Real-Time Imaging 7, 77–95 (2001)
doi:10.1006/rtim.2000.0222, available online at http://www.idealibrary.com on IDEAL®

Motion Segmentation and Tracking with Edge Relaxation and Optimization using Fully Parallel Methods in the Cellular Nonlinear Network Architecture

In this paper we outline a fully parallel and locally connected computation model for the segmentation of motion events in video sequences based on spatial and motion information. Extraction of motion information from video series is very time consuming. Most of the computing effort is devoted to the estimation of motion vector fields, defining objects and determining the exact boundaries of these objects. The split and merge segmentation of different small areas, those obtained by oversegmentation, needs an optimization process. In our proposed algorithm the process starts from an oversegmented image, then the segments are merged by applying the information coming from the spatial and temporal auxiliary data: motion fields and motion history, calculated from consecutive image frames. This grouping process is defined through a similarity measure of neighboring segments, which is based on intensity, speed and the time-depth of motion-history. There is also a feedback for checking the merging process, by this feedback we can accept or refuse the cancellation of a segment-border. Our parallel approach is independent of the number of segments and objects, since instead of graph representation of these components, image features are defined on the pixel level. We use simple VLSI implementable functions like arithmetic and logical operators, local memory transfers and convolution. These elementary instructions are used to build up the basic routines such as motion displacement field detection, disocclusion removal, anisotropic diffusion, grouping by stochastic optimization. This relaxation-based motion segmentation can be a basic step of the effective coding of image series and other automatic motion tracking systems. The proposed system is ready to be implemented in a Cellular Nonlinear Network chip-set architecture.

László Czúni[1] and Tamás Szirányi[2]

[1]*University of Veszprém, Department of Image Processing and Neurocomputing, H-8200 Veszprém, Egyetem u. 10, Hungary*
E-mail: czuni@silicon.terra.vein.hu
[2]*Analogical and Neural Computing Laboratory, Comp. & Automation Inst., Hungarian Academy of Sciences, H-1111 Budapest, Kende u. 13-17, Hungary*
E-mail: sziranyi@sztaki.hu

Introduction

In this paper we demonstrate a fully parallel methodology to solve motion segmentation problems by low-level algorithms based on limited local neighborhood connectivity. Generally, this class of tasks requires both low-level and high-level optimization procedures with a huge amount of computing power. Our efforts are aimed at finding solutions to these problems that need almost low-level, simple functions that can be implemented on special parallel VLSI architectures with superior speed. Then, the output of these low-level operations can be forwarded to a high-level processor responsible for controlling the whole operation and for final interpretation. Since most of the work would be done on a parallel processor array, significant speed-up could be achieved compared to other processor architectures as shown in later sections.

Parallel architectures and cellular arrays

There is a change in the implementation of special hardware architectures for image processing tasks. While in the past a lot of effort has been made to design real-time hardware systems, today most of those problems can be solved with high performance general-purpose personal workstations at low cost. On the other hand there is an increasing demand for a new generation of image analysis tasks, such as second generation video coding [1, 2], virtual reality, etc. These tasks are still far from being in the range of the computing power of conventional Complex Instruction Set Computers (CISC) in the near future. While new CISC processors, such as the Pentium MMX family, speed up multimedia applications tremendously, there is still a gap between real-time multi-object oriented video coding algorithms and the performance of these digital processors. One possible breakthrough is the use of cellular processor arrays (analog or digital) which render a simple processor for each pixel resulting in fully parallel operation. Contrary to the CISC class of processors, which can also show some parallel features as well, cellular processor arrays can utilize only simple pixel based functions defining a relatively narrow instruction set — although at a very high speed.

One simple example is Mitsubishi's so called "Artificial Retina" [3] or other smart pixel arrays in optical computing. These digital architectures' functionality is very limited but similar systems can forecast a more intelligent class of smart cameras of the future. The typical example for the higher cell complexity of cellular processor arrays is the family of the Cellular Neural/Nonlinear Network (CNN) chips [4, 5]. CNN can process images locally on the pixel level with small neighborhood connectivity. It can perform convolution, nonlinear (sigmoid) dynamics, etc. in a feedforward/feed-back operation mode. CNN Universal Machine (CNN-UM) [6] is a programmable computer based on the basic CNN architecture with many additional features like those possessed by conventional computers local and global memories, pixel-level logical and arithmetic functions, digital memories etc., all in a single chip.

Investigating the success of general-purpose high performance processors in the image processing area, it seems to be reasonable to develop new, low-level algorithms for this new class of cellular processor arrays, since they can be considered as general purpose image processing architectures with superior speed, integrated into VLSI. The key question is how to develop a successful cellular image processing system containing local connections and reduced instruction set, considering the limitations of the cell complexity of VLSI implementation. Some results have been achieved for the segmentation of moving image parts by using optical-flow estimation and Markov Random Field (MRF) in the CNN architecture using fully parallel and local functions [7]. Now, our goal is to develop an object-segmentation method using edge relaxation and better optical-field estimation.

Our previous work on energy optimization on parallel processor arrays

Fully parallel cellular processor arrays can be used for many low-level image processing tasks. However, a more interesting question is how complex tasks can be decomposed into low-level operations. This is not always completely possible, since a CISC-like computer is always needed at least for controlling the program cycles or setting global parameters.

In [8, 9] it is described how a high-complexity energy optimization problem can be approximated by a solution based on local operations. It has been shown that a Markovian labeling approach can be implemented in a fully parallel cellular network using simple functions and data representations. The main features of the algorithm are:

- Instead of global parameter estimation, pixel-level statistical estimation is used.

- For pseudo-stochastic relaxation the so-called Modified Metropolis Dynamics (MMD) are applied [10].
- Monogrid and multigrid implementations are investigated.
- Simple functions are used: addition, multiplication, equality-test, simple nonlinear functions (step, jigsaw).
- Speed-up in case of hardware realization is about 10–100 μs for 100 iteractions.
- Little degradation in segmentation quality: 1.5% segmentation error is achieved instead of 1.1% for a typical test image.

Although this optimization algorithm is not the key element of the current article, later on more details are revealed to ensure the clear understanding of similar stochastic optimization subroutines that can be used for segmenting motion fields.

Tasks of motion segmentation and tracking

The jobs in motion segmentation and tracking can be classified into the following main groups: estimation and segmentation of motion fields, spatio-temporal segmentation and tracking. Although there is a sequential order of these tasks in some way, an implicit, iterative estimation can result in much better performance. Now, we overview these groups briefly. The investigation of literature and our answers to these problems can be found in the next two sections.

Spatio-temporal segmentation can be considered as the most important information source to detect or to track different moving objects in a video sequence. One important application is the analysis of image sequences for the encoding of objects in the MPEG4 environment. It uses both spatial and motion information and by combining the two, it may answer questions such as whether two image areas belong to the same object or not. This is often impossible to answer correctly if only spatial or temporal information is employed. On the other hand, many times it is difficult to make these decisions anyway, since the two information sources cannot only complement but can also conflict each other. A more perfect solution would be the use of geometric models of objects but that is beyond this work.

Estimation of optical flow fields can be crucial for the performance of the whole problem of spatio-temporal segmentation since the estimated motion field is a basic input to all higher level algorithms. For this estimation it is necessary to assume some constraints such as the local homogeneity of optical flow vectors. We want to obtain a smooth motion field so estimation should be followed by segmentation or estimation and segmentation should be run simultaneously.

Tracking motion information means that not only the current velocity of video objects is analysed but also other characteristics such as the trajectory of moving objects or the motion in the past frames. But since we want to keep our attention on low-level processing rather than on object-level motion analysis we neglect trajectory information but utilize the history of motion on a local level.

The organization of the paper

The following two sections discuss the details of our model. First, main building blocks are listed and explained. Some of these blocks contain stochastic optimization algorithms (Markov Random Field (MRF) based segmentation, gradient based motion segmentation), while others show some nonlinear features (nonlinear diffusion). Basic operations are also listed that are needed by the subroutines. Then we take a short review of the literature of motion estimation and explain the parallel implementations and we also explain how the accumulated motion information of the past frames is involved in our model. The next section contains the iterative cycle of spatio-temporal optimization with details of how multi-modal information is utilized. Then, some experimental results can be found. The last section gives VLSI speed and complexity estimations based on software simulations [11] and measured physical running times of already realized functions on the CNN-UM [6]. A summary of our work is then presented. Figure 1 shows two sample frames used in our experiments.

Main Building Blocks of the Method

This section lists those image processing functions that are used as the building blocks of the whole processing cycle. These sub-tasks, such as finding edges, filtering noise and estimating motion parameters, can be considered as subroutines for solving the spatio-temporal segmentation problem.

