Real-Time Imaging **6,** 195–211 (2000)
doi:10.1006/rtim.1998.0159, available online at http://www.idealibrary.com on IDEAL®

Image Segmentation Using Markov Random Field Model in Fully Parallel Cellular Network Architectures

Markovian approaches to early vision processes need a huge amount of computing power. These algorithms can usually be implemented on parallel computing structures. Herein, we show that the Markovian labeling approach can be implemented in fully parallel cellular network architectures, using simple functions and data representations. This makes possible to implement our model in parallel imaging VLSI chips.

As an example, we have developed a simplified statistical image segmentation algorithm for the Cellular Neural/Nonlinear Networks Universal Machine (CNN-UM), which is a new image processing tool, containing thousands of cells with analog dynamics, local memories and processing units. The Modified Metropolis Dynamics (MMD) optimization method can be implemented into the raw analog architecture of the CNN-UM. We can introduce the whole pseudo-stochastic segmentation process in the CNN architecture using 8 memories/cell. We use simple arithmetic functions (addition, multiplication), equality-test between neighboring pixels and very simple nonlinear output functions (step, jigsaw). With this architecture, the proposed VLSI CNN chip can execute a pseudo-stochastic relaxation algorithm of about 100 iterations in about 100 μs.

In the suggested solution the segmentation is unsupervised, where a pixel-level statistical estimation model is used. We have tested different monogrid and multigrid architectures.

In our CNN-UM model several complex preprocessing steps can be involved, such as texture-classification or anisotropic diffusion. With these preprocessing steps, our fully parallel cellular system may work as a high-level image segmentation machine, using only simple functions based on the close-neighborhood of a pixel.

Tamás Szirányi[1], Josiane Zerubia[2], LászLó Czúni[3], David Geldreich[4] and Zoltán Kato[5]

[1]*Comp. & Autom. Inst., Hung. Academy of Sci., MTA SZTAKI, Kende u. 15, H–1111 Budapest, Hungary*
E-mail: sziranyi@lutra.sztaki.hu
[2]*INRIA Sophia-Antipolis, BP 93, 2004 route des lucioles, 06902, Sophia-Antipolis Cedex, France*
E-mail: zerubia@sophia.inria.fr
[3]*Veszprém University, Department of Image Proc. & NeuroComputing, Veszprém, Hungary*
E-mail: czuni@silicon.terra.vein.hu
[4]*INRIA Sophia-Antipolis, BP 93, 2004 route des lucioles, 06902, Sophia-Antipolis Cedex, France*
E-mail: geldreich@sophia.inria.fr
[5]*Computer Science Department, Hong Kong University of Science & Technology, Clear Water Bay, Kowloon, Hong Kong*
E-mail: kato@cs.ust.uk

1077-2014/00/060195+17 $35.00/0

Introduction

Markovian models of images may help us to make better image restoration, enhancement or segmentation. However, using segmentation methods based on Markov Random Field (MRF) models requires a huge computing power and quite a lot of time. For these reasons, MRF methods are usually used in off-line tasks, never in a real-time processing environment. Recently multiprocessor arrays, transputers or distributed computer systems have been used for MRF processing, but all of them are power-consuming and still slow considering the standard video-rate.

On the other hand, a new image processing architecture, the fully programmable Cellular Neural (or Nonlinear) Network (CNN) array [9,27] has been introduced in the past few years. The cells in the array may contain photo-sensors, logical functions, local memories [12], every cell has analog dynamics and feedback/feed-forward convolution kernels with programmable parameters. Some recent simulation results show that the analog VLSI accuracy is satisfactory for tasks such as texture segmentation [30] and image deblurring [23,29] or MRF-based segmentation [31]. In the latter case we have very simple functions on the pixel-cell level, so CNN running MRF-based algorithms can easily be integrated in an upgraded version of real VLSI circuits [12,27]. This CNN architecture can be integrated by digital kernels as well [17], meaning higher accuracy and more complex functionality.

However, implementing a fully parallel MRF segmentation method in only one VLSI chip allows the use of a small instruction set on the pixel-cell level. Herein, we show that this can be solved using very simple VLSI functions such as multiplication, addition, logical operations, comparison and jigsaw functions. Several other preprocessing steps can be involved into the structure. We may get a complex image processing system containing MRF segmentation, using a fully parallel cell-array with a very reduced set of pixel-level functions.

In this paper we suggest a computational tool-kit, based on simple VLSI functions, to implement different levels and complexity of image processing methods containing MRF segmentation. This architecture of programmable set of instructions gives a more robust VLSI implementation of image processing tools than any task-specific [e.g. 7] solutions.

In the next section we show the basic MRF models and optimization methods used in our paper. We analyse them considering the global and local (pixel-level) processes, and consider how to implement these processes in a fully parallel system. We briefly describe the Metropolis Dynamics (MD) and the modified MD (MMD) optimization methods, which are the best to compress the labeling process into a simple cell-arithmetic. Then the parameter estimation for the Markovian labeling-decision is introduced, followed by a simplification using unsupervised pixel-level estimation of labeling parameters.

In ''Fully Parallel Cellular Network Arrays'', the general architecture of a fully parallel computing structure is shown, by considering computational cost and complexity.

In ''MRF using Cellular Nonlinear Network'', we describe the CNN architecture and our MRF implementations using MMD optimization. Alternatively, some monogrid and multigrid models are also described. Simulation examples on different images are shown to demonstrate the efficiency of this simple and analog computing structure. Some noise and accuracy analyses show the robustness of this system.

In the final section, we show some preprocessing methods, such as texture-labeling, deconvolution and nonuniform diffusion. These algorithms can be easily incorporated in the above architecture to make sophisticated image segmentation.

MRF Segmentation Models and Optimization

In image processing, MRF modeling has received a great deal of attention in the past decade [2,4–6,13–15, 19,20,33]. This type of modeling, originally introduced in vision by Geman and Geman [13], has been widely used for edge detection [33], image restoration [5], stereovision, long range motion and image classification [19].

For all these early vision processes, where the image is represented on a lattice, the problem is posed as a minimization of a cost function which is constructed from the observed data, *a priori* information on the world and constraints. The cost function obtained is usually non-convex and several relaxation techniques have been proposed to reach an optimum. The first group of methods deals with stochastic relaxation and is

based on Simulated Annealing (SA) [13,21]. These algorithms converge asymptotically towards the global minimum but require a great deal of computation. The second group of methods is related to deterministic relaxation. These techniques are suboptimal but require less computational time than the previous ones. That is why so many deterministic relaxation algorithms have been recently investigated (Graduated Non Convexity (GNC) [5], Iterated Conditional Mode (ICM) [4], Mean Field Annealing (MFA) [33], Modified Metropolis Dynamics (MMD) [20]).

Optimization upon energy function in the MRF model

First, we briefly give an introduction to the theory of MRF [1,25], then we describe a general image model used in the following sections.

Let $\mathcal{S} = \{s_1, s_2, \ldots, s_N\}$ be a set of sites (or pixels). $\mathcal{G} = \{\mathcal{G}_s \mid s \in \mathcal{S}\}$ is a neighborhood system for $\mathcal{S}$ if

1. $s \notin \mathcal{G}_s$
2. $s \in \mathcal{G}_r \Leftrightarrow r \in \mathcal{G}_s$.

A subset $C \subseteq \mathcal{S}$ is a clique if every pair of distinct sites in C are neighbors. In image processing the most commonly used neighborhood systems are the homogeneous systems. In this case, we consider $\mathcal{S}$ as a lattice and define these neighborhoods as

$$\mathcal{G}^n = \{\mathcal{G}^n_{(i,j)} : (i,j) \in \mathcal{S}\},$$
$$\mathcal{G}^n_{(i,j)} = \{(k,l) \in \mathcal{S} : (k-i)^2 + (l-j)^2 \leq n\}.$$

Obviously, sites near the boundary have fewer neighbors than interior ones. Furthermore, $\mathcal{G}^0 \equiv \mathcal{S}$ and for all $n \geq 0 : \mathcal{G}^n \subset \mathcal{G}^{n+1}$. Figure 1 shows a first-order neighborhood corresponding to $n = 1$. The cliques are $\{(i,j)\}, \{(i,j),(i,j+1)\}, \{(i,j),(i+1,j)\}$.

Let $\mathcal{X} = \{X_s : s \in \mathcal{S}\}$ denote any family of random variables so that $\forall s \in \mathcal{S} : X_s \in \Lambda$, where $\Lambda = \{1, \ldots, L\}$ is a common state space. Furthermore, let $\Omega = \{\omega = (\omega_{s_1}, \ldots, \omega_{s_N}) : \omega_{s_i} \in \Lambda, 1 \leq i \leq N\}$ be the set of all possible configurations. $\mathcal{X}$ is a MRF with respect to $\mathcal{G}$ if

1. for all $\omega \in \Omega$: $P(\mathcal{X} = \omega) > 0$,
2. for every $s \in \mathcal{S}$ and $\omega \in \Omega$:

$$P(X_s = \omega_s \mid X_r = \omega_r,\ r \neq s) = P(X_s = \omega_s \mid X_r = \omega_r,\ r \in \mathcal{G}_s).$$

The functions in 2 are called the *local characteristics of the MRF*, and the probability distribution $P(\mathcal{X} = \omega)$ of any process satisfying 1 is uniquely determined by these conditional probabilities. However, it is extremely difficult to determine these characteristics in practice. Gibbs distribution and the *Hammersley-Clifford* theorem provides us a simple way to overcome this problem.

