Real-Time DSP Implementation for MRF-Based Video Motion Detection

Christophe Dumontier, Franck Luthon, and Jean-Pierre Charras

***Abstract*— This paper describes the real-time implementation of a simple and robust motion detection algorithm based on Markov random field (MRF) modeling. MRF-based algorithms often require a significant amount of computations. The intrinsic parallel property of MRF modeling has led most of implementations toward parallel machines and neural networks, but none of these approaches offers an efficient solution for real-world (i.e., industrial) applications. Here, an alternative implementation for the problem at hand is presented yielding a complete, efficient and autonomous real-time system for motion detection. This system is based on a hybrid architecture, associating pipeline modules with one asynchronous module to perform the whole process, from video acquisition to moving object masks visualization. A board prototype is presented and a processing rate of 15 images/s is achieved, showing the validity of the approach.**

***Index Terms*—Digital signal processor (DSP), Markov random field (MRF), motion detection, real-time implementation.**

I. Introduction

MARKOV random field (MRF) modeling is widely used in image processing, e.g., for motion analysis [1]–[3], image restoration [4], and texture analysis [5]. Although the performance of such algorithms is usually very good, their structure is complex and the data flow to process is large. Consequently, the computation cost is high. Since the original paper by Geman and Geman [4], the locality (neighborhood structure) and parallelism of MRF models have been used to speed up the computations. Various real-time implementations of MRF-based algorithms, either on parallel machines or neural networks, have been proposed.

- *Single Instruction Multiple Data* machines (SIMD): this approach fully takes parallel characteristics of the algorithm into account. For example in [6], an MRF-based global region labeling algorithm is implemented on a SIMD array of over 40 000 processing units. The key-points in SIMD implementations are the distribution of data onto the processors and the communication between processors.
- *Multiple Instructions Multiple Data* machines (MIMD): for example in [7], a motion detection and interpretation algorithm is implemented on an MIMD vision machine based on twelve transputers. These machines offer an attractive solution for real-time implementation, but their size, cost and complexity remain too high and limit their use in specific applications.
- *Cellular Neural Networks* perform well by exploiting both parallelism and locality. Moreover, MRF modeling implies the minimization of an energy function which can be solved by electrical networks [8]–[11]. For example, in [12], a Hopfield network is used and simulated to implement an MRF-based optical flow estimation algorithm. Compared to parallel machines, cellular analog networks induce dedicated hardware design (ASIC) which limits the flexibility and adaptivity of implementations. Current CMOS technology constraints only allow the implementation of simple algorithms and restrict the size of computed images (100×100 pixels). Main advantages of such implementations are relaxation convergence speed (faster than 1 μs) and reduced hardware size.

This paper addresses the problem of video motion detection based on MRF modeling with real-time implementation constraints in mind [13]. An alternative solution to parallel machine and neural network approaches is proposed here, based on a split-technology (pipeline/asynchronous) with standard programmable devices (DSP, FPGA, RAM). In Section II, the MRF-based motion detection algorithm is presented. Section III describes the alternative architecture of the machine. Section IV discusses real-time implementation and Section V reports some experimental results.

II. Motion Detection Algorithm

The algorithm described below is derived from the work of Bouthemy et al. [2]. Some definitions are recalled first, then our algorithm is compared to the work in [2].

A. Definitions

Motion detection is a binary labeling problem whose goal is to attribute to each pixel $s = (x, y)$ of image S at time t one of the two following label values

$$l_s = \begin{cases} 1, & \text{if } s \in \text{moving object} \\ 0, & \text{if } s \in \text{static background.} \end{cases}$$

With the hypothesis of static camera and little variation of scene illumination between two consecutive images (I_{t-1} and I_t), motion information at any pixel s is closely related to the temporal change of the intensity function $I_t(s)$. Therefore, observations are defined as

$$o_s = |I_t(s) - I_{t-1}(s)|. \tag{1}$$

Let $l = \{l_s, s \in S\}$ and $o = \{o_s, s \in S\}$ represent one particular realization of label and observation fields L and O,

Manuscript received May 15, 1997; revised November 3, 1998. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Steven D. Blostein.

The authors are with the Signal and Image Laboratory, National Polytechnic Institute, 38031 Grenoble Cedex, France (e-mail: Franck.Luthon@inpg.fr).

Publisher Item Identifier S 1057-7149(99)07548-X.