Elementary functions

In our model, the key elements can be executed in parallel on all image pixels and can be realized by a set

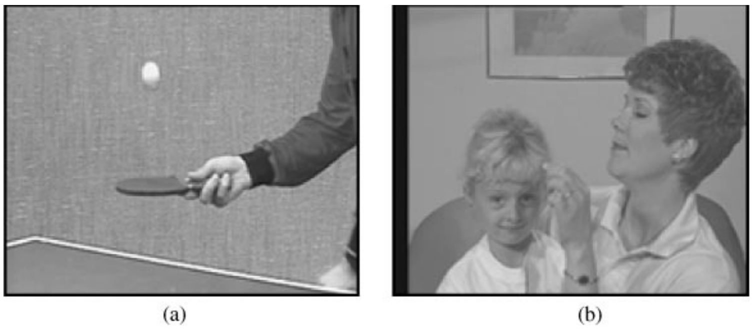

Figure 1. Samples from the video sequences used in our tests. (a) “Table Tennis”, (b) “Mother and Daughter”.

of VLSI functions in analog circuits [4]. An important limit of physical realization is the radius of local connectivity. We use only first (four neighbors are connected) or second order (eight neighbors are connected) neighborhood relations, higher order connectivity would make it difficult to design the circuits for hardware manufacturers. In some cases larger connectivity [8] can be replaced by multi-scale representation and processing, e.g. [9, 12] show some results of multi-scale processing in a pixel-level MRF environment.
Cell functions and components:

- Comparison of neighboring pixels.
- Convolution: a basic function already realized in many CNN chips. Since it is often used in many image processing algorithms its execution speed is a key parameter.
- Arithmetic and logical functions, relations: these functions are also core elements as they are executed on the whole pixel array simultaneously. Their implementation also calls for fast parallel processors.
- Cell memories: results of arithmetic and logical functions are stored locally in cell memories that can be analog or logical (storing only binary values). Naturally, the memory per pixel has an upper limit depending on hardware technology. So we should keep it as low as possible.
- Non-linearities: absolute value, gradient, etc.

Nonlinear diffusion

Anisotropic (nonlinear) diffusion [13, 14] is an effective tool for image enhancement. The main idea can be well approximated by the next nonlinear state equation [15,16]:

$$\frac{dI_{xy}(t)}{dt} = \mathbf{D}iv\mathbf{G}rad\left(\left[I_{xy}[1-k|\mathbf{G}rad(\mathbf{S}(I(t)))|]_{xy}\right]\right) \quad (1)$$

Here, the state-matrix $\boldsymbol{I}$ (an image) is processed by the $\mathbf{D}iv\mathbf{G}rad$ heat-diffusion modified by the gradient-content of the actual image (with $\mathbf{S}$ pre-diffusion to avoid noise-effects), where k is a normalization constant. Figure 2 shows a sample image as the result of this modified diffusion. Edges are preserved while noise is cleaned up.

Since other non-linear filters, such as rank order filters, can also be implemented in our parallel framework [17], we will use them (e.g., median filter) for oversegmentation purposes.

Stochastic optimization on cell arrays

Since the work of Geman and Geman [18] there is a lot of examples where stochastic optimization and Bayesian approaches are used for solving image labeling problems. However, the idea of parallel, low-level cooperating processes is much older, many basic ideas were already reviewed in [19].

In a previous work [8, 9], we gave an MRF model with very simple functions to solve a general segmentation problem based on local observations of image intensity. For the easier understanding of the motion segmentation algorithms in the latter sections, now, we outline this gray-scale segmentation algorithm. In this case intensity information is applied, but generally the algorithm can be used for the segmentation on any scalar two-dimensional field, e.g., the magnitude of motion vectors, as illustrated by Figure 3. To introduce pixel based statistics we assumed that the segmented image should be between our observation and the

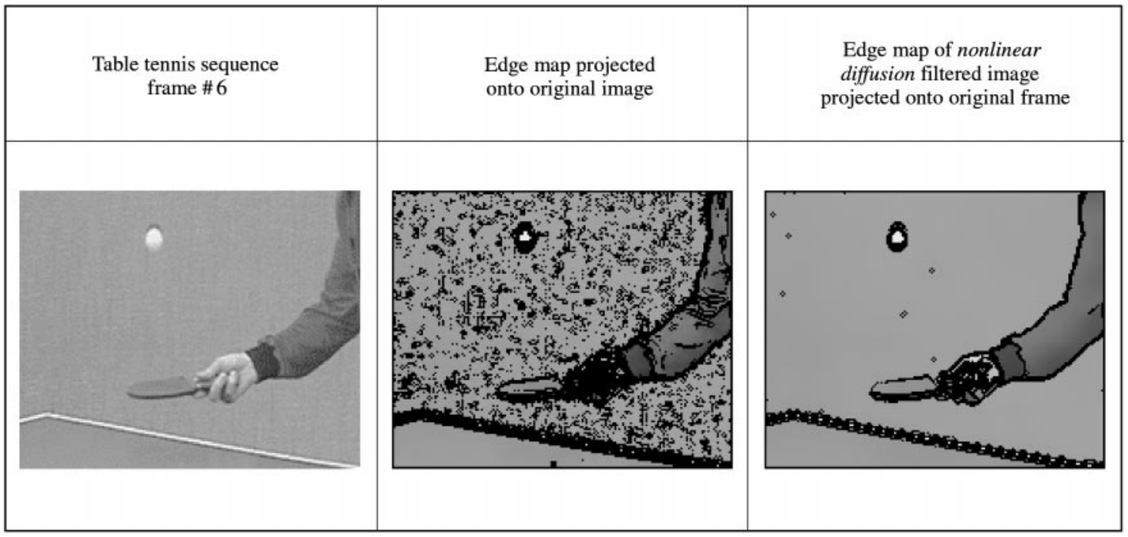
Table tennis sequence frame # 6
Edge map projected onto original image
Edge map of *nonlinear diffusion* filtered image projected onto original frame

Figure 2. The effect of the proposed nonlinear diffusion for the enhancement of main edges.

smoothed version of the observation. This assumption was described with local statistics: for each pixel s expected value (μ_s) and variance (δ_s) of intensity in a neighborhood were calculated. Then we constructed an energy model of two components to be minimized in a pseudo-stochastic optimization process: The first component was responsible to characterize the deviation of a new random state (see 2nd point of algorithm below) from the local statistics and the second one to ensure local homogeneity. This way the local energy of any labeling ω at pixel s is defined as:

$$E_s(\omega) = \frac{(\mu_{\omega_s} - \mu_s)^2}{2\delta_s^2} + \sum_{\{s,r\}\in C} U(\omega_s, \omega_r), \qquad (2)$$

where μ_{ω_s} corresponds to the gray level value associated with the class ω_s. $U(\omega_s, \omega_r)$ is equal to $-\beta$ if $\omega_s{=}\omega_r$ and to $+\beta$ otherwise (β is a fixed parameter). $\{s, r\}\in C$ means that the pixels s and r should be in the same clique C (a set of sites is called a clique if any two distinct sites in the clique are neighbors). As for the energy optimization, we proposed to use the Modified Metropolis Dynamics (MMD) [10] algorithm. MMD is a pseudo-stochastic relaxation process where new states are generated randomly and these new states are accepted or rejected:

1. Pick up a random initial configuration ω^0 and set temperature: $T{=}T_0$
2. Generate a random label (using a uniform distribution over the set of labels) and compute the energy difference ΔE_S for each new label η_s. The new label at pixel s is accepted according to the following rule:

$$\omega_S^{k+1} = \begin{cases} \eta_S, & \textit{if } \Delta E_S \le 0 \\ \eta_S, & \textit{if } \Delta E_S > 0 \textit{ and } \ln(\alpha) \le \left(-\frac{\Delta E_S}{T}\right) \\ \omega_S^k & \textit{otherwise}. \end{cases}$$

   where α is a constant, $\alpha\in\,]0, 1[$ and it is chosen at the beginning of the algorithm.
3. $k{=}k{+}1$, decrease T and repeat step 2 unless the global energy change is below a threshold.

To achieve good segmentation results a 2nd order neighborhood clique system was used in a monogrid model, but as it is also shown, multi-layer implementation can reduce these relations to 1st order connections. Memory requirements were between eight and 15 memories per cell, depending whether the mono- or multigrid model was used.

Figure 3 shows the application of this segmentation method to the scalar field of velocity vector magnitudes. The direct transition of this optimization model to motion vector segmentation is limited, since the optimization of optical vector fields would involve too large number of possible classes. In a later section we will give modified models to avoid this problem.

Estimating the motion displacement field

Our approach of motion segmentation is based on estimating and segmenting the dense optical vectors rather than using an image motion model of six or eight

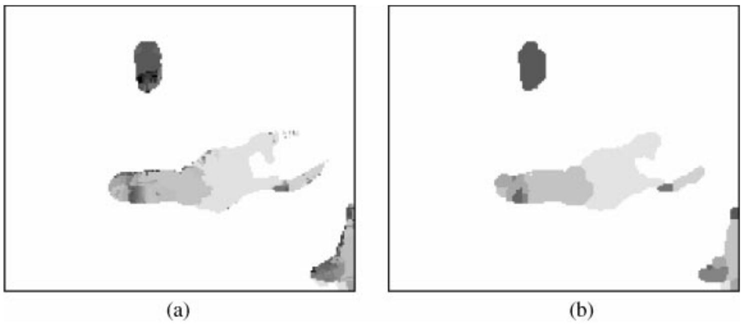

Figure 3. (a) Magnitude of optical vectors, (b) segmented with the MRF model to 9 classes. The optimization ran for 50 iterations. First the motion field was estimated with the correlation technique, then it was segmented with the MRF based method.

parameters as in [20]. This is not a serious limitation, since, if needed, camera ego-motion can be determined from motion vectors too. Instead of using a 2D or 3D parametric model, projections of vector flow fields can be used for ego-motion estimation as described in [21]. On the other hand, our model can be described with a small number of parameters reducing computation demands.

