1}{Z}\exp(-E(W)/T)$. Labeling could be done simply by Gibbs sampling, for example, but Gibbs sampling individual spins can be very slow. To speed up sampling, we define an energy function over additional variables, the clusters c, such that the equilibrium distribution $P(W, C) = \frac{1}{Z}\exp(-E(W, C)/T)$ still has the same marginal distribution, $\sum_C P(W, C) = P(W)$, as defined above. Then we define a Markov process over this joint system consisting of two steps: (1) sampling of clusters given spins $P(W, C \to W, C')$ and (2) sampling of spins given clusters $P(W, C \to W', C)$. The claim to prove consists of two aspects. If detailed balance holds, applying these two steps in succession should (1) result in the desired equilibrium distribution $P(W, C)$, which has the desired marginal distribution over spins $P(W)$ and (2) this needs to be the Boltzmann distribution (Swendsen & Wang, 1987; Binder & Heermann, 1988; Neal, 1993). In the detailed balance, the probability $P(W)$ of a configuration W, multiplied with the probability $P(W \to W')$ for the transition into the configuration W', is identical to the probability of the reverse process. Hence:

$$P(W)P(W \to W') = P(W')P(W' \to W), \tag{3.1}$$

where $P(W)$ is the probability for a configuration (W) and $P(W \to W')$ is the probability for the transition between (W) and (W').

The forming of clusters and the assigning of labels are independent events. Therefore, we get the transition probabilities by summing over all

[3] Z is the partition function.

possible cluster configurations that allow a transition from spin configuration W to W' and vice versa:

$$P(W \to W') = \sum_C P(W \to C)P(C \to W') \tag{3.2}$$

$$P(W' \to W) = \sum_C P(W' \to C)P(C \to W). \tag{3.3}$$

Dividing both equations, we have:

$$\frac{P(W \to W')}{P(W' \to W)} = \frac{\sum_C P(W \to C)P(C \to W')}{\sum_C P(W' \to C)P(C \to W)} = \exp\left(\frac{E(W) - E(W')}{T}\right). \tag{3.4}$$

The right-most equality in equation 3.4 is the condition of detailed balance, which we have to show. This is equivalent to showing that the following equation holds:

$$\frac{P(W \to W')}{P(W' \to W)} = \frac{P(W \to C)P(C \to W')}{P(W' \to C)P(C \to W)} = \exp\left(\frac{E(W) - E(W')}{T}\right). \tag{3.5}$$

The equivalence of equations 3.4 and 3.5 is spelled out in equations 3.6 and 3.7:

$$P(W \to C)P(C \to W') = \exp\left(\frac{E(W) - E(W')}{T}\right) \times P(W' \to C)P(C \to W) \tag{3.6}$$

$$\begin{aligned} &\sum_C [P(W \to C)P(C \to W')] \\ &= \sum_C [\exp\left(\frac{E(W) - E(W')}{T}\right) P(W' \to C)P(C \to W)] \\ &= \exp\left(\frac{E(W) - E(W')}{T}\right) \sum_C [P(W' \to C)P(C \to W)]. \end{aligned} \tag{3.7}$$

Thus, equation 3.5 needs to be proved.

In the following equations, only the contributions that are different for the configurations W and W' are calculated, because all others cancel in equation 3.5.

$$P(W \to C) \sim \prod_{\substack{\langle ij \rangle_C \\ w_i = w_j \\ c_i \neq c_j}} \exp(0.5E_{ij}/T), \tag{3.8}$$

where $\langle ij \rangle_c$ are the neighboring spins that are taken into account by forming a cluster (see the remarks concerning equation 2.2). The probability of moving from a cluster configuration C to a label configuration W' is given by:

$$P(C \to W') \sim \prod_{\substack{\langle ij \rangle_c \\ c_i \neq c_j \\ w'_i = w'_j}} \exp[(-0.5E_{ij})/T] \prod_{\substack{\langle ij \rangle \setminus \langle ij \rangle \\ c_i \neq c_j \\ w'_i = w'_j}} \exp[(-E_{ij})/T] \tag{3.9}$$

where $\langle ij \rangle \setminus \langle ij \rangle$ are the neighboring spins that are not taken into account by forming a cluster. Now equation 3.5 can be evaluated by constructing the other term similar to equations 3.8 and 3.9:

$$\begin{aligned} \frac{P(W \to C)P(C \to W')}{P(W' \to C)P(C \to W)} &= \frac{\prod_{\substack{\langle ij \rangle \\ c_i \neq c_j \\ w'_i = w'_j}} \exp(-E_{ij}/T)}{\prod_{\substack{\langle ij \rangle \\ c_i \neq c_j \\ w_i = w_j}} \exp(-E_{ij}/T)} \\ &= \exp\left(\frac{E(W) - E(W')}{T}\right), \end{aligned} \tag{3.10}$$

which proves the claim.