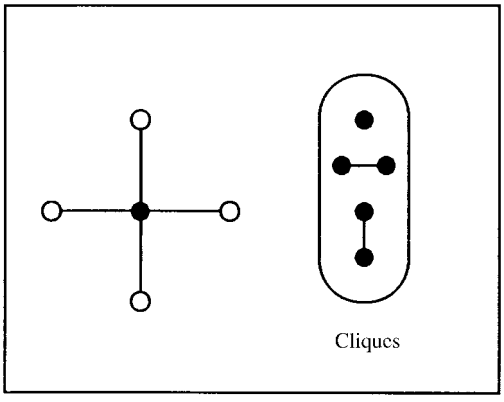


Figure 1. First order neighborhood-system with cliques.

A Gibbs distribution relative to the neighborhood system $\mathcal{G}$ is a probability measure π on Ω with the following representation:

$$\pi(\omega) = \frac{1}{Z} \exp\left(\frac{-\mathcal{E}(\omega)}{T}\right), \tag{1}$$

where Z is the normalizing constant or *partition function*:

$$Z = \sum_{\omega} \exp\left(\frac{-\mathcal{E}(\omega)}{T}\right),$$

T is a constant called the *temperature* and the *energy function* $\mathcal{E}$ is of the form

$$\mathcal{E}(\omega) = \sum_{C \in \mathcal{C}} E_C(\omega).$$

Each E_C is a function defined on Ω depending only on those elements ω_s of ω for which $s \in C$. The restriction of ω to the sites of a given clique C is denoted by ω_C. Such a function is called a *potential* . One of the most important theorem is probably the *Hammersley-Clifford* theorem which points out the relation between MRF and Gibbs distribution:

$\mathcal{X}$ is a MRF with respect to the neighborhood system $\mathcal{G}$ if and only if $\pi(\omega) = P(\mathcal{X} = \omega)$ is a Gibbs distribution with respect to $\mathcal{G}$.

Using the above theorem, the definition of the MRF is completed by the knowledge of the clique potentials $E_C(\omega_C)$ for every C in $\mathcal{C}$ and every ω in Ω.

Let us now briefly review a general MRF image model: given $\mathcal{F} = \{f_s\}_{s\in\mathcal{S}}$ a set of observed image data where f_s stands for the gray-level at pixel s. A very general problem is to find the labeling $\hat{\omega}$ which maximizes $P(\omega \mid \mathcal{F})$ based on the Bayes decision theorem. Following [20], we calculate the *energy* for a pixel using the MRF image model with the following energy function where the connection between clique potentials and probabilities is kept by the Hammersley-Clifford theorem:

$$\mathcal{E}(\omega,\mathcal{F}) = \sum_{s\in\mathcal{S}} \left(\ln(\sqrt{2\pi}\sigma_{\omega_s}) + \frac{(f_s - \mu_{\omega_s})^2}{2\sigma_{\omega_s}^2} \right) + \sum_{\{s,r\}\in\mathcal{C}} \beta\delta(\omega_s,\omega_r), \tag{2}$$

where

$$\delta(\omega_s,\omega_r) = \begin{cases} -1 & \text{if } \omega_s = \omega_r \\ +1 & \text{if } \omega_s \neq \omega_r \end{cases}$$

and β is a positive model parameter controlling the homogeneity of the regions of the image. Each class $\lambda \in \Lambda$ is represented by a Gaussian model, its mean value is μ_λ and its standard deviation is σ_λ.

Parallel solutions

MRF image segmentation methods are usually based on the local calculations of probability and potential functions. If we use a serial high-level machine with a strong computing power, all functions can be done at any complexity level. If we want to use parallel solutions, we should define the necessary complexity for local computation. Using the Connection Machine [16,19,20], a group of pixels is considered together through virtual processor assignment at high accuracy and complexity. It is a partly parallel solution. Using fully parallel machines of smart but reduced complexity cells, such as CNN, where the global computation is more difficult to process, we should redefine the task. Basically, the process could be divided into two families described below.

First, there are global processes, such as

1. G1: Image grabbing from camera or file.
2. G2: Checking the stopping conditions.
3. G3: Image saving or transfer.
4. G4: Image statistics computation (histograms, estimation of labeling parameters).
5. G5: Simulated annealing process-control.

Second, there are local processes using only a reduced neighborhood:

1. L1: Neighborhood labeling comparisons.
2. L2: Local energy calculation.
3. L3: Labeling decision.

An imaging cellular system can grab the image by itself, using on-board photo-sensors, such as the CNN VLSI chips [12]. Such a cellular network should be a real-time video device, so we do not need to deal with stopping conditions, since:

1. The process should be convergent in time, overtime is not a problem.
2. The process should be faster than the video-frame (for an analog CNN chip the whole MRF process is between 0.01–0.1 ms as simulations show for real VLSI parameters [12]).
3. We stop an iteration-series at a pre-defined but satisfactory number of steps.

The computation of image statistics and parameter estimation is a sequential function and it can be done in parallel with the image transfer by-fly. These statistical parameters together with the simulated annealing parameters must be supplied in parallel to every cell, which means that the system needs parallel data loading and controlling. The parallel data loading method is a basic function of the CNN VLSI chips.

However, as we show later, parameter estimation for labeling may be done on pixel-level, using a parallel unsupervised estimation method.

The local steps (L1–3) can be computed in a parallel process considering a small neighborhood only. In the case of the Gibbs Sampler for combinatorial optimization, a labeling process in a cell needs:

1. Energy calculation using logarithmic function, addition, multiplication;
2. Random number generation;
3. Exponential function, division;
4. Logical steps;
5. Local memories.

The above operations can be easily executed by a sequential processing computer, but it is far from technological limits if we introduce them into the cells of a huge pixel-array on a single chip. The recent digital array processor chips can process the most simple functions only such as addition and multiplication [17]. CNN-UM chips [12] may have on-board photo-sensors, memories and logical functions. We suggest an architecture containing the following items only:

1. Addition, multiplication;
2. Logical steps;
3. Local memories;
4. Simple stored functions (jigsaw, comparison).

Random number generation can be a pseudo-random process, only the first random numbers of the initial random-array are filled by serial process if necessary. Division is only done in preprocessing when the inverse of standard deviations of classes are calculated (see Eqn (2)). Using the MMD algorithm [20], we need only 7–8 memories per cell. The inverse of the variance and the mean of a possible level should be fed into the cells by parallel lines. If they are stored in pixel-memories, a cell needs additional memories counted twice the number of the possible labels. Later we show that only two additional memories are necessary if we use a simplified pixel-level parameter and label estimation method.

In the case of digital parallel machines with a serial controller and higher-level arithmetic CPU, the task of energy computation in a cell may be much easier. The CPU might calculate the histogram, and singleton energies (expressed by the first component of Eqn (2)) are computed for every label-class and every possible gray-level of f_s. These values are fed into the cells in parallel using a data-bus. The cells store the values which correspond to their f_s values. Following this storing process makes the whole cell-level computation very simple.

MMD as an easy-to-parallelize optimization solution

Basically, the MMD algorithm [20] is just a modified version of Metropolis Dynamics [22] which turns the algorithm into a pseudo-stochastic relaxation. The difference between the original Metropolis method and the MMD is the choice of the threshold ξ used in the dynamics to accept a new state. For the original method, ξ is chosen randomly at each iteration, however, in the MMD algorithm ξ is a constant threshold, say α, chosen at the beginning of the algorithm. This simply means that the jump to a new labeling state η is allowed if this does not increase the energy "excessively". The threshold α controls this increase of energy. The algorithm is highly parallel and can be described as follows:

1. In order to get the convergence of the algorithm, partition the entire image into disjoint regions $\mathcal{R}_n$ $(1 \le n \le M)$ such that pixels which belong to the same region are conditionally independent given the data of all the other regions. Pick randomly an initial configuration ω^0, with $k = 0$ and $T = T_0$.
2. Using a uniform distribution, pick a global state $\eta \in \Omega \setminus \{\omega^k\}$. For each site $s \in \mathcal{R}_n$ $(1 \le n \le M)$, the local energy $\mathcal{E}_s(\eta')$ is computed with $\eta' = [\omega^k_{s_1}, \omega^k_{s_2}, \ldots, \eta_s, \ldots, \omega^k_{s_N}]$.
3. Compute $\Delta\mathcal{E}_s = \mathcal{E}_s(\eta') - \mathcal{E}_s(\omega^k)$. The new label η_s at site s is accepted according to the following rule:

$$\omega^{k+1} = \begin{cases} \eta & \text{if } \Delta\mathcal{E} \le 0, \\ \eta & \text{if } \Delta\mathcal{E} > 0 \text{ and } \ln(\alpha) \le \left(\frac{-\Delta\mathcal{E}}{T}\right), \\ \omega^k & \text{otherwise} \end{cases} \qquad (3)$$

   where α is a constant threshold ($\alpha \in (0, 1)$), chosen at the beginning of the algorithm.
4. Decrease the temperature $T = T_{k+1}$ (k = number of iterations) and go to Step 2 if $\Delta\mathcal{E}_{glob} > threshold$.