Basically, there are three main approaches to estimate dense optical vectors without involving a model of six or eight parameters:

- In correlation or block matching techniques each small patch of the image is compared with nearby patches in the next frame.
- Gradient based algorithms compute optical vectors from spatial and temporal derivatives of image intensity.
- Spatio-temporal filtering methods estimate optical flow in the frequency domain.

We have investigated several optical flow estimation and segmentation techniques from the aspects of parallel implementation. In [22] and [23] good comparisons on computation complexity and performance can be found. Correlation based techniques are widely used in MPEG applications but their serial implementation is very time consuming. If optimization is required, iterative or relaxation algorithms are used to obtain reliable results [24, 25]. However, the iterative revaluation of the displaced frame difference increases time complexity and may not fit our computation model based on local interactions. Generally, correlation techniques seem to be rejected in the simple-instruction-multiple-data (SIMD) model that we addressed, but now we show that if no optimization is needed then our fully parallel architecture can compute vector estimates with the correlation technique.

According to [22], gradient based approaches have good accuracy. However, it is also well known that these methods need a large temporal support (best results needed 15 frames to be stored in memory simultaneously). In [26] a recursive filter is designed to avoid the need for large memories, and they report similar results in accuracy as the original model.

In [27] filter banks had been built for spatio-temporal filtering. Their method is well suited for parallel implementation and has good accuracy as reported, but its application could be limited due to the large number of filters needed for general image flows.

In the next sections we describe two models for the estimation and segmentation of motion information. Both solutions have about the same complexity and can be used with stochastic relaxation steps. The first is a correlation technique, which does not combine motion field estimation with optimization into one step. The second is basically a gradient approach; it jointly estimates and classifies optical vectors through an optimization process.

A fast parallel correlation technique

If motion field estimation itself is reliable then it is not always necessary to combine the estimation and segmentation into one process. Its main advantage and disadvantage originates from the same fact: we do not re-evaluate motion information during segmentation. Obviously, this is computationally more effective but no sophisticated algorithm ensures the confidence of results. Since with this method we can still achieve good segmentation, results can be satisfactory for many motion based applications.

In this approach, the most time consuming task is the computation of the displaced frame difference, or the so-called sum-of-squared-differences (SSD):

$$E(x_0, y_0, t+1, V(x_0, y_0, t+1)) = \sum_{x,y \in N_{x_0,y_0}} [I((x, y, t+1)) - I(x - V_x(x_0, y_0, t+1), y - V_y(x_0, y_0, t+1), t)]^2 \quad (3)$$

That is the SSD at point x_0, y_0 at time $t+1$ is calculated by shifting a small neighborhood of the previous frame with the supposed motion vector $V(x_0, y_0, t+1)$ (by V_x and V_y components accordingly).

To speed up this search to find the most appropriate vector with the least SSD value, there are two basic approaches: The first is to reduce the number of patches (by critical features) used in the matching, the second is to use sophisticated search methods to avoid a full search. Instead of these techniques we propose a five-step algorithm, where each step can be easily implemented in cell array architectures:

1. *Spiral* movement of the whole current frame. In each step the current frame is shifted with one pixel position in a spiral order. So the spiral scan (see Figure 4) is performed for each neighborhood N_{x_0,y_0} simultaneously.
2. Subtraction from the next frame to get the difference image.
3. Self-multiplication to get the square.
4. Smoothing in a local neighborhood with a heat diffusion or convolution.
5. If the resulted correlation value is smaller than the previously stored reference value on the pixel level, store it as a new reference and store the recent parameters of the spiral-offset too, as the motion vector.

This way not only one patch but all pixels' neighborhoods are correlated with the succeeding frame in one computation step of the processor array. To run the search for all image pixels in a 5×5 window, 24 steps of one-pixel shifts (in spiral order) of the image frame is necessary. See Figure 3 showing motion fields for the "Table Tennis" sequence obtained with the correlation technique.
One possible answer for noise filtering is to apply statistical change detection [28]. The differences (changes) of succeeding image frames are smoothed and thresholded. This threshold can be based on a general noise model or on the specific noise parameter of the camera. Where no change is detected by statistical change detection between two subsequent frames, the displacement can be neglected. Figure 5 illustrates motion of the "Mother and Daughter" sequence in the $x(5a)$ and $y(5b)$ directions. The corresponding filtered motion fields ($5c$ and $5d$) were obtained with statistical change detection.

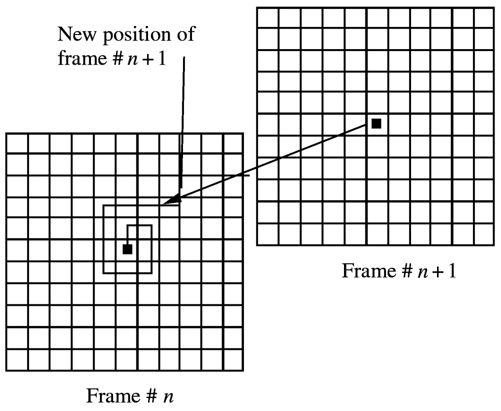

Figure 4. The spiral movement of the image frame over the preceding frame. Position after the series of steps: up, right, down, down, left, left, up, up, up, right, right, right.

Optimization of the motion field is not possible during motion field estimation in the correlation approach, since the iterative revaluation of the SSD does not match the spiral-translation model. Instead, segmentation is carried out after the estimation process as illustrated in Figure 3.

A gradient based method: simultaneous estimation and segmentation by energy minimization

When estimated motion information is not satisfactory due to heavy noise and temporal aliasing, then we should run optimization during segmentation. In this case, we reconsider motion information that can be very time consuming in many cases. In [24] and [25] this technique is used with the correlation approach. On the contrary, we have chosen the gradient approach for optimization, because the continuous revaluation of the SSD in the correlation based model does not fit our parallel framework. In other words: the spiral movement of the whole image bridged the problem of communication of processing elements for the calculation of the SSD, but such an optimization would require the repetition of this spiral image-translation process.

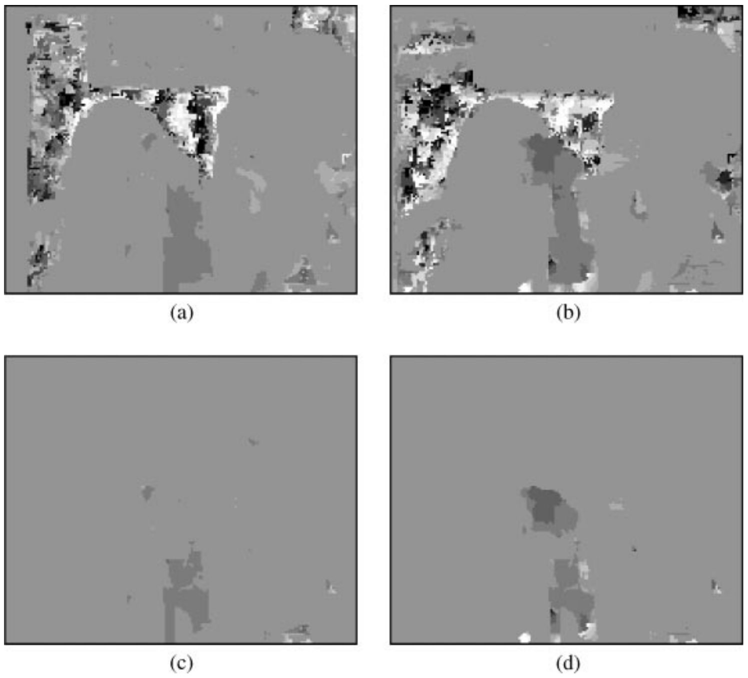
Figure 5. Motion field estimation of the frame #76-77 of the "Mother and Daughter" sequence. (a) x component, (b) y component of velocity vectors, (c) x component and (d) y component filtered with statistical change detection.

This problem is superable with the gradient based method, where the revaluation of the SSD can be achieved for all pixels simultaneously. Starting from the intensity conservation law:

$$I(x,y,t) = I(x - \int_0^t V_x(x,y,t), y - \int_0^t V_y(x,y,t), 0), \quad (4)$$

the gradient constraint approach can be formulated into the following well-known equation:

$$M(x,y) \cdot V(x,y) + b(x,y) = 0, \quad (5)$$

where V is the optical flow field and M and b are:

$$M = \begin{bmatrix} \sum\sum WI_x^2 & \sum\sum WI_xI_y \\ \sum\sum WI_xI_y & \sum\sum WI_y^2 \end{bmatrix} b = \begin{pmatrix} \sum\sum WI_xI_t \\ \sum\sum WI_yI_t \end{pmatrix}.$$

I, with corresponding lower indices, means spatial and temporal derivatives of the image, and W is a Gaussian window for spatial support [22]. This description introduces a constant model for V in a small neighborhood, giving further constraints to solve the problem in the least square sense, as to find the most appropriate vectors V, obviously, we must choose the vector that approximates the constraint equation with the least error.