4 Characterization of the ECU Algorithm

In order to quantify the ECU algorithm, we use the magnetization m and the magnetic susceptibility χ of the system (see Chen, Ferrenberg, & Landau, 1992):

$$m(W) = \frac{qN_{\max}(W) - N}{(q-1)N}, \qquad \chi = \frac{N}{T}(\langle m^2 \rangle - \langle m \rangle^2), \tag{4.1}$$

where $N_{\max}$ is the number of spins whose label w occurs most frequently.

The magnetic susceptibility χ can be used to localize the different phases of the system where large fluctuations of the magnetization occur. Other interesting quantities are the spin-spin correlation $\langle \delta_{w_i, w_j} \rangle$ and the number of clusters $\langle Q \rangle$. The spin-spin correlation indicates how many adjacent spins are in the same state.

To show the basic properties of the ECU algorithm, a very simple image was presented to the system consisting of two rectangles having different averaged gray values and different standard deviations of the noise (see Figure 2A). In the following we will demonstrate the behavior of the system for changing temperatures. (This is done merely to characterize the system, because annealing is not necessary when segmenting an image with the ECU algorithm.) Figure 2B shows the averaged magnetization at different

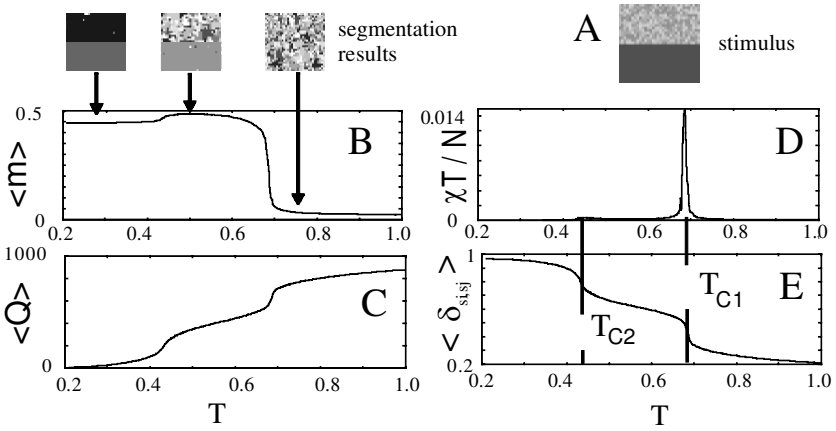

Figure 2: The behavior of the system at different temperatures. As the system parameter, we use $N = 32 \times 32$, $\Delta = 1$, $\alpha = 0$, and $q = 10$. In this simulation, we use cyclic boundary conditions. (A) The stimulus consists of an image containing two rectangles of different averaged gray values ($\overline{g}_{bottom} = 60$, $\overline{g}_{top} = 120$) and different standard deviations ($\sigma_{bottom} = 0$, $\sigma_{top} = 25$) of the noise. In this particular simulation, Θ (see equation 2.6) was not computed but set to 25. As a function of the temperature we plot, (B) the average magnetization and examples of some segmentation results, (C) the average number of clusters, (D) the magnetic susceptibility, and (E) the average spin-spin correlation.

temperatures. At high temperatures ($T \to \infty$), all spins are totally disordered ($lim_{T\to\infty}\{< m >\} = 0$). As the system reaches temperature T_{c_1}, the spins, representing the lower rectangle, show a phase transition, indicated by an abrupt increased averaged magnetization, by large fluctuations of the magnetization (χ; see Figure 2D), and by an abrupt decrease in the number of clusters ($\langle Q \rangle$; see Figure 2C).

As the temperature is lowered, the magnetization and the spin-spin correlation are slowly increased until a second-phase transition occurs at T_{c_2}. At this temperature, the spins representing the upper rectangle are getting ordered, again indicated by abrupt changes of the variables describing the system. In the low-temperature phase ($T \leq 0.4$), the spins representing the upper and lower rectangle have different labels.