There is no explicit formula for threshold α. In practice, α is determined empirically according to the following rule: in the case of a noisy image, α is chosen nearly equal to zero; otherwise, α is chosen nearly equal to one. If the temperature is less than a certain threshold ($\frac{\Delta\mathcal{E}_{min}}{-\ln\alpha}$), then only the jumps to states of lower energy are allowed. The algorithm converges to a local minimum [20]. In [20] different optimization methods have been tested and compared. The initial temperature for the algorithms using annealing (that is Metropolis or MMD) was $T_0 = 4$ and the schedule is given by $T_{k+1} = 0.95 \cdot T_k$.

This MMD method has a simple computational complexity that can really be implemented in a VLSI architecture.

Approximation for unsupervised parallel segmentation on CNN

We can use a lower number of local memories if we apply an approximation simplifying the original MMD algorithm. Herein, we introduce an easy-to-use automatic parameter estimation method.

We have made the assumption that we could work at the pixel level (because computation needs to be very local for the CNN). The observation at pixel s ($f_s = x_s + n_s$) is supposed to come from an original gray level value (x_s), disturbed by an additive Gaussian white noise (n_s). Furthermore, we assume that a crude approximation of x_s could be obtained by the following mean μ_s and standard deviation σ_s:

$$\mu_s = \frac{f_s + s_s}{2} \quad \text{and} \quad \sigma_s = \frac{|f_s - s_s|}{2}, \tag{4}$$

where s_s is a smoothed value of f_s obtained through anisotropic diffusion for instance [8,10,24]. Then, following Eqn (2), we use an *ad hoc* criterion to get a final segmentation $\hat{\omega}$:

$$\hat{\omega} = \min_{\omega \in \Omega} \left(\sum_{s \in S} \frac{(\mu_{\omega_s} - \mu_s)^2}{2\sigma_s^2} + \sum_{C \in \mathcal{C}_2} E_C(\omega_C) \right),$$

where μ_{ω_s} corresponds to the mean gray level value associated to the class ω at site s. It is automatically estimated considering the main peaks in the histogram of input (or smoothed input) image, while the logarithmic component of Eqn (2) can be neglected since it has a constant value in the comparison of local energy-functions.

Using the above equation it is easy to define the local energy of any labeling ω at site s:

$$\mathcal{E}_s(\omega) = \frac{(\mu_{\omega_s} - \mu_s)^2}{2\sigma_s^2} + \sum_{\{s,r\} \in \mathcal{C}_2} V(\omega_s, \omega_r), \tag{5}$$

where $V(\omega_s, \omega_r)$ is equal to $-\beta$ ($\beta > 0$) if $\omega_s = \omega_r$ and to $+\beta$, otherwise. A large value of β will favor the formation of homogeneous regions.

The estimation of $\hat{\omega}$ is done through the energy minimization using an MMD algorithm.

In the above unsupervised estimation we consider two different scales of the image resolution: original and smoothed. It brings the scale-space train of thought [8] into the theory, especially when the smoothing effect is a nonuniform diffusion [8,10,24], see the preprocessing section.

Fully Parallel Cellular Network Arrays

In this section we show a fully parallel architecture that is as simple as possible, but which executes the MMD optimization without any approximation. The statistical parameters (μ_λ, σ_λ) of the different ($\lambda \in \Lambda$) classes can be estimated by the histogram of the original image. In very noisy cases, smoothing of the image [10,32] can remove the high-frequency noise-content from the contiguous areas. It makes the histogram peaks of the different regions more separable, so the statistical parameters of the different regions can be better estimated by using the histogram of the smoothed image.

In the original digital solution a parallel two-color MMD algorithm was used [20]. In this cellular solution we can implement a fully parallel algorithm. According to [2], we know that the algorithm will converge as long as less than 100% of the pixels are updated at the same time. It is true in our case, except for events which occur with almost zero probability.

Another key-point is the generation of a random map for each iteration. First, a real random map is generated, e.g. using a serial filling in from a noise-generator through a nonlinear quantizer (considering the possible μ_{ω_s} values). Starting from this map, we generate the next random map using a common *random* offset value for the whole array and a jigsaw like nonlinear pixel-function. This function maps all the $[-\psi, +\psi]$, $[-2*\psi, -\psi]$ and $[+\psi, +2*\psi]$ intervals into $[-\psi, +\psi]$, where ψ is the amlitude interval of λ and f. The consecutive random elements of the same site are independent because of the random offset, while neighbors are independent because of the initialization. There is some statistical correlation between the different sites due to the deterministic pixel-based function of the random generator. However, as we have found in the simulations described in the following section, practically it does not influence the results.

General model using a simple memory-function for the Singleton energies

In the case of digital parallel machines with a serial controller and higher-level arithmetic CPU, the task of the energy computation in a cell may be a simple memory-function. The CPU might calculate the histogram of the whole image and the characterizing μ_{ω_s} values. Singleton energies are computed for every label-class and every possible gray-level of f_s. These values are fed into the cells in parallel through a data-bus. The cells store the values which correspond to their actual f_s values. Using this storing process makes the whole cell-level computation simpler and faster. In the case of 10

classes, data-precision of 8 bits and memory-transfer of 50 ns, the whole loading time is: $10 * 256 * 50\,\text{ns} = 128\ \mu\text{s}$. Considering the simplification in the cell-level computing the whole iteration process becomes simpler. This solution needs L additional memories/cell and fully parallel data loading.

Model for homogeneous variance

We can suppose that class-variances in images are frequently uniform for the different regions. In this case, $1/\sigma_\lambda^2$ is the same for all ($\lambda \in \Lambda$) different classes. We can apply an important simplification in the cell-level computation, since the denominator and the additive constant in the first part of Eqn (2) is the same for the different classes. Now, the whole energy computation can be done in a simple cell-operation. Figure 2 shows the timing-memory scheme for the initialization and the process kernel. M# denotes the numbering of the seven local memory-sites in the cells. Dashed boxes show the initial arrays (processed only once), while dotted boxes show the initial maps to be cancelled later. Ovals denote the pixel-level functions (random offset, jigsaw function, step function, MMD decision), analog convolution-based function (smoothing), nonlinear neighborhood detection (equality tests) and parameters (constants: β and α, time-dependent: temperature). In the initialization, the μ_ω values are calculated for each class and $1/\sigma_{image}^2$ for the image.

If we use the original *Metropolis Dynamics*, then an additional cell-memory is necessary to make another probabilistic value instead of the global α. Step function is a series of "*if* " and "*less than*" and "*greater than*" relations.

In this configuration a cell contains the following things:

1. 7 (MMD) or 8 (MD) memories in every cell
2. addition, multiplication
3. comparison between the neighbors
4. logical and jigsaw functions
5. some smoothing in a small neighborhood (if necessary)

In this configuration we have the next global loading lines, parallel to each cell:

1. inverse variance and mean values of the different classes
2. 1 (MMD) or 2 (MD) signals containing the probabilistic offset values
3. T temperature line for simulated annealing
4. β value for weighting the neighborhood energy

MRF using Cellular Nonlinear Network

The CNN architecture gives a very high speed tool for parallel window-based dynamic processing of spatially discretized 2D or 3D data-structures. The CNN structure is well-suited for image processing. The normalized differential state-equation of the analog IIR-convolution effect of the local analog dynamics [9,27] can be described with matrix-convolution operators [29,30]:

$$\frac{d\boldsymbol{X}}{dt} = -\boldsymbol{X} + \boldsymbol{A} * \boldsymbol{Y} + \boldsymbol{B} * \boldsymbol{U} + \boldsymbol{J} \tag{6}$$

$\boldsymbol{U}, \boldsymbol{X}, \boldsymbol{Y}$ are the $M \times N$ input, state and output matrices while $\boldsymbol{J}$ is an $M \times N$ offset matrix. Here, the boundary condition is $\boldsymbol{U}, \boldsymbol{X}, \boldsymbol{Y} = 0$ if $(i,j) \notin [0,0; M, N]$. There is a non-linear function between the state and the output, $Y_{i,j} = f(X_{i,j})$, where $f(.)$ is usually a sigmoid. $*$ denotes convolution implemented by weighted local interconnections in a VLSI chip. $\boldsymbol{A}$ and $\boldsymbol{B}$ represent the feedback and feed-forward connections.