It is easy to see that the elements of M and b can be directly computed in a parallel way, well suited to our framework and this calculation is required only once per each video frame. The evaluation of a motion vector candidate in each pixel position at time t, carried out by some multiplications and additions with the elements of M and b, foretell that the evaluation can be a part of a motion optimization/segmentation algorithm. This way, motion vector estimation and segmentation can be combined into one model and can be processed by the simultaneous change of V.

The segmentation algorithm itself is very similar to the MRF based method described in the section on stochastic optimization. By the labeling process, during which V is changed, our aim is to get a homogenous vector field that mostly satisfies the motion equation. This is forced by one component of the energy function of the label field. Besides this energy component, which is responsible for the fidelity of motion observations, a second part is necessary to encourage the formation of homogenous regions. This second smoothness term can be calculated as the sum of the Euclidean distance between the velocity vector and its local neighbors. Another case is when the local similarity is measured by a simple penalty function similar to our pervious model:

$U(\omega_s, \omega_r)$ is equal to $-\beta$, if $\omega_s=\omega_r$ and to $+\beta$ otherwise. Then the energy function to be minimized is:

$$E_s(\omega) = |M \cdot V_s(\omega) + b| + \sum_{\{s,r\}\in C} U(V_s(\omega), V_r(\omega)), \quad (6)$$

where ω is the appropriate label field and $V_s(\omega)$ is the corresponding velocity candidate. As we want to optimize a vector field, there might be too many possible candidates and so it would make it impossible to achieve fast convergence. Due to this reason we made two restrictions on the label field:

- For the initial vector field we use the vectors obtained by the first estimation. During the optimization no new values are introduced.
- In the optimization process a label can be changed only to the value of one of its neighbors.

By these restrictions, fast convergence can be reached within cc. 100 iterations by the MMD optimization method. We found several authors applying the same constraints with success under similar conditions [24, 29]. In our example in Figure 6 we show one dense motion field from the "Table Tennis" sequence. Frame #2 has been processed. The first motion field is the initial state obtained with the gradient technique. Only two image frames had been stored in the memory simultaneously to compute time derivative I_t, that is the reason why the estimation can be weak [22]. Utilizing the described optimization process we obtained the motion fields (c, d, e, f) of Figure 6.

Pixel level tracking: motion history

As we will see, accumulated motion information of the recent frames is an important part of our spatio-temporal segmentation model. In many cases the currently measured motion field itself cannot describe the motion very well. With the help of motion history information, temporal uncertainty of estimation can be reduced and long-range information can be accumulated. We track the motion of each point and register if it has stopped or was in motion within a given period of time:

This way we get a motion history field denoted I_{MH}, where V is the corresponding motion field (with components V_x and V_y and the magnitude of M determines the memory-length of the process. In this motion history, areas with greater value mean regions that have been moving longer in the last M frames. Greater M means that the algorithm has longer memory and thus motion transparency is weaker. Figure 7 shows two succeeding motion maps and the corresponding motion history maps of the sequence "Mother and Daughter".

Morphology operators on cell arrays

The implementation of morphology operators on parallel machines is very reasonable [30], since these functions are simple and operate on a close neighborhood of a pixel. Just to hug to the CNN-UM computer model, we refer to [31] where the implementations of many gray-scale and binary morphological operators on the CNN-UM are described. For our purpose we need only binary operators especially for thinning edge maps obtained during the optimization process. Small patch removal is also useful in motion analysis e.g., to mask out moving regions below a given size.

Disocclusion removal in parallel

Vector fields, obtained by any motion field estimation technique, generally suffer from errors of disocclusion. It is a systematic error, since we know that estimating motion from image projections is an ill posed problem. The removal of disocclusion effects needs higher interpretation of motion, at least (observable) background areas should be recognized. If we assume that background regions are separated from objects in the front, then we can give an approximation to the solution of the problem of disocclusion with low-level steps in parallel. Measuring the optical vector for every pixel, the false stripes (disoccluded background areas) can be removed from the estimated motion field in a consecutive series of steps. In our approximation we are able to handle a finite number of directions and magnitude of velocity vectors. The number of iterations of the disocclusion removal algorithm depends on the maximum velocity and the number of directions present

$$I_{MH}(x,y,t+1) = \left\{ \begin{array}{ll} I_{MH}(x - V_x(x,y,t+1), y - V_y(x,y,t+1), t) + 1 & \textit{if} \quad V(x,y,t+1) \neq 0 \ \textit{and} \ I_{MH}(x,y,t) < M \\ I_{MH}(x - V_x(x,y,t+1), y - V_y(x,y,t+1), t) - 1 & \textit{if} \quad V(x,y,t+1) = 0 \ \textit{and} \ I_{MH}(x,y,t) > -M \end{array} \right\} \quad (7)$$

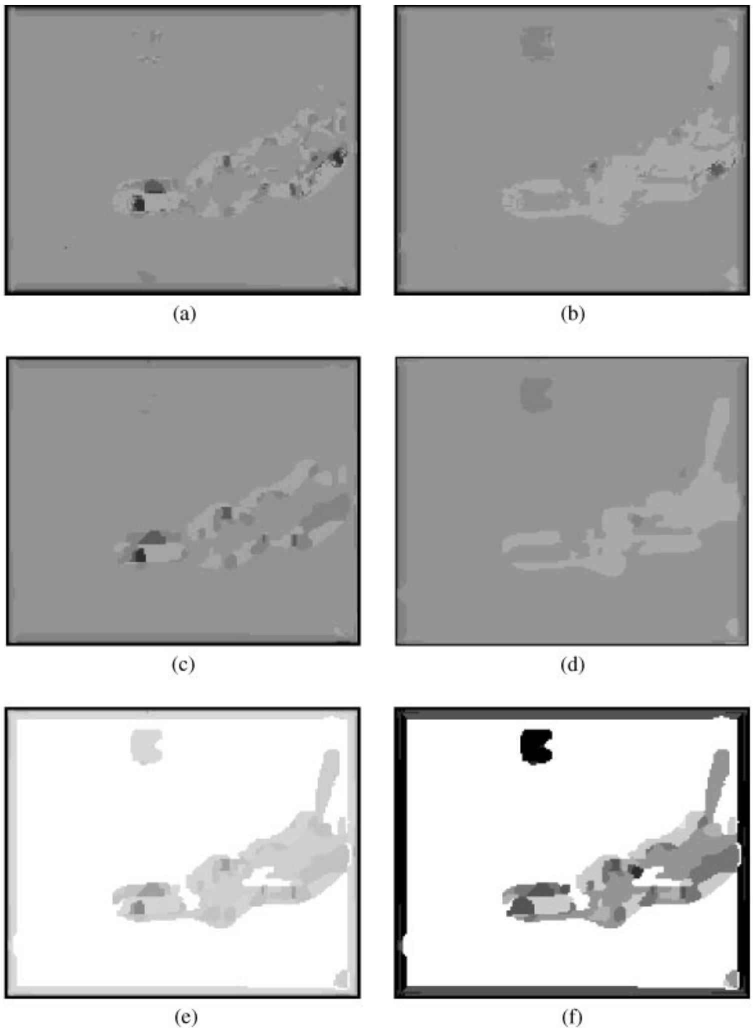

Figure 6. Estimation of the motion displacement field by the gradient method with only 1 memory support without and with stochastic relaxation on "Table Tennis" #2. Estimations and results are given in x and y directions and as the magnitude and direction of velocity vectors. (a) motion in direction x after first estimation; (b) motion in direction y after first estimation; (c) motion in direction x after optimization; (d) motion in direction y after optimization; (e) speed of velocity; (f) direction of velocity.

in the segmented motion field. We should repeat the following steps below for all featuring directions Φ.Φ can take a value from the eight basic orientations: E, W, S, N, NE, NW, SW, SE and can be represented numerically in the cell memories of every pixel. V_Φ equals the maximum speed in direction Φ, the initial state of algorithm is the segmented motion field.

0. Choose Φ as one possible direction present in the motion field and initialize $V\Phi$.
1. Set every pixel to "background" if there is a "background" neighbor on the opposite direction (E–W, S–N, etc.) and has different value than "P" (for the definition of "P" see the next step).
2. Decrease the speed of those pixels that lay in the direction of the pixels that are set to "background" in the previous step (#1). If the speed of them reduces to zero, mark them "P" (as "processed", a new value different from "background" and any other speed.
3. Decrease $V\Phi$. If $V\Phi$ is greater than zero then go to step #1 otherwise go to step 0.

While most of the algorithm is a deterministic parallel labeling process based on local operations and decisions, some serial functions are still needed. Such as those for histogram analysis of the initial segmented motion field to determine the possible directions of motion and estimation of the maximum speed in all directions.