5 Comparing Different Algorithms

A comparison of the three update algorithms—local (Gibbs sampler), cluster (Blatt et al.), and the ECU algorithm—is shown in Figure 3. The input for all simulations consists of two rectangles and a thin line surrounding and separating them (see Figure 3A). In all simulations, the parameters are the

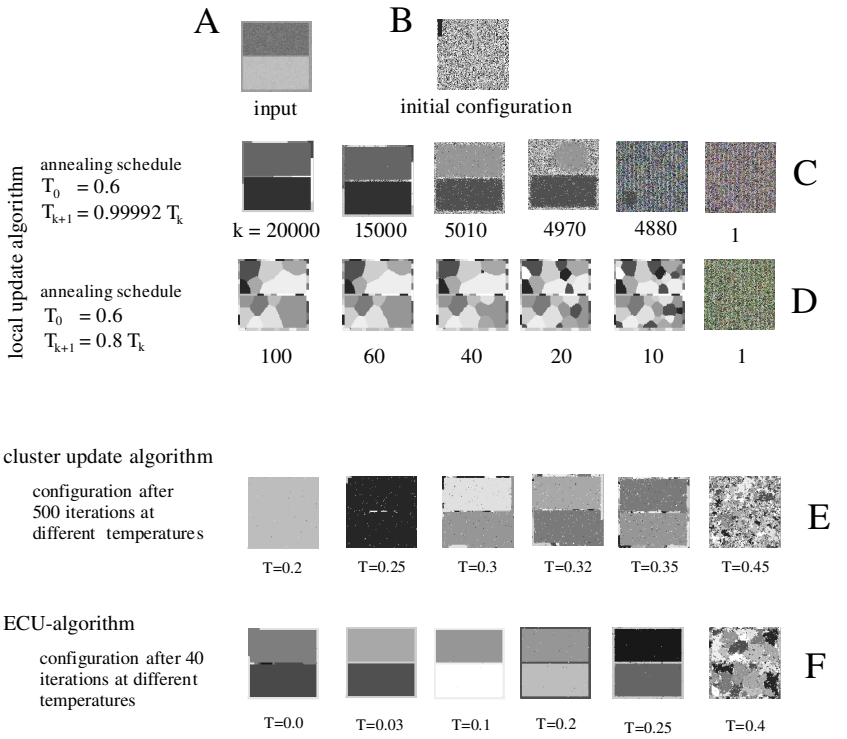

Figure 3: Comparison of different update rules. As the parameter, we use $N = 128 \times 128$, $\alpha = 0$, and $q = 10$. (A) The input image for all simulations consists of two rectangles and a background. Most pixels of the background are covered by the two rectangles, and only thin lines of the background are visible. (B) The initial random configuration. (C, D) Configurations of a local update algorithm (Gibbs sampler) at different iterations ($\Delta = 5$). (E) Configurations of the cluster update algorithm (Blatt et al., 1996) ($\Delta = 1$). (F) Configurations of the ECU algorithm ($\Delta = 1$).

same, with the exception of the temperature, which is varied.

The simulations start with a random configuration, shown in Figure 3B. Figures 3C and 3D shows the spin-lattice configuration of a local update algorithm at different iterations. The initial temperature T_0 was set to 0.6. At every iteration, the temperature is cooled according to $T_{k+1} = 0.99992 T_k$ (see Figure 3C). At iteration $k \approx 5000$, the rectangles start to move to an ordered configuration, and 50 iterations later, both rectangles are ordered. At iteration $k \approx 14{,}000$ the background starts to organize. Although we used a very slow annealing schedule, the background is divided into several segments ($k = 20{,}000$).

If one uses faster annealing schedules, which is desirable if confronted with any close-to real-time problem (e.g., robot vision), the rectangles are also divided into several segments (see Figure 3D). Cluster update algorithms label the input image correctly only within a small range of temperatures. The task of image segmentation is to find this range.

Figure 3E shows the configuration of the spin lattice after 500 iterations of the cluster update algorithm for several runs at different temperatures. For every temperature, the system starts with a random configuration, shown in Figure 3B. At high temperatures (e.g., $T \approx 0.45$), the spin lattice is disordered. At temperatures between $T = 0.32$, and $T = 0.35$, the input image is more or less correctly labeled by the spin configuration. At $T = 0.32$, parts of the background and the lower rectangle are bound together. If the temperature is decreased to $T = 0.25$, all three objects are bound together.

Figure 3F shows the spin lattices configuration of the ECU algorithm after 40 iterations. The same parameters and the same connection strengths between spins are used as in the simulations (Figures 3C–E). If the temperature T is lower than 0.25, the input image is always segmented correctly after 40 iterations. As the temperature increases, the system is increasingly disordered. However, if the temperature were set to zero, randomly occurring incorrect label associations (due to the initialization process) would be frozen, and an incorrect segmentation might occur (see Figure 3E, left). Thus, in general, zero temperature is to be avoided.