In the image processing model of the CNN, the feedback convolution is responsible for the deconvolution [23,28–30] or the pattern shifting [27] capability of the CNN structure.

The members of convolution kernels $\boldsymbol{A}, \boldsymbol{B}$ and $\boldsymbol{J}$ (if it is uniform) together are called templates [9,27].

The CNN Universal Machine (CNN-UM) [12,27] chip architecture contains local memories and logical/arithmetic functions between the different image-levels, and global programs (template series) and on-chip A/D converters. Special local nonlinear functions can be embedded in the structure if they are easily produced by simple VLSI methods.

MMD implementations with some approximations

The original MMD algorithm [20] was implemented on a Connection Machine (CM200) [16] with 8K processors.

Once we have implemented the proposed model into a VLSI environment, functions are related to the special

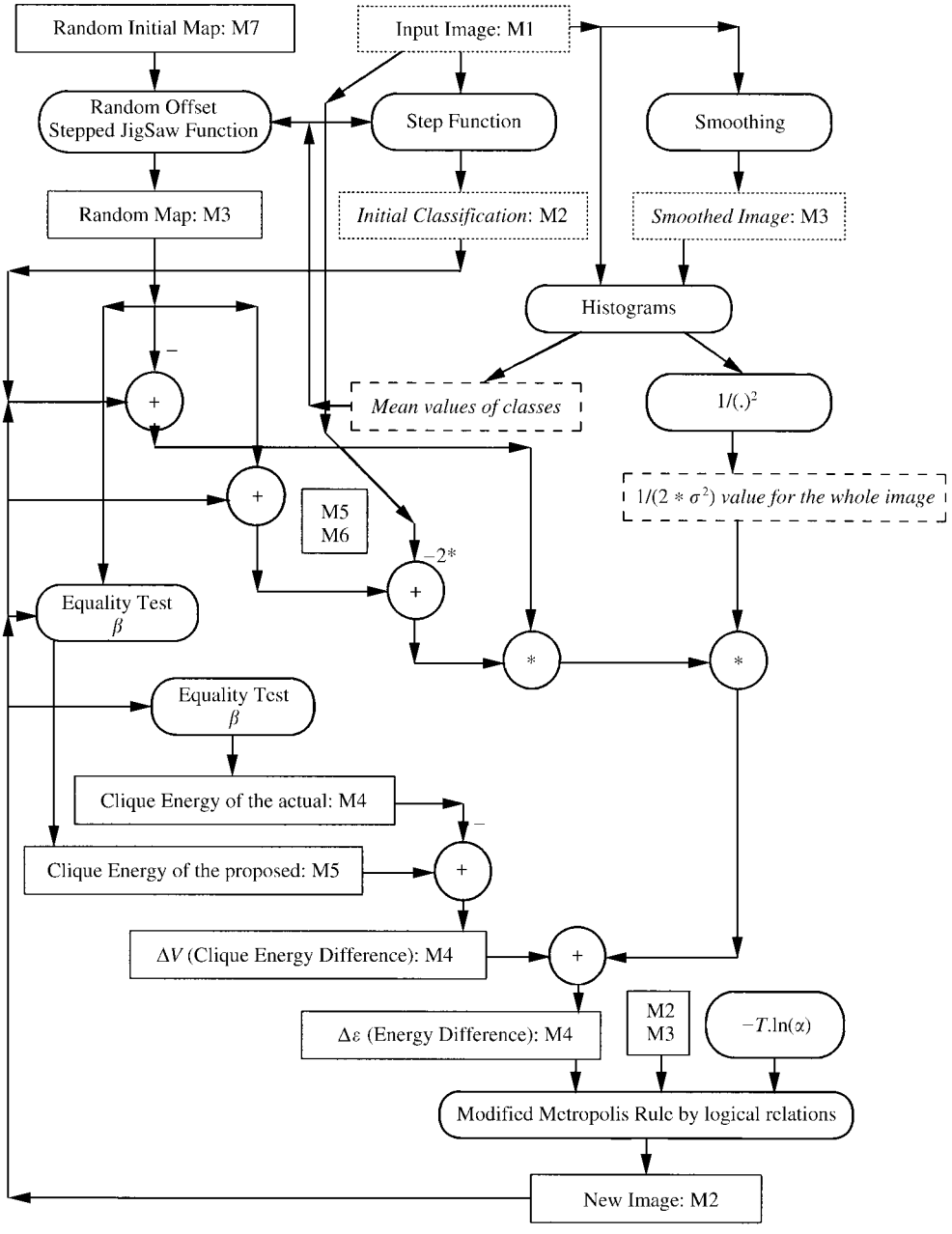


Figure 2. Architecture of Markov Random Process and MMD optimization, considering homogeneous class variance, in a fully parallel cellular system using seven local memories and simple functions.

hardware requirements. The use of probabilistic maps, energy functions and Gaussian distributions should be adapted to the VLSI constraints. Our CNN MRF model contains only primitive functions, such as: jigsaw, window and step functions. It uses two arithmetic functions: addition and multiplication. A simple equality detector is used for each neighbor of clique-assignment C_2 in the energy calculus.

In an analog hardware it is not easy to implement data-driven memory addressing. For this reason, we should avoid the use of stored label-data in the local

memories. Calculation of a probability is simpler in VLSI architecture with our approximations of Eqn (4, 5) (minimum number of local memories and simple functions). In some close-to-the-trivial cases our model is equivalent to the original Bayesian–Gaussian model [13,20].

Monogrid model

In order to be able to implement the proposed method on CNN we had to introduce important modifications [31] with respect to the original model proposed in [20]:

1. first, we worked with a larger neighborhood (3rd order MRF (12 neighbors), using only the cliques of order 1 ($\mathcal{C}_1 \equiv S$) and 2 ($\mathcal{C}_2$), i.e. the singletons and the doubletons).
2. second, we have made the assumption that we could work at the pixel level (if computation needs to be very local for the CNN). The energy optimization is described by Eqn (3).

Our method calculates the energy-difference for each pixel situation based on Eqn (5). The variances in the denominators are the same for a given pixel through the process. Let μ_{ω_s} be the mean gray-level of class ω at pixel s. The difference can be calculated using only additions and multiplication:

$$\begin{aligned}(\mu_{\omega_s^{k+1}} - \mu_s)^2 - (\mu_{\omega_s^k} - \mu_s)^2 \\ = (\mu_{\omega_s^{k+1}} - \mu_{\omega_s^k})(\mu_{\omega_s^{k+1}} + \mu_{\omega_s^k} - 2\mu_s).\end{aligned} \tag{7}$$

The energy part related to the doubletons $\mathcal{C}_2$ in Eqn (2) is calculated using a simple equality detector for each neighbor ($\omega_s = \omega_r$ or $\omega_s \neq \omega_r$) for the state ω_s^k or estimation ω_s^{k+1}. Summing the two parts of energy, values of α and the temperature are used in a comparison to detect if ω_s^{k+1} is accepted or if ω_s^k remains. In this case, it is checked whether $\Delta\varepsilon \leq -T_{k+1} ln(\alpha)$ (then ω_s^{k+1} is accepted); if not, ω_s^{k+1} is rejected. We only need a threshold comparison because $-T_{k+1} ln(\alpha)$ is a general value for the whole array for a given step.

Figure 3 shows the timing-memory scheme for the initialization and the process kernel. M# denotes the numbering of the eight local memory-sites in the CNN-UM architecture [27]. Every step of the kernel process can be done in a CNN-UM. In the initialization the μ_s and $1/\sigma_s^2$ values are calculated for each pixel. Currently, the division is not a CNN step in VLSI, but it can be done in a serial line. However, some restricted division can be implemented in VLSI using a nonlinear pixel-function with some limited accuracy. Due to the nature of the σ calculus no high precision is needed for the division.

Experiments on monogrid CNN MRF model

The experiments have been executed on a simulator system using a fixed-point hardware accelerator board in PC [26]. It has limited accuracy, so we should normalize the values to avoid over- or under-flow problems. Since the bounding error and the nonlinear saturation of the CNN simulator result in some computational errors, an implemented VLSI system should likely be robust [29].

In Figure 4, we show the segmentation results using a noisy input test image (SNR = 5dB). Parameters are $T_{k+1} = 0.95\ T_k$, $\alpha = 0.3$ and $\beta = 10.0$. The misclassification error is 1.5%. Using the original MMD algorithm [20] on CM [16], the error is 1.3%, and it is 1.0% with the Metropolis algorithm. We have checked the segmentation error (considering the un-noisy original image) with respect to the number of iteration steps for different α values. We have found that the process is convergent and settles in about 100 iteration steps.

In Figure 5, the unsupervised segmentation of a SPOT satellite image is shown. We defined 4 output classes. Parameters are $T_{k+1} = 0.95\ T_k$, $\alpha = 0.3$ and $\beta = 0.5$.