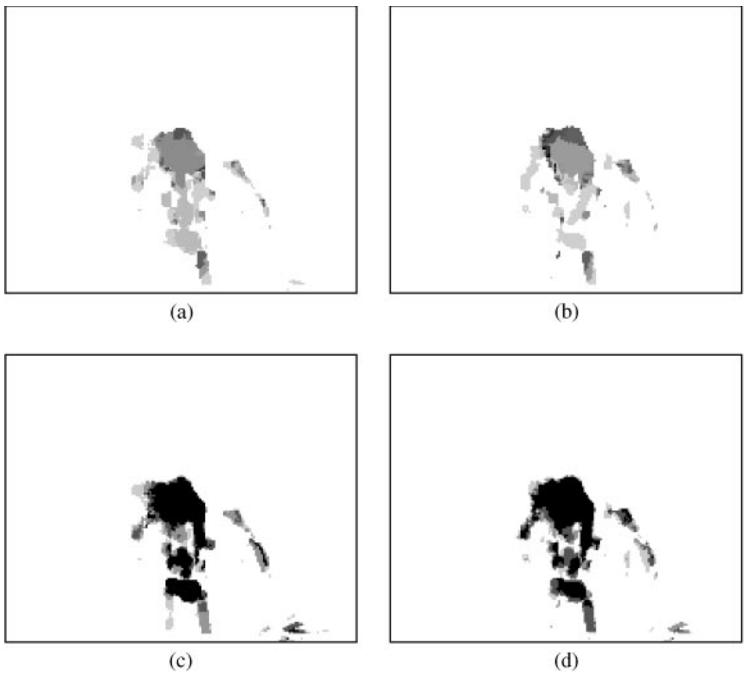

Figure 7. Motion and motion history of the sequence "Mother and Daughter". (a) Speed #81; (b) speed #82; (c) motion history #81; (d) motion history #82.

Edge Optimization for Spatio-Temporal Segmentation and Tracking

The basis of the segmentation algorithm

While many probabilistic approaches use labeling algorithms for spatio-temporal segmentation [29, 32, 33] in our model we employ fast contour based segmentation. In this contour based optimization method there is no need for registering regions or to deal with graph based representations that is not possible in the framework of cell array processors.

Our algorithm is mainly based on three inputs: the oversegmented image (based on gray-scale information), the estimated/segmented optical flow and the motion history information. We found that in many cases the joint utilization of intensity values and the current motion information (motion estimated between two consecutive frames) were not enough to satisfactorily define the objects' contours. On the other hand, the probability that two neighboring image blobs belong to the same object is the higher the more the following requirements are satisfied:

- The two blobs have similar color (or gray-scale intensity value).

(In case of textured areas, texture filters [34] can be applied to colorize these regions.)

- The two blobs have similar velocity.
- The two blobs had similar activities in the recent past.

In our spatio-temporal segmentation process we apply a split & merge algorithm to find coherent image areas based on these three features of neighboring regions.

To reduce the dimensionality of the problem it is possible to replace motion vectors with scalars by a clustering method. In our experiments we simply dropped one component, the segmentation algorithm seemed to be quite robust and gave satisfactory results when we considered only the magnitude of velocity vectors.

The segmentation process

Instead of constructing an explicit energy model as in previous sections we introduce an implicit optimization algorithm where contours are responsible to get an optimal spatio-temporal segmentation of video sequences.

Three edge maps are generated during the algorithm: edges separating areas of different intensity values (E_{in}) edges separating different motion fields (E_m) and edges separating fields of different motion history values (E_{mh}). Edge-fragments of these three maps are different subsets of the spatio-temporal binary edge map E_{segm}, which is a subset of the edge map of the oversegmented image (E_{os}). The three edge maps (E_{in}, E_m, E_{mh}) are weighted and then added to form a unified edge map (E_u) that is thresholded and used to modify the actual E_{segm}. Then the intensity, motion and motion history fields are updated by diffusion inside the contours of the new E_{segm}. If the difference between the new state of the three feature fields and their previous state is too large, some edges may be restored. Then at the next iteration the three different edge maps are measured again and a new unified map is formed, etc.

The optimization is based on the following implicit model:

When the three edge maps are added to form a new unified edge map, the applied threshold criterion is analogous to evaluating a *Dam-potential* between the neighboring segments S_i and S_j:

$$D(S_i,S_j) = \sum_{k=1}^{3} w_k \left| L_k(S_i) - L_k(S_j) \right| \qquad (8)$$

where L_1 = intensity, L_2 = motion (magnitude of the segmented motion field), L_3 = motion history (see section on pixel level tracking), while w_k is a weighting coefficient. If $D(S_i, S_j)$ is above a threshold, then the edge is kept, otherwise deleted at that location.

The reconstruction of edges is a necessary part of the algorithm, because the merging of similar neighboring regions *in one step* can result in the merging of distant areas that have very different values (see Figure 9). Hence we use the following expressions to measure the effects of the edge removal. First, we define the new average feature values over a segment:

$$L_k(S_M) = \sum_{S_i \subseteq S_M} \frac{A_i L_k(S_i)}{A_M}$$

L_k is the kth feature value of the unified region S_M obtained by merging regions S_i, corresponding segment-areas are denoted by A_M and A_i. The change due to the formation of a new region S_M is expressed for each S_i ($S_i \subseteq S_M$) by the difference of the old and the new levels:

$$Q(S_M,S_i) = \sum_{k=2}^{3} |L_k(S_M) - L_k(S_i)|. \qquad (9)$$

If $Q(S_M, S_i)$ is above a predefined value, then the previously eliminated but stored edge-fragments around S_i are reconstructed again. Notice, that no intensity is considered in the edge reconstruction process. It means that regions with different intensity can be merged more easily than with different motion information.
Alternatively, instead of measuring the change for each S_i separately, we can measure the accumulated error over S_M as a volumetric average:

$$Q(S_M) = \frac{1}{A_M} \sum_{k=2}^{3} \sum_{\forall S_i \in S_M} |L_k(S_i)A_i - L_k(S_M)A_i|. \qquad (10)$$

In this case the reconstruction of edges applies for the whole area of S_M, causing the restoration of all previous edges over S_M. This second solution results in larger areas while the other is rather to maintain smaller segments with large contrast. Note that in eqns (9) and (10) averaging over an area means the running of diffusion inside the edge-defined borders.

The *individual* steps of the proposed algorithm are the following (see Figure 8 for illustration, Figure 10 shows some examples):

1. *Segment the input image*, based on intensity observations, possibly to a large number of segments of characteristic closed regions. The resulted segmented image is called over-segmentation, and it gives the finest partitioning that could be achieved in the whole spatio-temporal segmentation process. Good oversegmentation can be generated with the help of anisotropic diffusion or median filtering of the input frame. Both can be implemented in the parallel framework [15, 17].
2. *Produce the edge map of the oversegmented intensity field* (E_{os}) *by an edge-detector* [31]. E_{os} is a binary map showing the more-or-less closed segment-borders of the oversegmented image parts. In the segmentation process the state variable is the current edge map, the binary E_{segm}.

- Starting condition: $E_{segm} = E_{os}$.

3. *Diffuse intensity, motion and motion history fields inside the regions defined by E_{segm} with the help of external edge controlled diffusion. Then make the gray-scale edge maps of these fields, namely E_{in}, E_m*

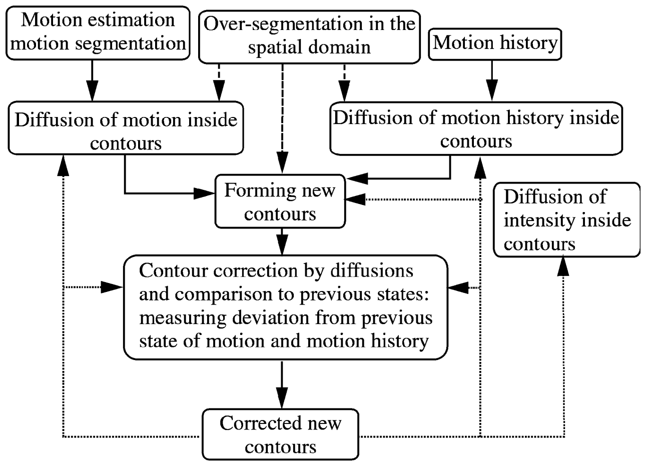
Motion estimation motion segmentation
Over-segmentation in the spatial domain
Motion history
Diffusion of motion inside contours
Diffusion of motion history inside contours
Forming new contours
Diffusion of intensity inside contours
Contour correction by diffusions and comparison to previous states: measuring deviation from previous state of motion and motion history
Corrected new contours

Figure 8. Iterative edge based spatio-temporal segmentation. Dotted lines symbolize feedback of the new contour to the following iterations and to the error computation. The broken lines mean data transfer only for the first iteration.

and E_{mh} respectively. These non-binary (gray-scale) maps contain the edge-strength values between the different diffused areas in the same points where the oversegmented binary edge-segments are in E_{segm}.