6 Introducing Visual Latencies

So far we have described the basic properties of the spin model and the chosen dynamics of the ECU algorithm. Now we introduce visual latencies that naturally occur in the visual system of the higher vertebrates. The first part of the system (see Figure 1) contains a luminance-dependent delay line. Pixels with a high luminance pass the delay line faster than low-luminous pixels. The time to pass the delay line is defined as

$$t_{lat}(i) = (g_{\max} - g_i) f_{lat}, \tag{6.1}$$

where f_{lat} is a factor that determines the maximal latency, $g_{\max}$ is the maximal gray value, and g_i is the gray value of pixel i. The latency differences of two pixels i and j are therefore $\Delta t_{lat} = t_{lat}(i) - t_{lat}(j) = (g_j - g_i) f_{lat}$.

Due to the temporal structure, which now defines the similarities of the input image, the interaction strength of the spins (see equation 2.5) is now redefined in the following way:

$$\begin{aligned} J_{i,j} &= 1 - \frac{|t_{lat}(i) - t_{lat}(j)|}{f_{lat}\Theta}; \text{ if } t > t_{lat}(i) \wedge t > t_{lat}(j) \\ J_{i,j} &= 0; \text{ else.} \end{aligned} \tag{6.2}$$

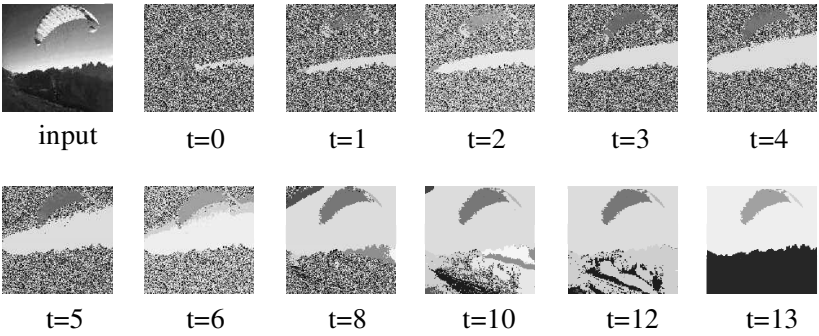

Figure 4: The stimulus consists of an $N = 128 \times 128$ image (upper left), containing a paraglider, a shaded sky, and some hills. The other panels show the label distribution of the system at different iteration steps. The labels are coded as gray values. Spins with the same label (same gray value) belong to one object; different gray levels indicates the assignment to different objects. As parameters we used $T = 0.008$, $f_{lat} = 0.04$, $\Delta = 4$, $\alpha = 1$, and $q = 10$.

Note that equations 2.5 and 6.2 are very similar; the gray-value differences in equation 2.5 are recoded only as latency differences in equation 6.2. The latency $t_{lat}(i)$ defines the time when the spin i representing pixel i is activated the first time. This activation has the consequence that spin i can interact with other spins that are already activated. Therefore, the temporal structure of the input data stream also influences the temporal development of ordered structures in the spin lattice.

7 Segmentation of Real Images

Figure 4 shows snapshots of the label distributions of the model if a stimulus (upper left) is given to the system. In general, a few parameters need to be set in order to make the algorithm work. The following can be used as an adjustment guideline. The temperature is rather uncritical; it should be low but must not be equal to zero (use, e.g., $T = 0.01$). Furthermore, set $f_{lat} \in [0.00, 0.08]$. If the image contains very little noise, one should let $f_{lat} \to 0$. The same holds for Δ Set it to 2.0 for noise-free images and to about 5.0 if noise exists. Setting α to 1.0 will almost always work. The number of labels q is determined by the number of objects that are to be expected in the visual scene. Most often $q \approx 10$ will suffice.

The simulation starts with a random label configuration, and each label is represented by nearly the same number of spins. During the first iterations, only the brightest objects (short latency) are processed (the paraglider and the lower part of the sky). According to the competition included in the dynamics of the system, the spins of the paraglider will receive the same

label, while the spins representing the sky will get a different label. Due to the latency, processing of the parts with a lower luminance does not start before iteration 9. At iteration 13, the whole image is segmented into four different areas.[4] The efficiency of the ECU algorithm can be judged from iterations 12 and 13, where a surface of nearly $40 * 128$ pixels (hills) is flipped during only one iteration. In comparison, local update algorithms need numerous updates and a careful annealing schedule to achieve this.

Figure 5 shows the configuration of the spin lattice at different iterations for two more examples of real images. The input consists of a moose on a meadow (see Figure 5A) and a canvas painting of a woman lying on a sofa (see Figure 5B). Again the brightest areas are processed first: the sky and the woman. The darkest objects, the moose and the sofa, are processed later, at iterations 32 (Figure 5A) and 34 (Figure 5B), respectively. In Figure 5A, five objects are detected: the sky, the meadow, two different hills, and the moose. In Figure 5B, six objects are detected: the wall, the carpet, the woman, and three segments representing the sofa. The sofa is divided into three segments because they are disconnected over distances much larger than the interaction range of the spins.