Robustness

In [29] it has been shown that the analog structure of the CNN is highly robust against parameter noise, image noise, the imperfect estimation of parameters and parameter accuracy in processes including feedback effects, such as the tasks of image deblurring and texture segmentation.

Robustness of the MRF process on the CNN array has been tested by using a scaling factor in the representation of energy. It causes saturation errors in the calculations. In a wide dynamic range (two magnitudes) the performance became stable. It shows the fact that the MRF segmentation is self-consistent: emerging computational errors are suppressed in the consecutive iterations.

Multigrid models

Nowadays multiresolution, multiscale, hierarchical approaches are widely applied in the field of image processing. Markov Random Fields are one typical area where the advantages of such techniques seem to be

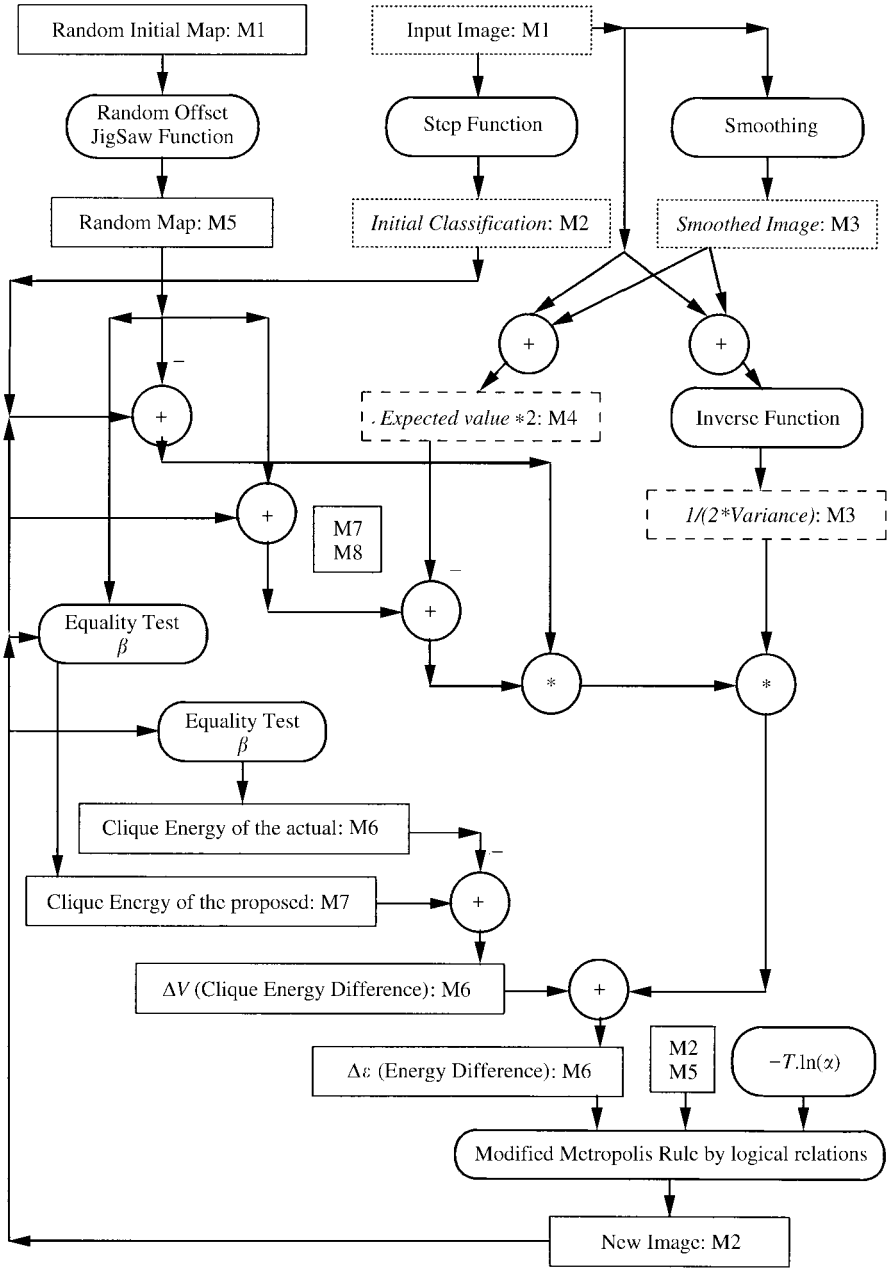


Figure 3. Architecture of Markov Random Process using CNN-UM with 8 local memories and simple functions when the labeling decision is a simplified approximation like in Eqns (4, 5).

tremendous. A review on this widespreading area can be found in [14].

Here are some reasons why multigrid models are preferable:

1. in a general case, many phenomena (e.g. fractal signals) have intrinsic multiscale properties
2. in our case, we face an optimization problem with possible local minima. Optimization with a multiscale/hierarchical model avoids being trapped into local minima, resulting in faster convergence and becoming less sensitive to initial configurations.

Generally speaking, the common feature of multiscale models is the representation of images on several levels

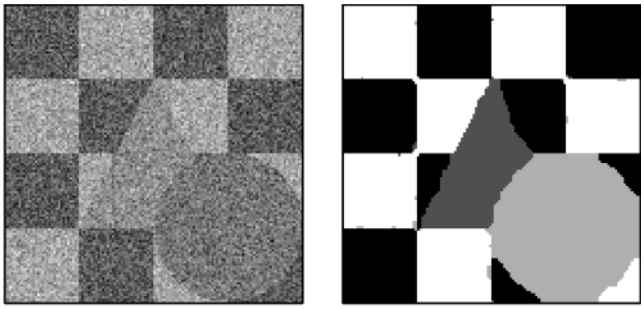


Figure 4. Monogrid segmentation.

with decreasing resolution. There can be significant differences in the definition of cliques and energy functions in the different approaches (see [14] for details).

The VLSI implementation of the CNN-MRF model with third-order neighborhood system can be technologically costly, on the other hand, as we experienced, systems of first-order neighborhood with unsupervised pixel-level parameter-estimation do not give satisfactory results. What we expect from multigrid implementations is to reduce neighborhood connectivity of the monogrid MRF model at comparable results and the ratio of number of operations/iteration ratio.

Multiscale model

The following multiscale model has been introduced by Perez *et al*. in [15]. Now, we are giving only a very short description of this model, but still to be able to understand its main idea.

In this model, we have a top-down strategy from coarser representation of the image to finer scales (we used $2 * 2$ sites to build up a coarser block). Optimization is started on the coarsest level and the obtained equlibrium state serves as the initial state for optimization of the finer layer below. During the optimization process there are no interactions between the scales except for the initial data. Important feature of this model is the calculation of clique potentials: at a given scale it is done through the cliques of the finest scale: always the cliques of the finest scale are considered for computation, but with a restriction that the blocks of n^{2i} sites have the same value on the layer being optimized (n is the rescale ratio, i is the processed level). Cliques of the finest scale, corresponding to a certain level under optimization, are partitioned into two sets: one set contains the cliques which are included in any block of the processed level (their energy contribution is given by $p^i\beta$ in Eqn (8)), whereas, the other set of cliques contains the ones which sit astride any two neighboring blocks (Eqn (9)). Clique potentials of a given level are the sum of these clique potentials of the finest scale.

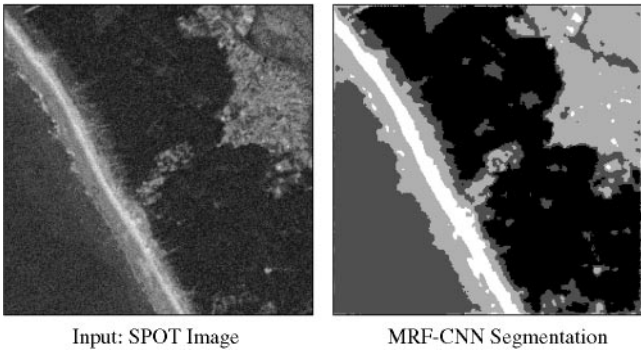


Figure 5. Satellite image segmentation, four classes.