External edge controlled diffusion is similar to anisotropic (nonlinear) diffusion (see previous section), however the edge control should act from the external E_{segm} edge map. We found that in some cases the anisotropic diffusion may not smooth feature fields inside strong contours uniformly—in spite of the large

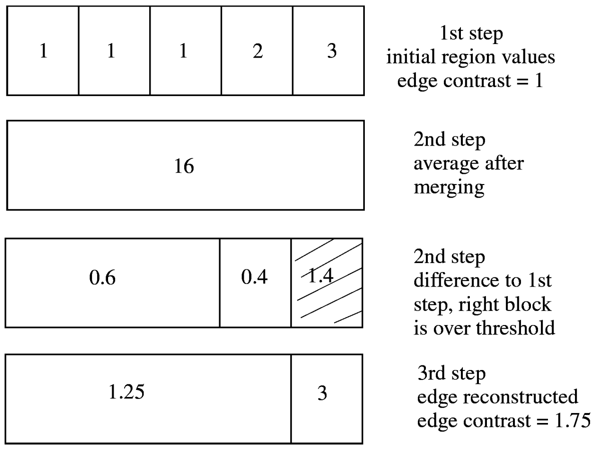

Figure 9. Edge reconstruction in the edge based optimization model. In the first step all five regions are merged but then at the next step the one on the right is separated. The difference between its value and the average of the five blocks was over a threshold of 1.0.

number of iteration steps of the numerical approximation of the formula (eqn 1). Naturally, we do not expect precise averaging like in a region based segmentation method with conventional numerical solutions. Instead, after a given number of steps we stop the diffusion process. Then, as a supplement, a new series of operations begin when we change every pixel's value to its greatest neighbor, except if a pixel has at least two neighbors with corresponding edge points represented in the external edge map E_{segm}. This last condition ensures that we get homogenous areas inside contours and it prevents averaging between regions separated by the "external" edges. This last series of steps is also responsible to get rather continuous contours than leaking edge lines and curves.

Note that while diffusion takes a long computation time on a conventional digital computer (or any SISD architecture), this time is proportional with the diffusion radius on the parallel array. For comparison, on the CNN chip a diffusion process on the whole image of radius r takes a time similar to the one needed to read out r different values from a memory!

4. *Weight and add together the three maps E_{in}, E_m and E_{mh} to form a map E_u.* With the increase of the weight of one component we can control how much the segmentation process should lean on that given type of information. In our experiments we applied about the same weights (w) for the two motion type maps and a significantly smaller weight for the intensity map:
$$w(E_{in}) : w(E_m) : w(E_{mh}) = 0.2 : 1.2 : 1.2.$$
5. *Threshold the superimposed edge-map E_u* and reduce the edges in E_{segm}:
$$E_{segm} := E_{segm} \backslash E_u^{(thresholded)}.$$
Edges of E_{segm} below a threshold in E_u are neglected: closed contours can leak or whole edges can disappear this way.
6. *Approximate the average motion and motion history feature fields by external edge controlled diffusion inside the contours of the modified E_{segm}.* This diffusion is just similar to step 3.
7. *Correct E_{segm} with reconstruction* (E_{rec}). Naturally, the optimal control of the merging of different areas with different intensity, motion and motion history must be a reversible method [1]. Although our cell array framework does not enable us to process a graph based optimization or higher-level understanding, we can still make a feedback to

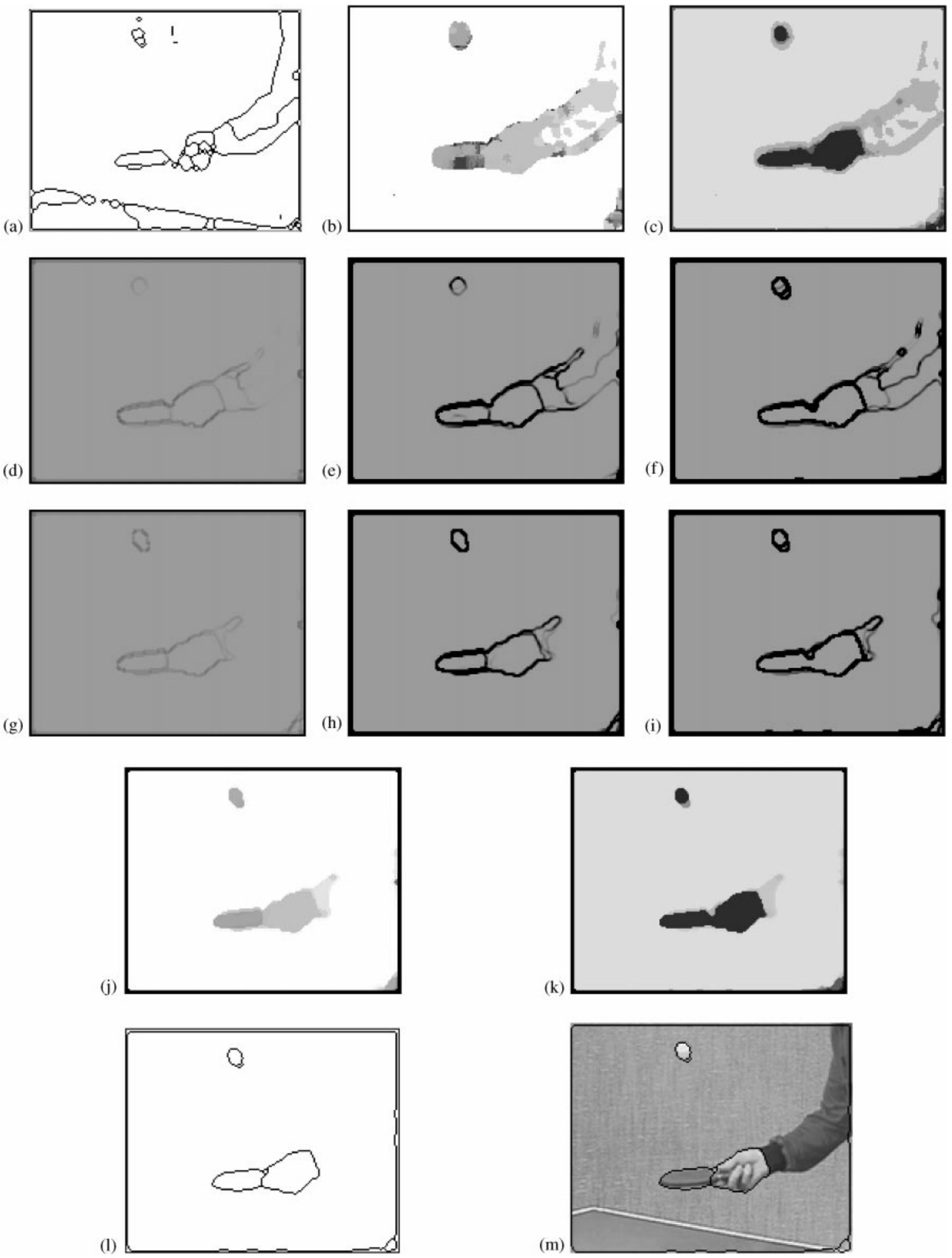

Figure 10. Spatio-temporal segmentation of the sequence "Table Tennis". (a) Oversegmentation obtained by nonlinear diffusion, median filtering and edge detection; (b) magnitude of velocity vectors; (c) motion history; (d) edge map of color; (e) edge map of velocity and (f) edge map of motion history after the 2nd iteration; (g) edge map of color; (h) edge map of velocity; (i) edge map of motion history after the 8th iteration; (j) map of velocity; (k) map of motion history after the 10th iteration; (l) final contours after the 10th iteration, (m) final contours projected onto the input image.