8 Performance Quantification of the ECU Algorithm

The performance of the model is determined in a series of simulations in which the basic parameters of the system temperature, latency, and the extent of the interaction neighborhood are varied. The stimulus given to the system consists of two rectangles with average gray values of 100 and 120. The task for the system was to segment input images that contain different amounts of noise, like those shown on top of Figure 6A.

An image is segmented correctly if the spins representing rectangle 1 are all in the same state, and the spins representing rectangle 2 are in a different state. Therefore the performance is measured by a quantity denoted as the relative amount of misclassified pixels (RAMP).

In Figure 6A RAMP is measured as a function of the latency factor f_{lat} (see equation 6.1) and noise. The latency factor f_{lat} was varied from $f_{lat} = 0\ iteration/\Delta g$ up to $f_{lat} = 0.4\ iteration/\Delta g$. If the gray-value difference Δg of two pixels is, for example, 20, then the latency difference between these pixels is varied between 0 and $0.4 * 20 = 8$ iterations. The signal-to-noise ratio (snr) was varied between ∞ and 1. The temperature was set to 0.05. For each data point, 100 simulations were used; afterward, the data points were smoothed using a weighted binominal averaging procedure.

At high snr, the image was always segmented correctly by the ECU algorithm, which is not necessarily the case for local update algorithms at such low temperatures. If the $\sqrt{snr}$ reaches 2.5, a few pixels are misclassified,

[4] There is a small strip with a different label to the right of the paraglider.

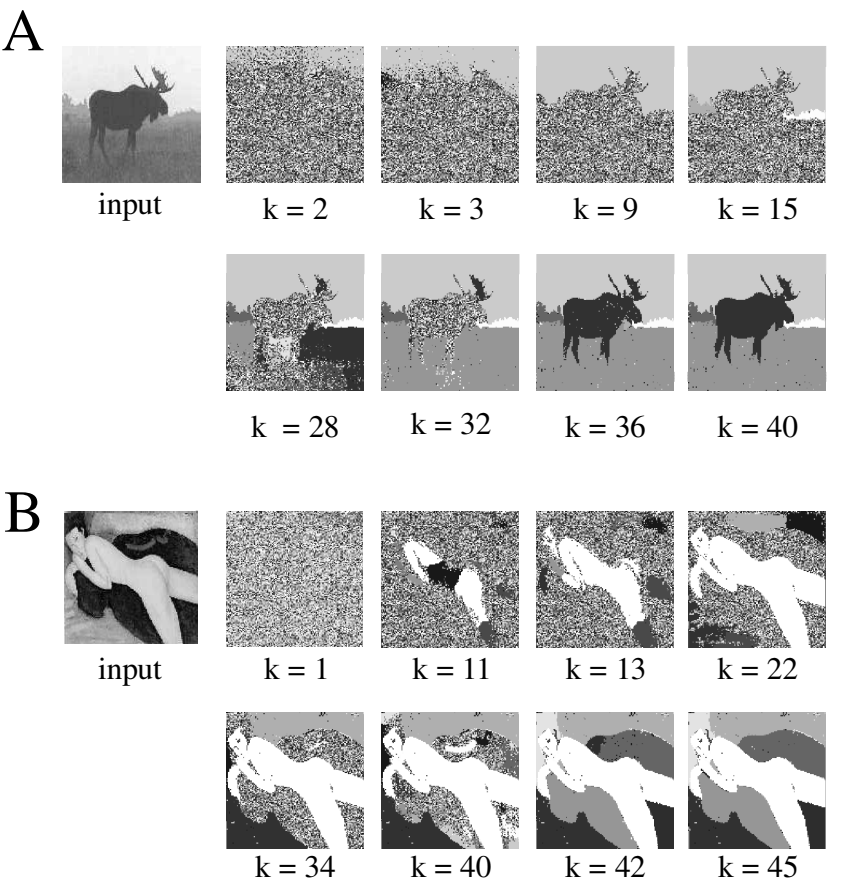

Figure 5: (A) The stimulus consists of an $N = 128 \times 128$ image, containing a moose. The panels show the label distribution of the system at different iteration steps. As parameters, we used $T = 0.008$, $f_{lat} = 0.2$, $\Delta = 5$, $\alpha = 1$, and $q = 10$. (B) Same as (A), but the input image consists of a painting of a woman and a sofa. Parameters are the same as in (A).

and the RAMP is greater than zero. As the noise increases, the number of misclassified pixels also increases, due to fact that the similarities between the pixels are not calculated correctly. The latency introduced in our system has a positive effect on the segmentation quality, improving it by about 50%.