The following form of energy components at level i represents the above description:

$$U_1^i(\xi^i, \mathcal{F}) = \sum\nolimits_{s^i \in S^i} V_1^i(\xi_{s^i}^i, (\mathcal{F})),$$

where

$$V_1^i(\xi_{s^i}^i, \mathcal{F}) = \sum\nolimits_{s \in b_{s^i}^i} \left(\log(\sqrt{2\pi}\sigma_{\omega_s}) + \frac{(f_s - \mu_{\omega_s})^2}{2\sigma_{\omega_s}^2} \right) - p^i\beta \tag{8}$$

and

$$U_2^i(\xi^i) = \sum\nolimits_{C^i=(r^i,s^i)\in\mathcal{C}^i} V_2^i(\xi_{C^i}^i)$$

$$\begin{aligned} V_2^i(\xi_{C^i}^i) &= \sum\nolimits_{(r,s)\in D_{C^i}} V_2(\omega_r, \omega_s) \\ &= \begin{cases} -q^i\beta & \text{if } \omega_r = \omega_s \\ +q^i\beta & \text{if } \omega_r \neq \omega_s \end{cases} . \end{aligned} \tag{9}$$

The sum of U_1^i and U_2^i gives the energy of level i. It is similar to Eqn (2) but in the multiscale model doubletons are also represented in U_1^i . Some explanations with respect to notations:

1. $\xi_{s^i}^i$ means the labeling of one block at scale i,
2. $s \in b_{s^i}^i$ means sites which build up a block at scale i,
3. C^i is one clique while $\mathcal{C}^i$ is the set of all cliques at scale i,
4. D_{C^i} is the set of the sites which build up clique C^i,

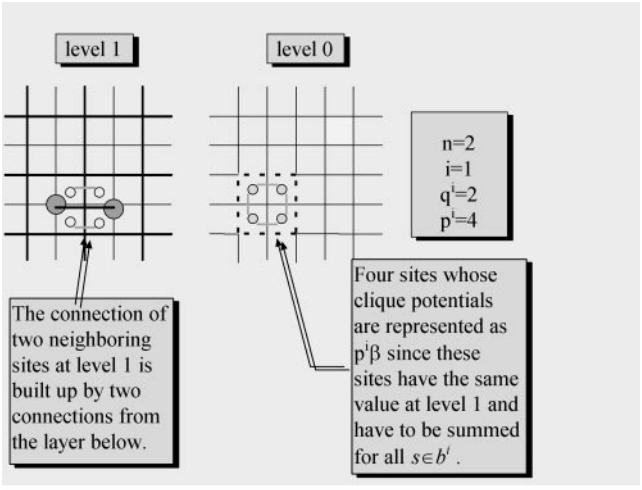


Figure 6. Multiscale cliques.

5. The number of cliques included in the same block at scale i is $p^i = 2n^i(n^i - 1)$, while the number of cliques between two neighboring blocks is $q^i = n^i$.

(In the equations, we consider blocks of size $n * n$ and a first-order neighborhood system.) Figure 6 represents the two kinds of cliques. The segmentation algorithm consists of two main steps:

1. energy optimization of a layer
2. initialization of the next layer from a coarser layer above.

Figure 7 illustrates them.

The adaptation of multigrid models to CNN

Besides Perez' model, we also tested a very simplified version of the Causal Hierarchical Model [6]. Here, a label-pyramid of decreasing number of sites is built up and a so-called *sequential MAP* estimation is used during optimization. In the case of four levels connected by vertical neighborhood only, we can achieve a very fast and low-complexity architecture. However, this simplified structure results in higher segmentation error ($>10\%$) for the test image of Figure 4 (left). In this simplified Causal Hierarchical model, no horizontal neighborhood system was taken into account in the energy computation. The reason of using only vertical connectivity in the Causal Hierarchical Model was the increased organizational requirements for other cases: considering additional local (same level) neighborhood

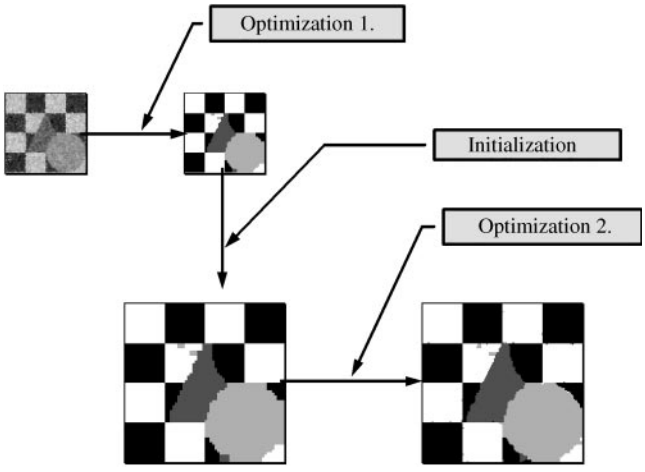


Figure 7. Main steps of multiscale segmentation.

information would result in a too complex, practically un-implementable cellular structure.

CNN-multiscale model

Now, we will show two implementations of the multiscale model described previously. They need more local memories and functional complexity but smaller neighborhood than in the case of the monogrid model to achieve good segmentation results. There is no direct interaction between two scales, the connection is kept by the initialization step and by clique potentials as explained before. This multiscale algorithm uses a first-order neighborhood system.

There are two possible algorithmic implementations of the model we built [11].

1. One is considering the model as a system with several scales where cliques originate from the finest layer. Only the current image scale being processed is represented in the memory as a CNN layer.
2. In the other implementation, we tried to implement what is behind the formulae: we had only one scale represented on one CNN layer and instead of building up a coarser (smaller) grid, we restricted the values so as to be the same in every $2 * 2$ block ($n = 2$).

Since both techniques are based on the same theory, we got similar results during testing, however, they have different complexity and different computational time. In both models, MMD optimization was used with various *ad hoc* parameters.

As Figure 8 illustrates, there is a significant increase in segmentation quality compared to the first-order monogrid model, best results of the first-order multiscale model are quite close to the results of the third-order monogrid model.

In Figure 9, we can see a real-image segmentation. The input image (a) is a part of an image from Airborne Multisensor Pod System Data Access Catalogue (http://info.amps.gov:2080/). Segmentation was done with a 2-level multiscale method at 2*80 iteration steps.

Characteristics of the Implemented Models

Table 1 roughly summarizes the capacities needed by the the monogrid and two multiscale models.

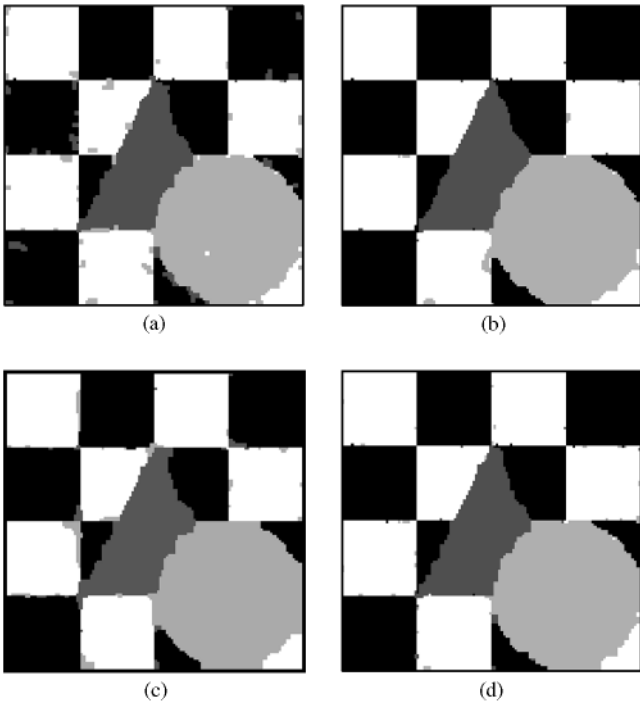


Figure 8. Segmentation results applying different smoothing methods in the preprocessing and different MRF models on the input image of Figure 4 (left): (a) *Monogrid* model, *first-order* neighborhood, 150 iteration steps, *nonuniform* smoothing in the preprocessing, misclassification error: 3.7%. (b) *Monogrid* model, *third-order* neighborhood, 150 iteration steps, *nonuniform* smoothing in the preprocessing, misclassification error: 1.6%. (c) *Multiscale* (2 scales), *first-order* neighborhood, 2*80 iterations, *uniform* smoothing in the preprocessing, misclassification error: 3.5%. (d) *Multiscale* (2 scales), *first-order* neighborhood, 2*80 iteration steps, *nonuniform* smoothing in the preprocessing, misclassification error: 1.7%.

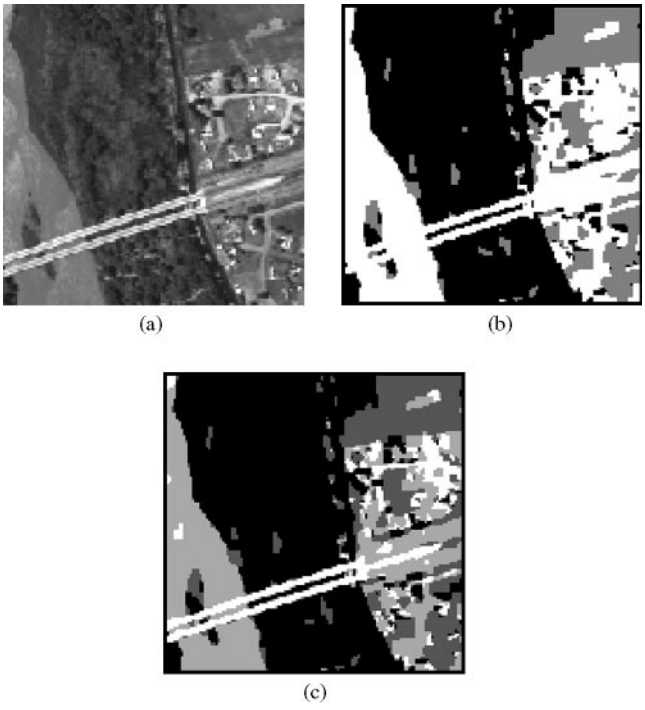


Figure 9. Multiscale MRF segmentation: (a) Air view of a scene with river, bridge, forest, green area and town (from left to right). Rio Grande, New Mexico, Segmented into: (b) 3 classes, (c) 4 classes.