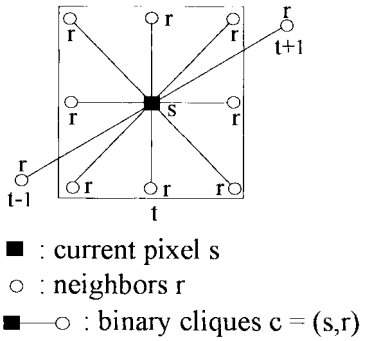

Fig. 1. Spatiotemporal neighborhood and binary cliques.

respectively. The most probable configuration of label field L, given one realization of observation field O, is obtained with the *maximum a posteriori* criterion (MAP). From the equivalence between MRF's and Gibbs distributions [4], we have

$$\Pr[L = l/O = o] \sim e^{-U(o,l)} \tag{2}$$

Maximizing this *a posteriori* probability is equivalent to minimizing an energy function $U(o,l)$ which is the sum of two terms, $U(o,l) = U_p(l) + U_o(l)$:

- the *a priori* model energy U_p is a regularization term given by

$$U_p(l) = \sum_{c \in C} V_c(l_s, l_r) \tag{3}$$

where $c = (s,r)$ denotes any binary clique in the spatiotemporal neighborhood shown in Fig. 1 and C is the set of all cliques. $V_c(l_s, l_r)$ is an elementary potential function associated with each clique c. In order to put homogeneity constraints into the *a priori* model (i.e., to give advantage to configurations where two neighbors have the same label), stepwise potential functions are used:

$$V_c(l_s, l_r) = \begin{cases} -\beta < 0, & \text{if } l_s = l_r \\ +\beta > 0, & \text{if } l_s \neq l_r \end{cases}$$

Three different constant potentials are taken for the three kinds of cliques: β_s for spatial, β_p for past, β_f for future.

- the observation energy U_o represents the link between labels and observations. It is given by the relationship

$$U_o(l) = \frac{1}{2\sigma^2} \sum_{s \in S} [o_s - \Psi(l_s)]^2 \tag{4}$$

where $[o_s - \Psi(l_s)]$ is supposed to be a centered Gaussian noise with variance σ^2 and Ψ is a function that models the observation behavior

$$\Psi(l_s) = \begin{cases} 0, & \text{if } l_s = 0 \\ \mu > 0, & \text{if } l_s = 1. \end{cases}$$

If the pixel s belongs to the static background, there is no temporal change between two consecutive images, so that observation is almost zero. If the pixel belongs to a moving area, observation is supposed to take a positive value close to μ which stands for the average value of nonzero observations. The parameter μ may be estimated on-line as proposed in [2].

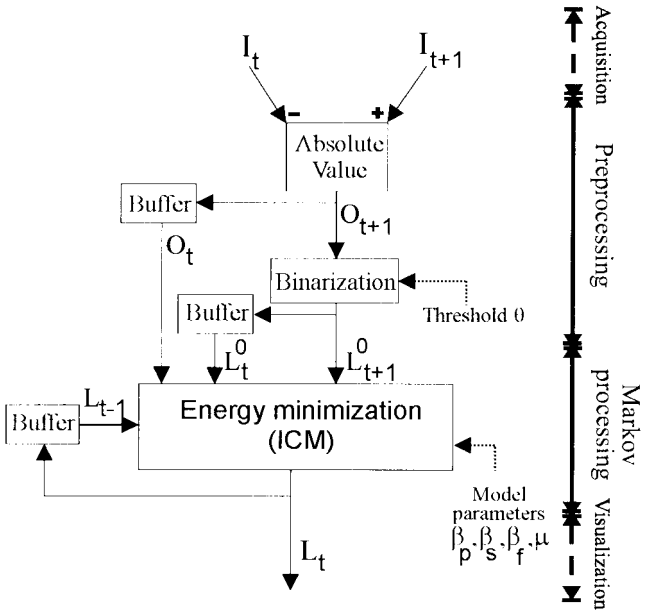

Fig. 2. Block diagram of the motion detection algorithm.

But for the real-time implementation described in this paper, it has been determined manually after experimental tests.

Since stochastic relaxation algorithms are prohibitive for real-time implementation, the deterministic relaxation algorithm iterated conditional modes (ICM) is used to find the minimum of $U(o,l)$ [14].

Fig. 2 summarizes the detection algorithm. At time t, it requires three consecutive frames. Suppose the past label field L_{t-1} has been determined as the result of previous relaxation. The current label field is initialized with a binary field L_t^0 derived from the observation field O_t (comparison to a threshold θ). A coarse estimate L_{t+1}^0 of the future label field is also derived from the binarization of field O_{t+1}. For each site s of the current image, the two label values one and zero are tested and the label which induces the minimum local energy in its spatiotemporal neighborhood is kept. The process iterates on the label field L_t^i until convergence, one iteration i corresponding to the scanning in x and y directions of the whole field at time t.

Note that the algorithm implies a one image delay for obtaining the motion masks at time t, since the frame at time $t+1$ is required.