Figure 6B shows the influence of the latency differences on the convergence speed of the system. For this test also, a rather simple image is used (see the inset), containing four square objects. The squares have different gray values, which lead to different total latencies t_{lat}. The brightest object is always processed at iteration zero. The second object is processed at it-

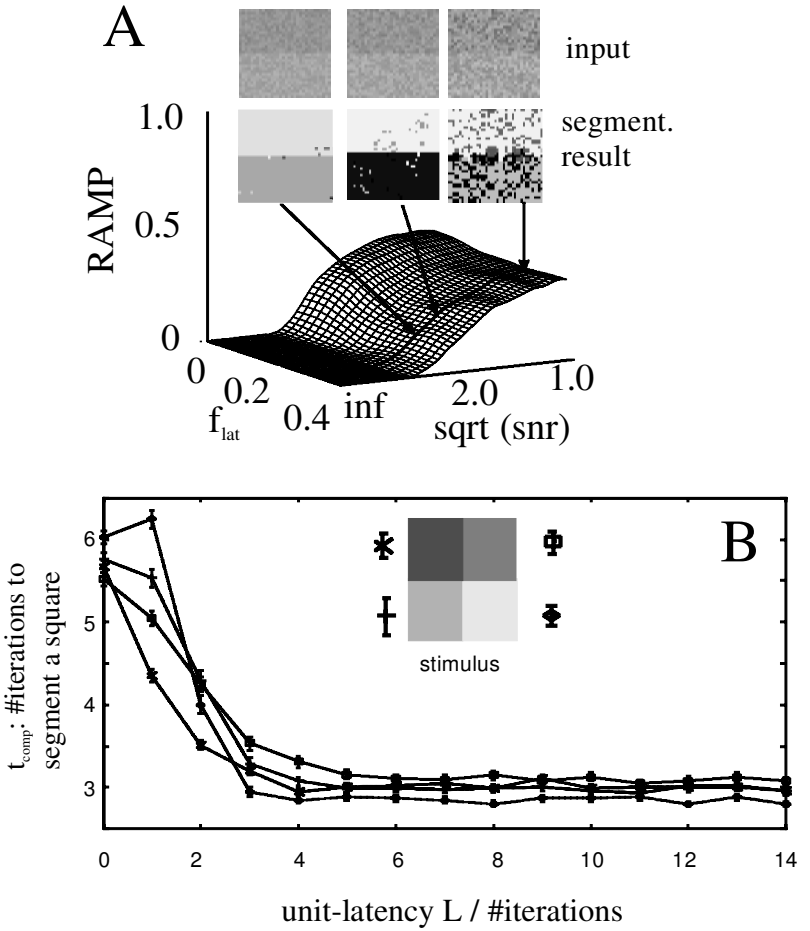

Figure 6: The input image ($N = 32 \times 32$ pixel) consists of two rectangles with averaged gray values of 100 and 120 and different noise ($\sqrt{snr}\ \infty$ down to 1). (A) RAMP as a function of latency and noise. We set $T = 0.05$, $\Delta = 5$, $\alpha = 0.5$, and $q = 10$. (B) Averaged number of iterations to segment a certain square as a function of the unit latency L. The input image ($N = 32 \times 32$ pixel) consists of four squares with different gray levels (inset). We set $\Delta = 3$, $snr = \infty$, $\alpha = 1$, and $q = 4$.

eration $t_{lat} = 1 \times L$, the third object at iteration $t_{lat} = 2 \times L$, and so on. The unit latency L is varied between 0 and 14 iterations (abscissa). With a unit latency of $L = 0$, all objects are processed simultaneously.

In 200 simulations using different initial random configurations, the average computational time (t_{comp}, in number of iterations) is determined to

segment a certain square and plotted against the unit latency L. We define $t_{comp} = t_{seg} - t_{lat}$, where t_{seg} is the actual iteration time reached when a given square is completely segmented and t_{lat} the total latency for this square. We use this particular measure because at iterations $t < t_{lat}$, nearly no computer time is allocated for the processing of that particular square.

In Figure 6B one can see that at $L = 0$, the averaged number of iterations to segment a square is nearly the same for all four objects ($t_{comp} \approx 5.7$). With increasing latency, the number of iterations necessary for the segmentation is decreased for all squares, until a plateau is reached ($t_{comp} \approx 3.1$). The number of iterations until a square is segmented is reduced by nearly 50% as compared to $L = 0$.