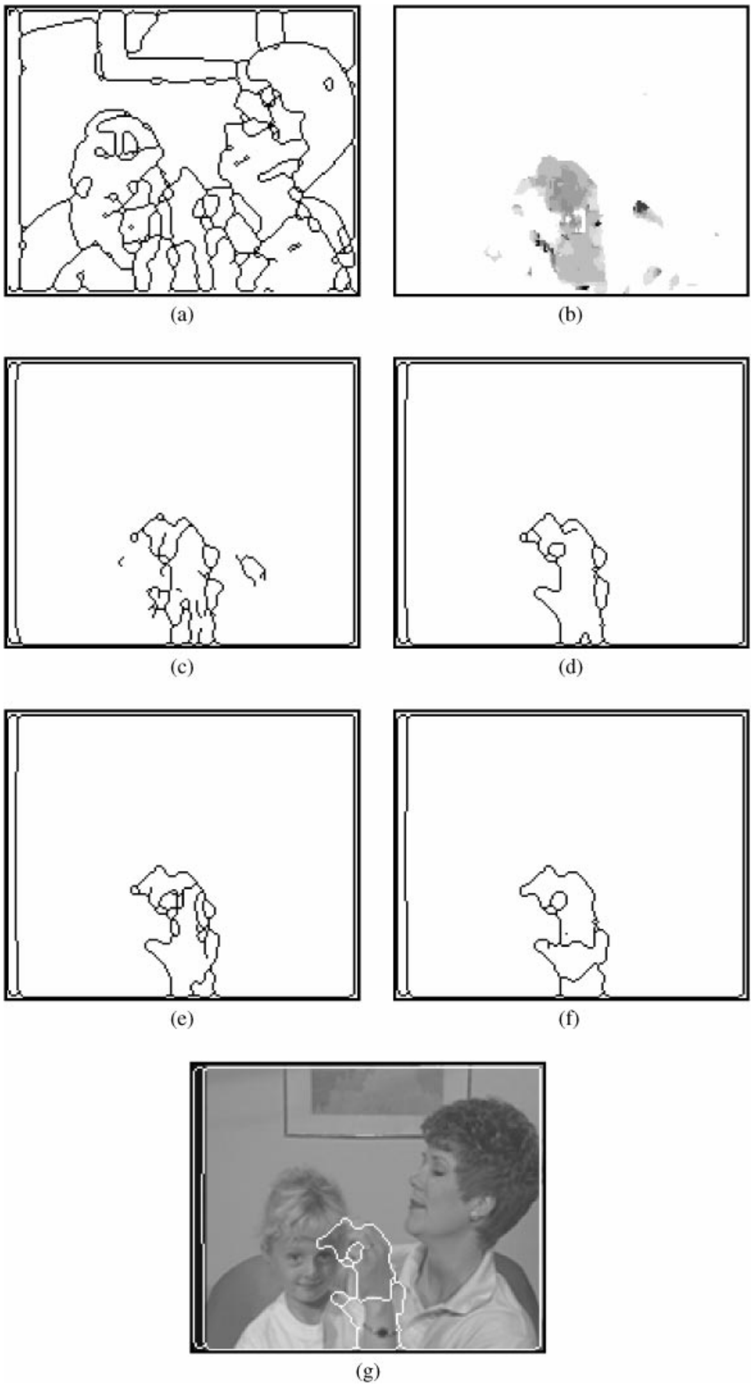

Figure 11. Edge optimization for the spatio-temporal segmentation of "Mother and Daughter". (a) Oversegmented input frame, (b) motion of the current frame, edges of the (c) Ist, (d) 3rd, (e) 5th, (f) 9th, iterations. (g) final edge map (10th iteration) projected onto the input image.

rebuild some lost edges. In every cycle, the change between the current motion fields and the previously segmented motion fields is measured. Over those areas, where the difference of the old and the new features (given by eqns (9) or (10)) is greater than a predefined value, a mask is generated (E_{rec}). Then with the help of this mask we can reconstruct edges from the stored edge map of the previous iteration cycle: $E_{segm} := E_{segm} \cup E_{rec}$. Figure 9 illustrates a typical situation (applying eqn (9)) when a cascade of edges are removed because neighboring areas were similar to each other, but the regions at the two margins had significant differences. One may think that the contour based segmentation is very sensitive for leaks on edges but this feedback can interact and correct the segmentation process.

8. *Cycle controlling*
   - *Decrease edge weights.* In our experiments we decreased edge weights by 0–20%. If this relaxation-factor is small, then edge destruction is slow; otherwise the different regions merge into each other faster.
   - *Go to step 3.*

According to our test results, approximately 10–15 iterations were sufficient to get stable edge contours. Morphology operators may then be used to get thin lines as a final result. The segmented motion history of the last iteration cycle can also be used as the input for calculation of motion history of succeeding frames. The resulted map (E_{segm}) contains the contours of the spatio-temporal objects. This map can be forwarded to a vector based DSP process for further analysis, such as MPEG-4 video transmission or motion analysis applications.

Experimental Results

The following examples are the results of a parallel simulation (with the help of a parallel VLSI programming language (Analogic Macro Code [11]). All steps, except the controlling and global parameter-setting operations, are defined with simple analog low-level operations, such as those in the previous section on elementary functions.

Figures 10 and 11 contain some results of the proposed algorithm. We also show some images of segmentation of motion/motion history at different steps of the iteration cycle (Figure 10 (d),(e),(f),(g),(h), (i)). In the first example ("Table Tennis" sequence) the algorithm marked those parts that had different motions. The upper part of the arm was handled separately from the hand since it had different color and either motion, or motion history was far from being uniform within the object's borders. Figure 10 (j) and (k) shows the average speed and motion history within the detected region's borders after the 10th iteration. Edge maps are also demonstrated at two different iteration levels.

Figure 11 illustrates how edges are being optimized through an iteration cycle of ten steps. The optimization for this image is stable after ten steps. Small moving areas were removed because no spatial content was present at those blobs, however the effect of shadow was only partly removed. Figure 12 is another example of how incomplete motion field is reconstructed by spatio-temporal segmentation.

VLSI Chip Speed and Complexity Estimation

Since spatio-temporal segmentation is a high-complexity task, even our fully parallel solutions require fast hardware implementations. We give computation complexity and running time estimations for the proposed

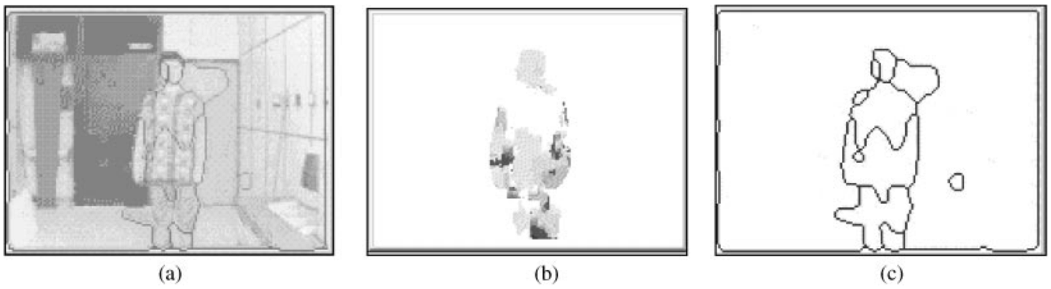

Figure 12. A moving person with detected object borders. In spite of the scanty motion field, the upper part of the body is still detected as a moving area (Some artifacts are also present, e.g., the shadows on the background.)

Table 1. Comparison of execution times on different image processor platform. Image size is 64 × 64, τ is the time constant of the analogue chip [5]

Technology	CNN τ = 200 ns λ = 0.5 μm	Pentium®II. @ 400 MHz & MMX™ λ = 0.25 μm	TMS320C80 @ 40 MHz λ = 0.35 μm	Matrox Genesis with C80 & NOA ASIC @ 50 MHz λ = 0.35 μm
Image Save/Load	90 μs	40 μs	80 μs	80 μs
Arithmetic Operation	500 ns	38 μs	156 μs	47 μs
Logical Operation	100 ns	30 μs	125 μs	40 μs
Conversion from analogue values to binary values/Memory Transfer	200 ns			
Convolution, 3 × 3	2μs	125 μs	383 μs	28 μs
Feedback Convolution (Dynamic IIR Spatial Filter)	5 μs			

algorithms on the CNN-UM parallel computation platform. With the help of our test programs, most of which ran on a software simulator [11], we estimated the number of different steps and the executions-times on the CNN-UM. For VLSI chip speed of basic operations we used data based on empirical tests [5]. Table 1 contains some physical parameters of a CNN chip [4] compared to other computation platforms.

Table 2 gives estimations about how fast our algorithms would run on a fully parallel architecture (like the CNN-UM). The full operation time would satisfy real-time requirements.

The following conditions are supposed in our comparison:

1. Test image size: 64 × 64 (currently available CNN chip is of size 64 × 64).
2. Gray-scale image (8bit/pixel).
3. DSP is used for parameter setting and controlling the segmentation cycles.

On the other hand, there are other physical parameters than speed: area, power consumption, pin-number etc. Table 3 shows the most important technical data of common image processing platforms. This table shows that with a CNN chip we can achieve extremely fast computation at low power consumption on a small area. And what is about the technology? While the CNN-UM technology parameters are restrained: 0.5 μm VLSI technology with very low clock-speed, the computing power of the CNN-UM is still superior regarding the other image-processor architectures.

Table 2. Execution-time estimations for the different algorithms. The table gives typical data for processing a 64 × 64 image. In the columns there are the necessary numbers of steps per iteration and estimated time given in msecs. MDF: estimating motion displacement field with the correlation technique. DR: disocclusion removal.
MRF: Markov Random Field based segmentation. NLDIF: non-linear diffusion. MH: estimating motion history. STS: spatio-temporal segmentation

Number of iterations		Number of instructions per iterations					Time
		Parallel Data-Transfer in Chip	Serial Data-Transfer	Arithmetic Operations	Logical Operations	Convolution Template	ms
MDF	120	11	—	12	—	2	1.4
DR	100	7	—	11	6	1	1.1
MRF	100	7	—	16	6	5	2.4
NLDIF	30	4	—	10	—	8	0.7
MH	120	5	—	3	—	1	0.6
STS	15	22	2	13	2	3	5.6
Σ							11.8

Table 3. Comparing some physical properties of different image processing platforms

Processor	Technology [μm]	Chip area [mm^2]	Pin count	Worst case power consumption [*W*]	Clock speed [MHz]
Pentium®II 400 MHz & MMX™	0.25	230	242	25.0	400
TMS320C80 40 MHz	0.35	~240	305	8.3	40
TMS320C62 × 250 MHz	0.25	n.a.	352	1.9	250
CNN $\tau = 200$ ns cP4000	0.5	90	120	1.2	Digital: 10 Analogue: 1

Conclusion

In our paper we outlined a fully-parallel spatio-temporal segmentation scheme based on small-neighborhood local computations and optimizations. The approach consists of two main modules:

1. Algorithms for image and motion segmentation of spatial and temporal information by optimization.
2. Contour-based split-and-merge spatio-temporal segmentation to utilize the information obtained in the first module.