B. Comparison with Bouthemy's Algorithm [2]

As regards temporal information, both algorithms work on three consecutive frames of the sequence. Both of them respect spatial coherence and take into account temporal discontinuities. Bouthemy's algorithm estimates the final label field in two steps with a *sliding pair* of images, while our algorithm works in one step with a triplet of images and gives advantage to the future with respect to the past by taking $\beta_f > \beta_p$. This helps to eliminate the areas uncovered by motion.

No significant difference has been observed as to the quality of the results produced by the two algorithms, but our algorithm is more efficient if computation complexity is taken into account, as in the following.

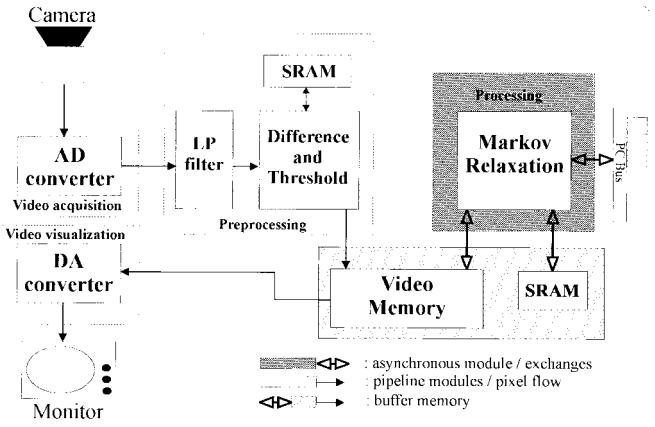

Fig. 3. Block diagram of the board with hybrid (pipeline/asynchronous) architecture.

- The number of parameters required for MRF modeling is lower in our case: four parameters $(\beta_s, \beta_p, \beta_f, \mu)$ instead of five in [2],
- The processing of the temporal direction requires one test only, while in [2] eight different configurations are tested,
- the operations involved in the processing of temporal and spatial dimensions are identical in our case,
- The number of data used during the relaxation process is minor in our algorithm in comparison with the *sliding pair* algorithm in [2].

Note that in our model, temporal neighbors are not literally speaking Markovian because their associated parameters (β_p, β_f) are not equal and their labels are not updated during relaxation. These neighbors must rather be considered as side information for the relaxation process.

III. Hardware Architecture

The algorithm block diagram (Fig. 2) exhibits two processing stages (preprocessing and Markov processing) and two video stream connection stages (acquisition and visualization). Taking into account the video stream connection within the design of the architecture alleviates a lot of problems linked to data control (when acquisition and vizualization are computed with an external board). Moreover, this integration makes the final board autonomous and enables its use in size-restricted applications (embedded applications). For that purpose, a hybrid architecture made of pipeline and asynchronous modules has been designed. Pipeline modules are synchronized by video clock (here, $f_{\text{pixel}} = 7,5$ MHz). The asynchronous module operates at a faster clock (33 MHz).

Fig. 3 gives an overview of this architecture which is composed in practice of two computation stages, two I/O stages and one buffer stage:

Input Stage: Video input stream control. This stage converts standard interlaced video input into digital images of size 512×512 and 8-b pixel depth. It works at standard European video rate, i.e., 25 images/s and uses standard 8-b AD converters.

Preprocessing Stage: This stage computes observations and initial labels. An optional lowpass (LP) filtering is added before this computation in order to improve the quality of initialization in case of very noisy sequences. This stage requires elementary operations only (difference, absolute value, threshold and convolution). These operations are easily implemented on logical components. Field programmable gate arrays (FPGA's) are used because of their flexibility. Initial images are made of two interlaced frames (odd and even) temporally shifted by 20 ms (CCIR standard). To comply with the spatial coherence of our model, only odd frames are processed, so that the image size computed by this stage is reduced to 256×256 pixels.

Markov Processing Stage: This stage performs the energy minimization to obtain the final masks of moving objects. It is implemented on a digital signal processor (DSP) since a careful analysis of the algorithm (cf. Section IV-B3) shows that most of computations involved are similar to convolutions. DSP's are particularly suited for this kind of calculation. Since the energy minimization is an iterative process, data need to be accessed several times during relaxation. Moreover, the number of operations performed and their complexity imply a high computation rate. These characteristics are not compatible with video flow computation. Consequently, this stage works asynchronously, at a rate faster than video rate, on data stored in a buffer memory. The DSP is also in charge of dynamic memory control that is performed under exception processing. This control takes about 8% of the CPU time (one exception appears at the beginning of each line, i.e., each 64 μs).