9 Discussion

9.1 Comparing the Algorithms. In this study we compared different versions of update algorithms for image segmentation in spin-lattice models and tried to find a solution to several of the most common problems associated with them. Figure 3 is exemplary for these problems, which unfortunately are very generic such that it is much easier to maladjust the update algorithm rather than to find an acceptable parameter set. In particular, it is almost impossible to find such a set for nasty scenes like the one in Figure 3, which looks so simple. Even with 20,000 iterations in the annealing schedule, the local update algorithm still fails to produce the correct result, because the extent of the local similarities in the border of the image is very small. A much slower annealing would ultimately achieve a correct segmentation, but 20,000 iterations took more than 10 hours on our workstation. Faster annealing leads to a failure in even the large rectangular areas of the image, resulting in a patchwork structure. Commonly used versions of cluster update algorithms (Blatt et al., 1996) in principle could lead to a fast and correct labeling of this image, but the range of temperatures for which this will be achieved is in many cases rather small. Figure 3 demonstrates that it is smaller than 0.03 (= 0.32–0.35) for this particular picture, and the results after 500 iterations are unsatisfactory. In fact, after several hours of trying to produce better results with this algorithm by adjusting the temperature in ever finer steps, we gave up.

The novel ECU-algorithm that we designed, on the other hand, produced exact results within only 40 iterations at low temperatures. Thus, our algorithm makes annealing unnecessary (like all cluster update algorithms). In addition, it is robust with respect to the chosen temperature. This usually allows setting the temperature within wide ranges.

Real images were segmented with the same speed and accuracy. Only for the paraglider image did we spend some time trying to find the optimal parameter set, which finally reduced the number of iterations to 13. For the moose and the painting of the lying women, we set the parame-

ters in a single shot according to our previous experiences. Thus, in these cases, we needed some more iterations to reach the final result; nevertheless, segmentation took less than 5 seconds without having particularly optimized the program code on our SUN SPARC 20 with respect to speed. In addition, our algorithm is able to recover even fine details (like the antlers of the moose) and structures that fade into the background (like its legs) will be labeled correctly. Furthermore, the image of the paraglider shows that even rather strong luminance gradients (sky) will be tolerated and correctly treated. The painting demonstrates that complex bent shapes, like the body of the women with its arms in different positions, are also correctly recognized as long as they are connected. In general, we found that segmentation failed to match our own expectations only in those cases where contextual knowledge is necessary to bind objects correctly.

9.2 Visual Latencies Revisited. Intriguingly, the initial starting point for this study was the realization that our own older studies (Opara & Wörgötter, 1996; Wörgötter et al., 1996) needed to be pursued in a different algorithmical context in order to better advertise the idea of using visual latencies in image segmentation.

Latencies are observed in every sensorial system of the brain. In particular, it has been reported that different luminance levels will induce different propagation delays such that the neuronal activity arrives with a different latency in the visual cortex (Levick, 1973; Bolz, Rosner, & Wässle, 1982; Sestokas, Lehmkuhle, & Kratz, 1987; Gawne, Kjaer, & Richmond, 1996). In our older studies we introduced this concept into a spiking neural network, and we showed that latencies strongly improve object segmentation in many cases. Top-layer (cortical) neurons of our network representing a bright object are active earlier than those representing a dark object. Consequently, neuronal assemblies that reflect these different objects can synchronize one after the other without mutual disturbance. Thus, these studies suggested that sensorial latencies could play a role in information processing in the brain, as have experimental studies in cat and monkey (Gawne et al., 1996; Wörgötter et al., 1996).

From the viewpoint of computer vision, however, the latency mechanism in the spiking neural network had only a conceptual character because the total CPU time allocated in order to analyze even simple scenes was very large, resembling that obtained with local update algorithms. Therefore, the question for us arose as to how to implement a visual latency mechanism and avoid this problem. This finally led us to the ECU algorithm, which by itself outperforms several other spin-lattice segmentation algorithms. In addition, we observed that its performance can still be enhanced by about 50% using visual latencies, and the effect is more pronounced for good signal-to-noise ratios.

10 Conclusions

The problems inherent in low-level image segmentation are so complex that the existence of a single optimal algorithmic solution is rather unlikely. Currently feasible, however, are attempts by which the performance limits of individual algorithmical classes are pushed forward in order to achieve better performance. The comparison of the different spin-lattice segmentation algorithms and the introduction of the ECU algorithms in this study were meant to contribute along this line. The second goal of this and our older work (Opara & Wörgötter, 1996) was to pursue a neuronal algorithmical concept (latencies) over different implementation stages in order to arrive at a solution that can be used under the close-to real-time requirements of computer vision problems.