The need for multiscale CNN-MRF model was set up mainly in order to reduce neighborhood connectivity of the monogrid model at comparable segmentation efficiency.

To sum up the results multiscale models need more sites of memory per cell but need smaller neighborhood connectivity than the monogrid model to achieve a good result. To decide which one to use in a VLSI chip environment depends on the available technological potential.

Preprocessing

There are several attempt to introduce different image processing methods in the MRF framework: texture analysis and synthesis [3], motion detection [7] or restoration from blurred and noisy images [18]. These combined methods are usually very specific for the given task. Sometimes, the composite algorithm is effective only for binary imaging effects, such as simple motion-detection or deconvolution to get binary image segments [18].

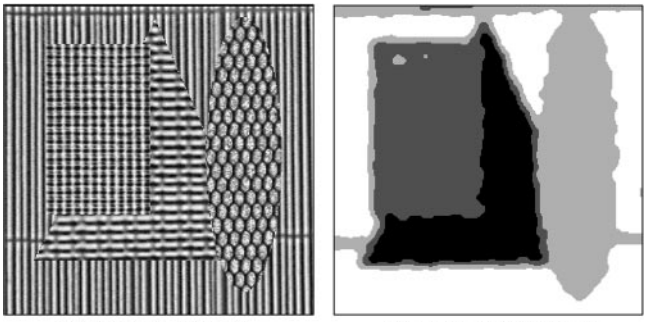
Table 1. Some characteristics of the models, # stands for the finest scale, + for the coarse scale

	Monogrid	Multigrid 1.	Multigrid 2.
Memories/Cell	8	15	10
Operations/Iterations	8	$24^{+}/8^{\#}$	13
Processed Image Size (inputs is of $N * N$)	$N * N$	$(N/2) * (N/2)^{+}$; $N * N^{\#}$	$N * N$

In our approach we tried to give some solutions for the continuous-valued image models using a general MRF segmentation combined with different parallel preprocessing enhancement methods. In the same cellular system, we can implement many of the preprocessing methods, such as deblurring [23,29], texture-featuring [29,30] and anisotropic diffusion [8,10,24,28]. Now, we show two examples of complex preprocessing.

Texture labeling

When instead of light-intensity the local-structure of the image (textures) is to be separated the (analog) cellular model is capable of pixel-level image segmentation considering the different texture-classes [29,30]. However, the output of such a segmentation may be quite noisy owing to the relatively small detection window, which is necessary to get sharp edges. The local inhomogeneities can be removed by using a MRF model. The CNN-based texture-recognizer method [29,30] performs well for many texture classes. Figure 10 shows a texture segmentation result based on the CNN texture segmentation filtering method [29,30]. Here, we use a single $\mathbf{3} \times \mathbf{3}$ analog CNN template as a spatial filter to make different (average) gray-tones for the different textures of similar flat histograms in the input. The segmentation and misclassification errors on the texture-borders and inside the textures are removed using the MRF method. The effect of foreign stripes at the border of two different textures can be removed using several texture-detector CNN templates [30].



Input image: 4 texture — CNN texture filtering and MRF cleaning

Figure 10. Texture segmentation in the CNN-UM architecture using one CNN filter and MRF post-processing.

Nonlinear/nonuniform diffusion

In the case of uniform heat-diffusion on the image, we have the following equation whose solution is a convolution of the signal with a Gaussian kernel:

$$\frac{\delta u(t,x,y)}{\delta t} = \Delta u(t,x,y), \quad u(0,x,y) = u_0(x,y) \tag{10}$$

In further improvement of this model [24] we take the local geometry of the image into consideration as described by the following equation:

$$\frac{\delta u(t,x,y)}{\delta t} = \nabla\{g(|\nabla u(t,x,y)|)\nabla u(t,x,y)\} \tag{11}$$

In this case, the anisotropic diffusion explicitly depends on the local gradient (as a function of g) leading to edge conserving property as seen in Figure 8(c, d) for the multiscale case, using Figure 4 (left) as the input image. To improve the effectiveness of the CNN-MRF model our aim was to decrease the probability of misclassification at the borders of the different homogeneous regions. For this reason, we introduced a nonlinear and nonuniform diffusion as the smoothing process. The following simple formula [28] was applied for the CNN where a new image F_{k+1} is obtained from the previous state F_k:

$$F_{k+1} = F_k + h\Delta F_k(1 - |\nabla G(F_k)|), \tag{12}$$

where Δ is the Laplace operator and ∇ is the grad operator which is normalized to 1 and $G(.)$ is a Gaussian pre-smoothing [8], h is a scaling. This means that, where the gradient is high, the smoothing of the image is small. In addition, the gradient calculation should have been preceded by a uniform smoothing (G), otherwise, the gradient would be effected by undesirable noise.

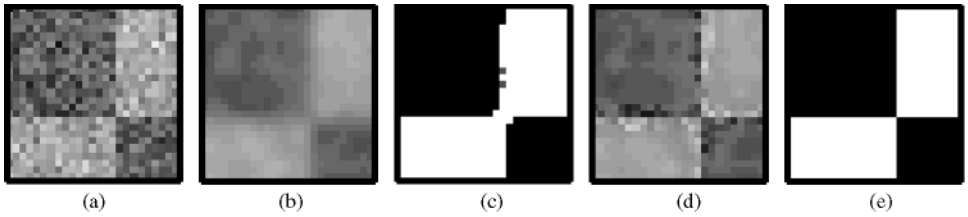


Figure 11. Effect of nonuniform diffusion in the preprocessing: (a) Original input image part. (b) Smoothed input. (c) MRF segmentation after smoothing. (d) Nonuniform diffusion on the input image. (e) MRF segmentation following the nonuniform diffusion step.

The effects of using nonuniform diffusion in the CNN-MRF model can been seen in Figure 11. In many cases, the preservation of edges can be very crucial for the segmentation procedure. A good example can be seen in Figure 12 representing a street scenery.

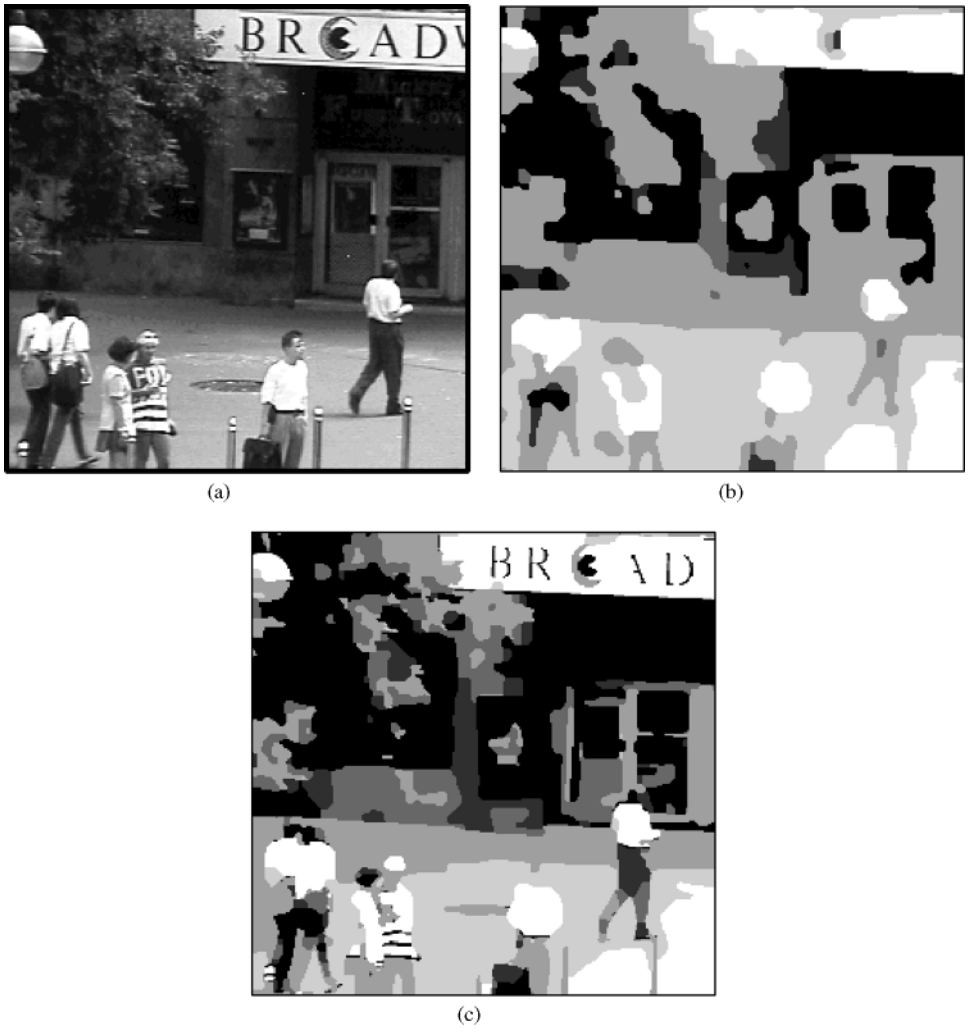


Figure 12. The role of the nonuniform diffusion in the preprocessing image segmentation and enhancement using monogrid MRF segmentation: (a) Input Street Scenery. (b) CNN-MRF segmentation preprocessed by *uniform* diffusion. (c) CNN-MRF segmentation preprocessed by *nonuniform* diffusion.