Buffer Stage: This stage enables asynchronous data accesses and ensures continuity of video data flow. A video RAM component (VRAM) is chosen for this purpose. This video memory is composed of two banks. One is used for input data storage and the other one is used for output mask storage. However, this memory is not fast enough for zero wait-state accesses, so that static memory (SRAM) is also used by the DSP for temporary data storage. Fig. 4 shows the synchronization of computations with respect to video flow.

Output Stage—Video Output Stream Control: This stage converts the resulting frames into a standard interlaced video signal to be visualized on a control monitor. It uses standard 8-b DA converters.

IV. Real-Time Implementation

A. Hardware Configuration

A printed circuit board (PCB) prototype was developed to validate this architecture. The main features of this board are the following:

- standard PC-ISA bus;
- one digital signal processor DSP Motorola 96 002 working at 33 MHz for implementing the deterministic relaxation algorithm;
- one FPGA Xilinx XC4003 working at standard European video rate (25 images/s) on images of size 512×512 for lowpass filtering initial images;
- one FPGA Xilinx XC4005 for differentiating consecutive images, computing the observation field and thresholding

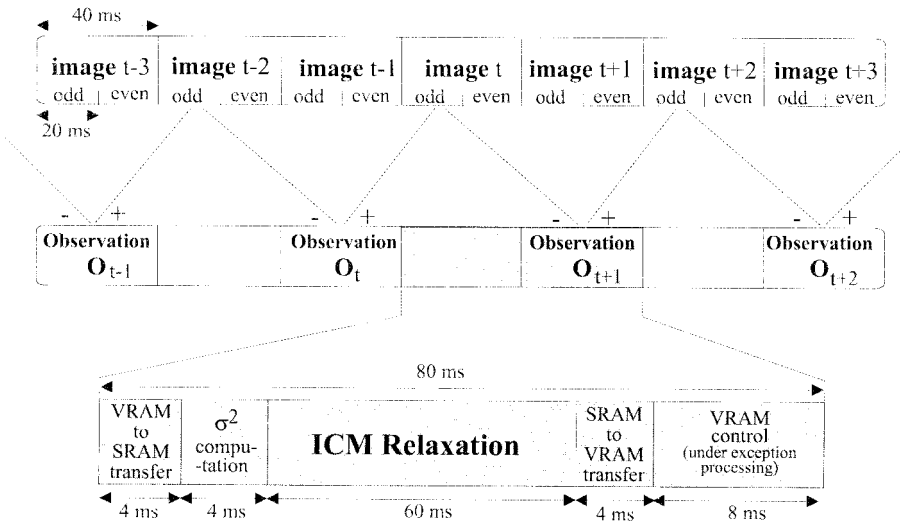

Fig. 4. Synchronization of computations with respect to video flow. (a) Video stream (25 images/s). (b) Preprocessing frames (12 images/s). (c) Details of computations performed to obtain the moving object masks.

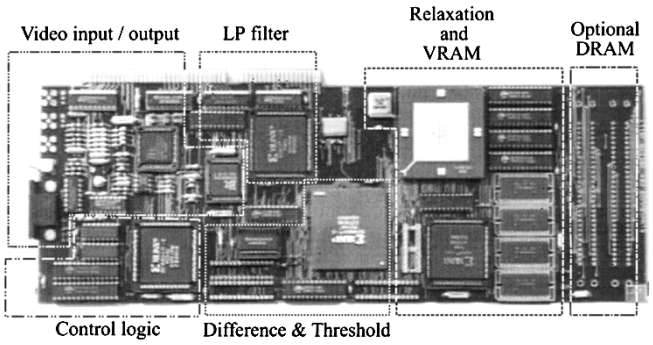

Fig. 5. PC board prototype photo.

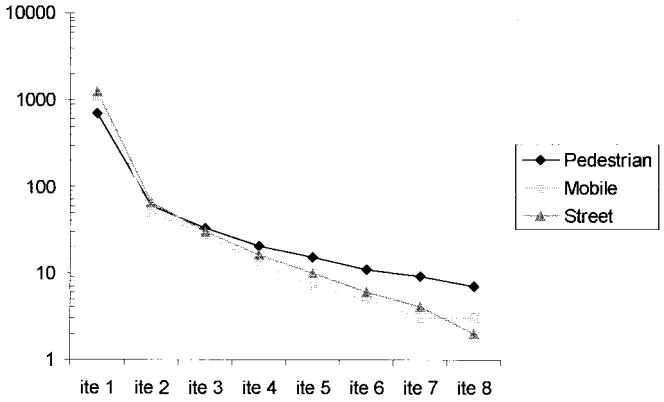

Fig. 6. Evolution of the number of pixels changing of label during the relaxation process. Results are given for three real-world scenes (image size 128 × 128).

observations (initial label field). This stage works on odd frames of size 256 × 256. A local SRAM memory is required for storage of image I_{t+1};

- 32 Kwords of zero wait-state SRAM for temporary data storage;
- two banks, 256 Kbytes each, of triple-port fast page mode VRAM for precomputed field storage (O_{t+1} and L^0_{t+1}) and final label field storage (L_t).
- combined DAC/ADC converters and look-up-tables (Brooktree components).