Acknowledgments

We acknowledge the support of the Deutsche Forschungsgemeinschaft (grant WO388 4-2, 5-2, 6-1).

References

Binder, K., & Heermann, D. W. (1988). *Monte Carlo simulation in statistical physics.* Berlin: Springer-Verlag.

Blatt, M., Wiseman, S., & Domany, E. (1996). Superparametric clustering of data. *Phys. Rev. Lett., 76,* 18.

Bolz, J., Rosner, G., & Wässle, H. (1982). Response latency of brisk-sustained (X) and brisk-transient (Y) cells in the cat retina. *J. Physiol, 328,* 171–190.

Chen, S., Ferrenberg, A. M., & Landau, D. P. (1992). Randomness-induced second-order transitions in the two-dimensional eight-state Potts model: A Monte Carlo study. *Phys. Rev. Lett., 69,* 1213–1215.

Eckes, C., & Vorbrüggen, J. C. (1996). Combining data-driven and model-based cues for segmentation of video sequences. Paper presented at the *WCNN World Conference on Neural Networks*, San Diego.

Ernst, U., Pawelzik, K., & Geisel, T. (1994). Multiple phase clustering of globally pulse coupled neurons with delay. In M. Marinaro & P. G. Morasso (Eds.), *ICANN '94: Proceedings of the International Conference on Artificial Neural Networks* (Vol. 1, pp. 1063-1066). London: Springer-Verlag.

Gawne, T. J., Kjaer, T. W., & Richmond, B. J. (1996). Latency: Another potential code for feature binding in striate cortex. *J. Neurophysiol., 76*(2), 1356–1360.

Geman, D., Geman, S., Graffigne, C., & Dong, P. (1990). Boundary detection by constrained optimization. *IEEE Trans. Pattern Analysis Machine Intelligence, 12*(7), 609–628.

Gray, C. M., König, P., Engel, A. K., & Singer, W. (1989). Oscillatory responses in cat visual cortex exhibit inter-columnar synchronization which reflects global stimulus properties. *Nature, 338,* 334–337.

Ising, E. (1925). Beitragzur Theorie des Ferromagnetismus. *Z. Physik*, *31*, 253–258.

Kirkpatrick S., Gelatt Jr., C. D., & Vecci, M. P. (1983). Optimization by simulated annealing. *Science*, *220*, 671–680.

Levick, W. R. (1973). Variation in the response latency of cat retinal ganglion cells. *Vision Res.*, *13*, 837–853.

Li, S. Z. (1995). *Markov random field modeling in computer vision.* Berlin: Springer-Verlag.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equations of state calculations by fast computing machines. *J. Chem. Phys.*, *21*, 1087–1091.

Neal, R. M. (1993). *Probabilistic inference using Markov chain Monte Carlo methods.* (Tech. Rep. No. CRG-TR-93-1). Toronto: Department of Computer Science, University of Toronto.

Nischwitz, A., & Glünder, H. (1995). Local lateral inhibition: A key to spike synchronization? *Biol. Cybern.*, *73*, 389–400.

Opara, R., & Wörgötter, F. (1996). Using visual latencies to improve image segmentation. *Neural Computation*, *8*, 1493–1520.

Potts, R. B. (1952). Some generalized order-disorder transformations. *Proc. Cambridge Philos. Soc.*, *48*, 106–109.

Sestokas, A. K., Lehmkuhle, S., & Kratz, K. E. (1987). Visual latency of ganglion X- and Y-cells: A comparison with geniculate X- and Y-cells. *Vision Res.*, *27*, 1399–1408.

Swendsen, R. H., & Wang, S. (1987). Nonuniversal critical dynamics in Monte Carlo simulations. *Phys. Rev. Lett.*, *58*, 86–88.

von der Malsburg, C. (1981). *The correlation theory of brain function* (Int. Rep. 81-2). Göttingen: Department of Neurobiology, Max-Planck-Institute for Biophysical Chemistry.

von der Malsburg, C., & Schneider, W. (1986). A neural cocktail-party processor. *Biol. Cybern.*, *54*, 29–40.

Vorbrüggen, J. C. (1995). *Zwei Modelle zur datengetriebenen Segmentierung visueller Daten*. Frankfurt am Main: Verlag Harri Deutsch, Thun.

Wolff, U. (1989). Collective Monte Carlo updating for spin systems. *Phys. Rev. Lett.*, *62*, 361–364.

Wörgötter, F., Opara, R., Funke, K., & Eysel, U. (1996). Utilizing latency for object recognition in real and artificial neural networks. *NeuroReport*, *7*, 741–744.

Received April 2, 1997; accepted December 9, 1997.