Table 2. Computational complexity and the required processing-time considering 100 ns for a simple operation and 1 μs for an analog filtering process

	Complexity of an Iteration Simple Operation/Analog Filter	Number of iterations	Processing time
MRF Monogrid	8/0	150	120 μs
MRF Multigrid	13/0	100	130 μs
Nonlinear Diffusion	5/1	10	15 μs
Texture Filtering	2/2	1	2.2 μs
Deblurring	0/1	1	1 μs

The nonlinear diffusion in the smoothing process and the multiscale MRF segmentation (see Figure 8) bring closer the two different multiresolution methods in one framework.

Conclusions

We have shown that MRF models can be implemented using simple VLSI functions and a parallel cellular array. The whole time-consuming process can be embedded in a fully parallel, high-speed architecture of simple functions. Using an analog solution, the CNN-UM architecture, the original MRF models have been modified to fit the requirements of analog VLSI implementations. These modifications do not change the performance too much. This analog-chip architecture is very robust. Segmentation is based on an unsupervised labeling method where local statistics are considered for the label-classes. This structure is ready for implementation on VLSI CNN chips. In Table 2 we show the computational complexity and the required processing-times of the different processing steps.

In the proposed fully-parallel architecture the MRF segmentation of images can be a basic step, as a part of a programmable multi-function architecture.

Acknowledgements

This work was partially supported by ''Balaton'' programme of the National Committee for Technological Development, Hungary and the French Ministry of Foreign Affairs and by the Hungarian Scientific Research Fund (OTKA No. 025374 and 019062).

References

1. Azencott, R. (1987) Markov fields and image analysis. *Proc. AFCET, Antibes,* 1987.
2. Azencott, R. (1992) *Parallel Simulated Annealing: Parallelization Techniques*. Wiley.
3. Bader, D.A. Jala, J., & Chellappa, R. (1995) Scalable data parallel algorithms for texture synthesis using Gibbs random fields. *IEEE Tr. Image Processing*, **4**: 1456–1460.
4. Besag, J. (1996) On the statistical analysis of dirty pictures. *Journal of Royal Statist. Soc*. **B-68**: 259–302.
5. Blake, A. (1989) Comparison of the efficiency of deterministic and stochastic algorithms for visual reconstruction. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **11**: 2–12.
6. Bouman, C.A. & Shapiro, M. (1994) A Multiscale Random Field Model for Bayesian Image Segmentation. *IEEE Trans. Image Proc*. **3**: 162–177.
7. Caplier, A., Luthon, F. & Dumontier, C. (1998) Real-time implementations of an MRF-based motion detection algorithm. *Real-Time Imaging*, **4**: 41–54.
8. Catte, F. Lions, P.L. Morel, J.M. & Coll, T. (1992) Image selective smoothing and edge detection by nonlinear diffusion. *SIAM J. Numerical Anal*. **bf 29**: 182–193.
9. Chua, L.O. & Yang, L. (1998) Cellular Neural Networks. *IEEE Trans. Circuit and Systems* **35**: 1257–1290.
10. Czúni, L. & Szirányi, T. (1996) Picture segmentation with an anisotropic preliminary step to an MRF model with Cellular Neural Networks. *Proc. of 13th ICPR, Vienna, 1996. D*: pp. 366–370.
11. Czúni, L., Szirányi, T. & Zerubia, J. (1997) Multigrid MRF Based Picture Segmentation with Cellular Neural Networks. *Proc. CAIP'97, Lect. Notes in Comp. Sci*. **1296**: 345–352
12. Dominguez-Castro, R., Espejo, S.M., Rodriguez-Vazquez, A., Carmona, R., Földesy, P., Zarándy, Á, Szolgay, P., Szirányi, T. & Roska, T. (1997) A 0.8 mm CMOS 2-D Programmable Mixed-Signal Focal-Plane Array Processor with On-Chip Binary Imaging and Instructions Storage. *IEEE J. Solid State Circuits* **32**: 1013–1026.
13. Geman, S. & Geman, D. (1984) Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Trans. Pattern Analysis and Machine Intelligence* **6**: 721–741.
14. Graffigne, C., Heitz, F., Pérez, P., Preteux, F., Sigelle, M. & Zerubia, J. (1995) Hierarchical Markov random field models applied to image analysis: a review. *SPIE Conf*., *San Diego*, 1995.
15. Heitz, F, Perez, P. & Bouthemy, P. (1994) Multiscale minimization of global energy functions in some visual recovery problems. *CVGIP: Image Understanding*, **59**(1): 125–134.

16. Hillis, W.D. (1985) *The Connection Machine*. MIT Press.
17. Ikenaga, T. & Ogura, T. (1996) Discrete time Cellular Neural Networks using highly parallel 2D cellular automata CAM. *Proc. of NOLTA'96*, *Japan*, *1996*, *221 pp*.
18. Kadaba, S.R., Gelfand, S.B. & Kashyap, R.L. (1996) Bayesian decision feedback for segmentation of binary images. *IEEE Tr. Image Processing*, **5**(7), 1163–1178.
19. Kato, Z., Zerubia, J. & Berthod, M. (1996) A Hierarchical Markov Random Field Model and Multitemperature Annealing for Parallel Classification. *Graphical Models and Image Processing*, **58**(1): 18–37.
20. Kato, Z., Zerubia, J. & Berthod, M. (1992) Satellite image classification using a modified Metropolis dynamics. *Proc. of ICASSP*, *San Francisco*, *1992*.
21. Kirkpatrick, S., Gellatt, C. & Vecchi, M. (1983) Optimisation by simulated annealing. *Science* **220**: 671–690.
22. Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H. & Teller, E. (1953) Equation of State Calculations by Fast Computing Machines. *J. of Chemical Physics* **21**(6) 1087–1092.
23. Miller, J.P., Roska, T., Szirányi, T., Crounse, K.R., Nemes, L. & Chua, L.O. (1994) Deblurring of Images by Cellular Neural Networks with applications to Microscopy. *Proc.*, *3rd IEEE Workshop on CNN and their Applications*, *Rome*, *1994*, pp. 237–242.
24. Perona, P. & Malik, J. (1990) Scale space and edge detection using anisotropic diffusion, *IEEE Tr. Pattern Analysis amd Machine Int.*, **12**(7): 629–639.
25. Rosanov, Y. (1982) Markov Random Fields. *Springer Verlag*.
26. Roska, T., Bártfay, G., Szolgay, P., Szirányi, T., Radványi, A., Kozek, T., Ugray, Zs. & Zarándy, Á. (1992) A digital multiprocessor hardware accelerator board for Cellular Neural Networks. *Int. J. Circuit Theory and Application*, **20**: 589–599.
27. Roska, T. & Chua, L.O. (1993) The CNN Universal Machine: An Analogic Array-Computer. *IEEE Trans. Circuits and Systems*, **40**(3 (II.)): 163–173.
28. Roska, T. & Szirányi, T. (1995) Classes of analogic CNN algorithms and their practical use in complex image processing tasks. *Proc. IEEE Nonlinear Signal and Image Proc.*, pp. 767–770.
29. Szirányi, T. (1996) Robustness of Cellular Neural networks in image deblurring and texture segmentation. *Int. J. Circuit Theory and Appl.*, **24**(5): 381–396.
30. Szirányi, T. & Csapodi, M. (1998) Texture Classification and Segmentation by Cellular Neural Network using Genetic Learning. *Computer Vision and Image Understanding*, **71**(3): 255–270.
31. Szirányi, T. & Zerubia, J. (1997) Markov Random Field Image Segmentation using Cellular Neural Network, *IEEE Tr. Circuits and Systems I.*, **44**: 86–89.
32. Szirányi, T., Zerubia, J., Geldreich, D. & Kató, Z. (1996) Cellular Neural Network for Markov Random Field Image Segmentation. *Proc. of CNNA'96, IEEE, Seville*, pp. 139–144.
33. Zerubia, J. & Chellappa, R. (1993) Mean field approximation using Compound Gauss-Markov Random Field for edge detection and image estimation. *IEEE Trans. Neural Networks* **8**: 703–709.