Fig. 5 shows a photo of the PCB prototype. This implementation includes on a single board all successive stages involved in the processing (from acquisition to visualization).

B. Software Implementation

1) Convergence Criterion: Pixel recursive updating is used for ICM relaxation [14]. Theoretically, the convergence of the ICM algorithm is reached when no more label change occurs after a scan of the whole image. Experience shows that this criterion is too strict and induces superfluous iterations that do not improve the final result (visually). Another criterion focuses on the relative variation of the global energy function between two consecutive iterations ($\frac{\Delta U(o,l)}{U(o,l)} < 0.01\%$). A third criterion stops the process after a fixed number N_i of iterations. This number is determined experimentally. For obvious reasons of computational simplicity, this third criterion is used here. Practical tests show that $N_i = 4$ is sufficient to obtain moving masks of good visual quality. Fig. 6 shows results obtained on three real-world scenes (presented in Section V).

2) Parameter Setting The motion detection algorithm depends on four MRF parameters $\beta_s, \beta_p, \beta_f, \mu$ and one threshold θ. Experimental tests were made on several synthetic and real-world image sequences. Good quality results were obtained with a fixed set of parameters: $\beta_s = 20, \beta_p = 10, \beta_f = 30$ and $\mu = 10$. However, for a specific application, these parameters must be adjusted to optimize the quality of the results. More weight is given to the future by taking $\beta_f > \beta_p$. This helps to deal with motion discontinuities, to take into account any innovation in motion in a faster way, and to better eliminate background areas which are uncovered during motion.

The contribution of each energy term (U_p and U_o) entering into the global energy $U(o,l)$ is fixed to approximately 50% each. Depending on the scene processed, this balance can be

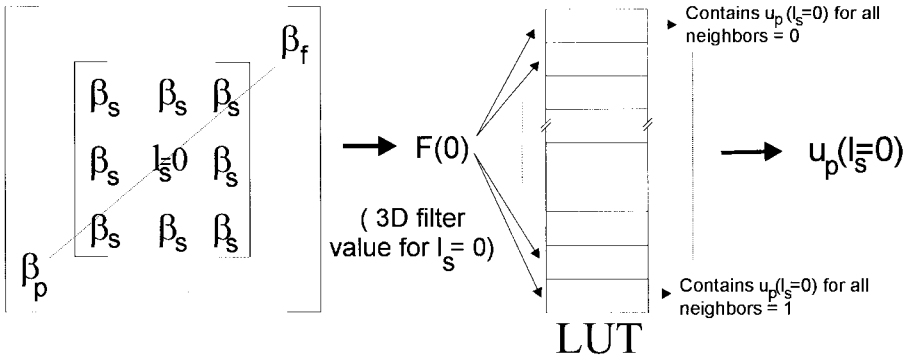

Fig. 7. Computation of $u_p(l_s = 0)$ that is equivalent to 3-D filtering + LUT operation at site s.

modified by introducing a weight parameter λ in the definition of the global energy $U(o,l) = U_p(l) + \lambda\, U_o(l)$.

The binarization operation also requires the definition of a threshold θ. This threshold is estimated by the relationship $\theta = \sigma_n + 2$ where σ_n represents the standard deviation of the noise induced by the acquisition process (camera + digitizer). Depending on the scene, this threshold can take values in the range $4 \leq \theta \leq 7$ (for images coded with 256 gray levels). The value of θ can either be set manually by the user, or computed directly by the processor (online computation of σ_n) and updated by the way of a serial link between the DSP and the FPGA. So, some flexibility is kept in the hardware implementation.

3) Storage and Computation Cost Two important points are the complexity of the calculation and the size of data flow. Since the algorithm used for energy minimization is iterative, data (frames) have to be stored. Five frames are actually required at each time: four binary frames (L_t^0, L_{t+1}^0, L_{t-1}, L_t) and one 8-b frame (O_t).

The contribution of each processing stage to the global computation load has been evaluated. Initialization process (*preprocessing*) accounts for about 10% and *Markov processing*, for 90%. This last process is divided into about 70% for $U_p(l)$ computation, 25% for $U_o(l)$ and 5% for the final choice of label at pixel s. These time evaluations take into account the kind of computations involved and the data access times.