Both parts can be realized with the same set of simple operations, the need for high-level control is minimized. Basic local instructions are dynamic convolution operators, simple arithmetic steps, logical relations and the simplest nonlinear functions (sigmoid and gradient in a neighborhood). As we have found in the current and previous tests [8, 9], these optimization algorithms are fast and give stable results in a reasonable number of steps

Our aims were to design optimal algorithms for fast implementation on parallel processor arrays. As time complexity estimations in the last section show, our approach can result in real-time operation if implemented in VLSI. The parameters of the latest CNN chip have been applied to estimate the possible implementation of our complex system. This work proves that global (semi-global) optimization of very complex image analysis problems is possible through simple parallel functions interpreted in local neighborhood.

Acknowledgement

Special thanks are due to Francesco Ziliani and Touradj Ebrahimi and the Signal Processing Laboratory (LTS) of EPFL (in Lausanne), and to Tamás Roska in the Analogic Laboratory (MTA SZTAKI, Budapest). This work was supported by the Hungarian Research Fund (OTKA No. T025374) and by the Swiss Research Fund (No. 7UNPJ048236).

References

1. Moscheni, F., Bhattacharjee, S. & Kunt, M. (1998) Spatiotemporal Segmentation Based on Region Merging. *IEEE Transactions on PAMI* **20**: 897–915.
2. Torres, L. & Kunt, M. (eds) (1996) *Video Coding: The Second Generation Approach*. Kluwer Academic Publishers.
3. Toyoda, T., Nitta, Y., Funatsu, E., Miyake, Y., Freeman, W., Ohta, J. & Kyuma, K. (1996) Artificial retina chips as image input interfaces for multimedia systems. *Proceedings of the Optoelectronics and Communications Conference, OECC'96, Chiba, Japan, July*, 1996.
4. Domínguez-Castro, R., Espejo, S., Rodríguez-Vázquez, A., Carmona, A., Földesy, P., Zarándy, Á., Szolgay, P., Szirányi, T. & Roska, T. (1997) A 0.8 μm CMOS Two-Dimensional Programmable Mixed-Signal Focal-Plane Array Processor with On-Chip Binary Imaging and Instructions Storage. *IEEE Journal of Solid-State Circuits* **32**: 1013–1026.
5. Linan, G., Espejo, S., Dominguez-Castro, R., Roca, E. & Rodríguez-Vázquez, A. (1999) A Mixed Signal 64 × 64 CNN Universal Machine Chip *Proceedings of MicroNeuro '99. IEEE, Granada, Spain*, pp. 61–68.
6. Roska, T. & Chua, L.O. (1993) The CNN Universal Machine: An Analogic Array Computer. *IEEE Transactions on Circuits and Systems-II* **40**: 163–173.
7. Szirányi, T., László, K., Czúni, L. & Ziliani, F. (1999) Object oriented motion-segmentation for video-compression in the CNN-UM. *Journal of VLSI Signal Processing* **23**: 479–496.
8. Szirányi, T. & Zerubia, J. (1997) Markov Random Field Image Segmentation using Cellular Neural Network. *IEEE Transactions on Circuits and Systems I* **44**: 86–89.
9. Szirányi, T., Zerubia, J., Czúni, L., Geldreich, D. & Kato, Z. (2000) Image Segmentation Using Markov Random Field Model in Fully Parallel Cellular Network

Architectures, *Real-Time Imaging* **6**: 195–211, doi: 10.1006/rtim.1998.0159.

10. Kato, S., Zerubia, J. & Berthod, M. (1992) Satellite image classification using a modified Metropolis dynamics. *Proceedings of ICASSP*, *San Francisco*.
11. Kozek, T., Zarándy, Á., Zöld, S., Roska, T. & Szolgay, P. (1998) Analogic Macro Code (AMC) — Extended Assembly Language for CNN Computers, Report. MTA SZTAKI, Budapest.
12. Czúni, L., Szirányi, T. & Zerubia, J. (1997) Multigrid MRF Based Picture Segmentation with Cellular Neural Networks. CAIP'97, Kiet. *Proceedings in Lecture Notes in Computer Science* **1296**: 345–352.
13. Alvarez, L., Guichard, F., Lions, P.L. & Morel, J.M. (1993) Axioms and fundamental equations of image processing. *Arch. Rational Mech. Anal.* **123**: 199–257.
14. Perona, P., Shiota., T. & Malik, J. (1992) *Anisotropic Diffusion*, *Geometry Driven Diffusion In Computer Vision*. Kluwer Academic Publishers, pp. 73–92.
15. Roska, T., Szirányi, T. (1995) Classes of Analogic CNN Algorithms and Their Practical Use in Complex Processing. *Proc. IEEE Non-linear Signal and Image Processing*, pp. 767–770.
16. Szirányi, T., Kopilovic, I. & Tóth, B.P. (1998) Anisotropic Diffusion as a Preprocessing Step for Efficient Image Compression, *Proceedings of the 14th ICRP*, *Brisbane*, *IAPR, Australia*, August 16–20, pp. 1565–1567.
17. Rekeczky, Cs., Roska, T. & Ushida, A. (1998) CNN Based Difference-controlled Adaptive Nonlinear Image Filters. *International Journal of Circuit Theory and Applications* (*CTA*) **26**: 375–423.
18. Geman, S. & Geman, D. (1984) Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **6**: 721–741.
19. Davis, L.S. & Rosenfeld, A. (1981) Cooperating Processes for Low-level Vision: A Survey. *Artificial Intelligence* **17**: 245–263.
20. Irani, M. & Anandan, P. (1998) A Unified Approach to Moving Object Detection in 2D and 3D Scenes. *IEEE Transactions on PAMI* **20**: 577–589.
21. Fejes, S. & Davis, L.S. (1998) What can projections of flow fields tell us about the visual motion. *Proceedings of the ICCV*, Bombay, India.
22. Barron, J.L., Fleet, D.J. & Beauchemin, S. (1994) Performance of optical flow techniques. *International Journal of Computer Vision* **12**: 43–77.
23. Laplante, P.A., Stoyenko, A.D. (eds) (1996) *Real-Time Imaging*, *Theory Techniques*, *and Applications*, IEEE Press.
24. Illgner, K. & Müller, F. (1997) Image segmentation using motion estimation. In V. Cappellini (ed), *Time-Varying Image Processing and Moving Object Recognition* **4**: Amsterdam: Elsevier Science, pp. 238–243.
25. Pan, J.N., Shi, Y.Q. & Shu, C.Q. (1998) Correlation-Feedback Technique in Optical Flow Determination. *IEEE Transactions on Image Processing* **7**: 1061–1067.
26. Fleet, D.J. & Langley, K. (1995) Recursive Filters for Optical Flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **17**: 61–67.
27. Shi, B.E., Roska, T. & Chua, L.O. (1998) Estimating Optical Flow with Cellular Neural Networks. *International Journal of Circuit Theory and Applications* **26**: 343–364.
28. Aach, T., Kaup, A. & Mester, R. (1993) Statistical model-based change detection in moving video. *Signal Processing* **31**: 165–180.
29. Bouthemy, P. & Francois, E. (1993) Motion Segmentation and Qualitative Dynamic Scene Analysis from an Image Sequence. *International Journal of Computer Vision* **10**: 157–182.
30. Pitas, I. (ed) (1993) *Parallel Algorithms for Digital Image Processing*, *Computer Vision and Neural Networks*, Wiley Professional Computing.
31. Zarándy, Á, Stoffels, A., Roska, T. & Chua, L.O. (1998) Implementation of Binary and Gray-Scale Mathematical Morphology on the CNN Universal Machine. *IEEE Trans. on Circuits and Systems 1: Fundamental Theory and Applications. (CAS-1)* **45**: 163–168.
32. Gelgon, M. & Bouthemy, P. (1996) A Region-Level Graph Labeling Approach to Motion-Based Segmentation. Technical Report, INRIA.
33. Kato, Z., Pong, T.C. & Lee, J.C.M. (1998) Motion Compensated Color Video Classification Using Markov Random Fields. *Proceedings of ACCV* 1: 738–745.
34. Szirányi, T. & Csapodi, M. (1998) Texture Classification and Segmentation by Cellular Neural Network using *Genetic Learning*, (*CVGIP*) *Computer Vision and Image Understanding* **71**: 255–270.