Moreover, a careful analysis of the algorithm [15] exhibits three interesting characteristics for the evaluation of

$$U(o,l) = \sum_{s \in S} u_p(l_s) + u_o(l_s) \tag{5}$$

where $u_p(l_s)$ and $u_o(l_s)$ represent the local energy values at pixel s contributing, respectively, to $U_p(l)$ and $U_o(l)$, as follows.

1) The local model energy $u_p(l_s)$ takes the same absolute value but a different sign for each of the two possible labels ($l_s = 1$ or 0) at a site s

$$u_p(l_s = 1) = -u_p(l_s = 0) \tag{6}.$$

2) the computation of $u_p(l_s)$ over the spatiotemporal neighborhood for one of the two possible labels is equivalent to a convolution with a 3-D-filter followed by a look-up-table (LUT) operation. First, the static label $l_s = 0$ is taken and a temporary value $F(0)$ is computed with the convolution kernel shown in Fig. 7. Then $F(0)$ is processed by the LUT to obtain the actual value of $u_p(l_s = 0)$. Next, the remark 1) is applied to obtain the value of u_p for the moving label ($l_s = 1$): $u_p(l_s = 1) = -u_p(l_s = 0)$;
3) the third remark concerns the evaluation of observation energy $U_o(l)$ given by (4). Theoretically, this evaluation must be done at each pixel s of image S (giving a value $u_o(l_s)$ for each pixel). In practice, $u_o(l_s)$ can only take 256 different values in the range $[0 \ldots \frac{255^2}{2\sigma^2}]$. It is not necessary to compute $u_o(l_s)$ for each pixel and at each iteration of the process, but only to compute once and for all the 256 possible values at the beginning of each relaxation process.

V. Experimental Results and Performance

Three examples of experimental results obtained with the proposed hardware board are given in Figs. 8–10. Sequences were acquired with a standard CCD camera and results are obtained with the fixed set of parameters given in Section IV-B2. Each experiment emphasizes one major property of the algorithm, as follows.

- Fig. 8 shows the noise reduction achieved by MRF relaxation. The scene contains three *walking* pedestrians, a pedestrian *hidden* behind a street lamp and a car appearing on the right of the street (in the last image of the sequence). The initial label field is very noisy (acquisition noise of a standard camera), but after relaxation, only the moving objects are detected.
- Fig. 9 shows the cancellation of uncovered areas. The mobile has a fast rotational motion. In the initial label field, a large uncovered area is visible (i.e., the position of the mobile at time $t-1$). After relaxation, the final label field only exhibits the mobile mask at time t.
- Fig. 10 illustrates the regularizing behavior of the algorithm and the quality of the masks (e.g., for video-surveillance or traffic control application). Note the detection of the pedestrian's shadow on the bonnet of the car (lower left corner of the image), and the fairly good reconstruction of the leg motion.

The main difficulties encountered by the motion detection algorithm concern the following two kinds of situations.

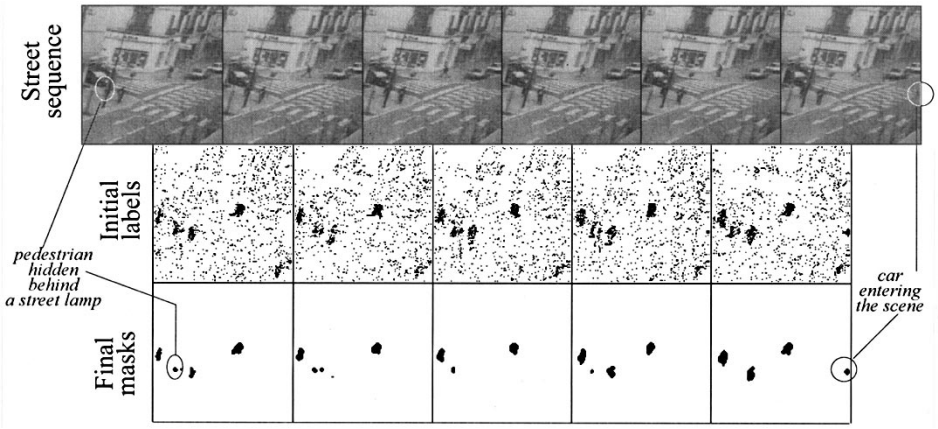

Fig. 8. Noise reduction.

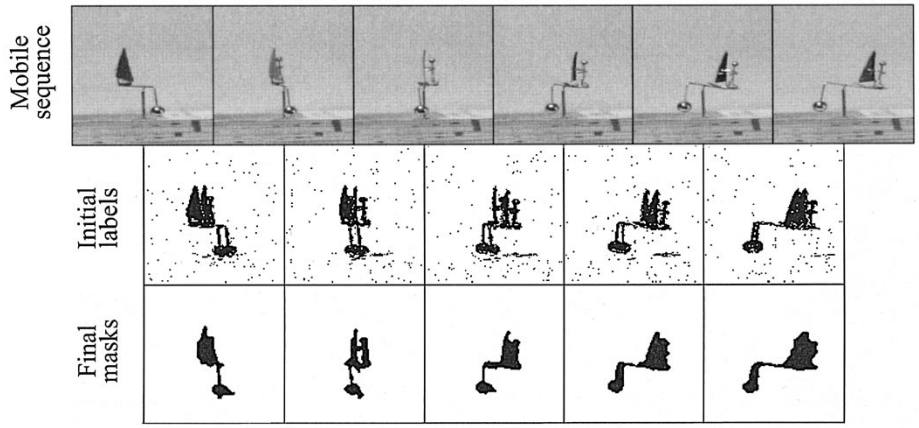

Fig. 9. Cancellation of uncovered areas.

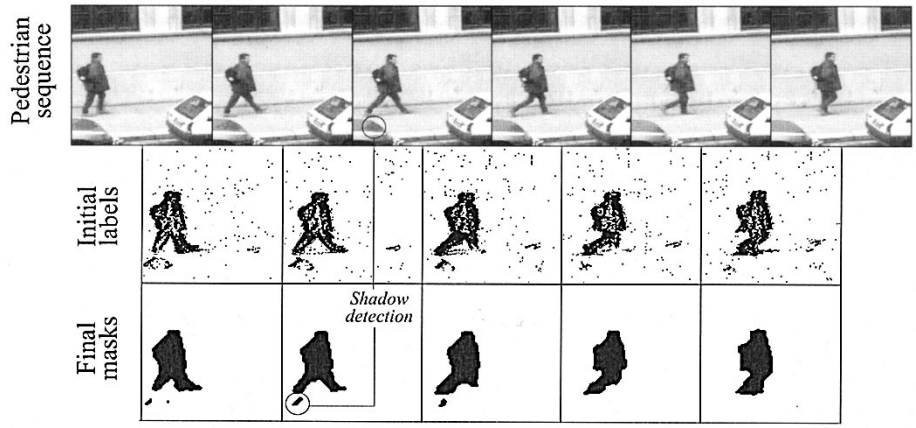

Fig. 10. Mask quality.

- *Very small moving objects*: a too small object is considered as noise in the frame. Specific MRF-based algorithms have been developed for these cases [16].
- *Low speed objects:* there is a lack of information given by the preprocessing stage (thresholded frame differences are zero almost everywhere). Spatiotemporal approaches and multiresolution framework may improve the detection quality for this kind of motion [17].

In practice, a processing rate of *12 to 15 images/s* for images of size 128 × 128 is achieved by our PCB prototype. This corresponds to about half the European standard video rate (25 images/s). Compared to the rate of ten images/s obtained with a parallel implementation of the same algorithm on a SIMD machine with 256 elementary processors [18], the performance achieved by our prototype is satisfying.

The main shortcoming is the limited image size (128 × 128 pixels). This size may be sufficient for many applications (intruder detection for example). Images of size 256 × 256 could be processed by our board, but this would decrease the computation rate by about a factor of four (four images/s).

Another solution consists in using the lowpass filter implemented in the preprocessing stage. With this filter, initial images (256×256 pixels) can be subsampled without aliasing and then, the relaxation stage still works on images of size 128×128. To get final label fields of size 256×256, a simple interpolation (duplication of pixels) is performed. Of course, this procedure induces a slight lowering in the precision of the mask edges.

VI. Conclusion

In this paper, a simple and robust MRF-based motion detection algorithm is presented and a hardware architecture is proposed for its real-time implementation. The algorithm is adapted in order to limit as much as possible the data flow and the computation cost. It practically works on three consecutive images (which implies a one image delay at the output) and integrates contextual spatiotemporal information to give homogeneous masks. The regularizing behavior of the algorithm is shown on several real-world sequences.

A hybrid architecture (pipeline/asynchronous) is proposed for real-time implementation and a PCB prototype is described. This implementation is *autonomous* and integrates the *whole processing* on a single board, from image acquisition to mask visualization. Logical programmable components (FPGA's) and ADC/DAC components are used for pipeline modules and a single digital signal processor (DSP) asynchronously performs the more complex tasks (energy minimization). The link between these modules is implemented by the way of a buffer memory (VRAM). Results are promising (15 images/s on images of size 128×128) and validate the perspective of using MRF algorithms in industrial applications. Indeed, this work shows that MRF algorithms do not necessarily imply a complex, bulky and expensive hardware implementation.

However, the intrinsic parallel property of MRF modeling is not completely exploited by our implementation. The processing rate and the computed image size could be increased by using an up-to-date processor, integrating some parallelism (e.g., programmable video processors like Texas Instrument TMS320C80 with MIMD architecture).

References

[1] J. Konrad and E. Dubois, "Bayesian estimation of motion vector fields," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 14, pp. 910–927, Sept. 1992.

[2] P. Bouthemy and P. Lalande, "Recovery of moving object masks in an image sequence using local spatiotemporal contextual information," *Opt. Eng.*, vol. 32, pp. 1205–1212, June 1993.

[3] D. W. Murray and B. F. Buxton, "Scene segmentation from visual motion using global optimization," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-9, pp. 220–228, Mar. 1987.

[4] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-6, pp. 721–741, Nov. 1984.

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

[6] G. Budzban and J. DeCatrel, "Markov random fields on a SIMD machine for global region labeling," in *Proc. SPIE*, 1991, vol. 1470, pp. 175–182.

[7] A. Bellon, J. P. Derutin, F. Heitz, and Y. Ricquebourg, "Real-time collision avoidance at road crossings on board the Prometheus-ProLab 2 vehicle," *Intell. Veh.*, Oct. 1994.

[8] J. Hopfield and D. Tank, "Neural computation of decisions in optimization problems," *Biol. Cybern.*, vol. 52, pp. 141–152, 1985.

[9] T. Poggio, V. Torre, and C. Koch, "Computational vision and regularization theory," *Nature*, vol. 317, pp. 314–319, Sept. 1985.

[10] J. Hutchinson, C. Koch, and C. Mead, "Computing motion using analog and binary resistive networks," *Computer*, vol. 21, pp. 52–63, Mar. 1988.

[11] F. Luthon and D. Dragomirescu, "A cellular analog network for MRF-based video motion detection," *IEEE Trans. Circuits Syst. I*, vol. 46, pp. 281–293, Feb. 1999.

[12] J. Konrad, M. Zaremba, G. Chan, and M. Gaudreau, "Parallel computation of dense motion fields using a Hopfield network," in *9th Scand. Conf. Image Anal.*, Uppsala, Sweden, June 1995, pp. 609–616.

[13] C. Dumontier, "Etude et mise en oeuvre temps réel d'un algorithme markovien de détection de mouvement par approche markovienne," Ph.D. dissertation, Nat. Polytech. Inst., Grenoble, France, Nov. 1996.

[14] J. Besag, "On the statistical analysis of dirty pictures," *J. R. Stat. Soc. B*, pp. 259–302, 1986.

[15] C. Dumontier, F. Luthon, and J. P. Charras, "Real time implementation of an MRF-based motion detection algorithm on a DSP board," in *7th IEEE Digital Signal Processing Workshop*, Loen, Norway, Sept. 1996, pp. 193–187.

[16] C. Hennebert, V. Rebuffel, and P. Bouthemy, "Detection of small and slow moving objects observed by a mobile camera," in *9th Scand. Conf. Image Analysis*, Uppsala, Sweden, June 1995, pp. 155–162.

[17] C. Bouman and M. Shapiro, "A multiscale random field model for Bayesian image segmentation," *IEEE Trans. Image Processing*, vol. 3, pp. 162–177, Mar. 1994.

[18] A. Caplier, F. Luthon, and C. Dumontier, "Real-time implementations of an MRF-based motion detection algorithm," *Real-Time-Imag., Spec. Issue Real-Time Motion Anal.*, vol. 4, pp. 41–54, Feb. 1998.

Christophe Dumontier was born in Pertuis, France, in 1967. He worked on real-time implementation of motion analysis algorithms for three years and received the Electronics Dr.Eng. degree in 1996 from the Grenoble Polytechnic National Institute, Grenoble, France.

Currently, he is an R&D Engineer at Intermec STC, Toulouse, France, since March 1998. He works on a European Research Project (ESPRIT) in the field of real-time image processing applied to 2-D bar-code reader domain.

Franck Luthon was born in Rambouillet, France, in 1963. He received the Electronics Eng. degree and the Dr.Eng. degree, both from the Grenoble Polytechnic National Institute, Grenoble, France, in 1985 and 1988, respectively.

He has been an Assistant Professor at Grenoble Polytechnic National Institute since October 1990, where he is currently with the Signal and Image Laboratory. His research interests are in the area of motion analysis and spatio-temporal segmentation in image sequences, with particular emphasis on real-time implementations. He has been an Associate Editor for the French signal processing journal *Traitement du Signal* since 1995.

Jean-Pierre Charras was born in Privas, France, in 1947. He received the Electronics Eng. degree and the Dr.Eng. degree from the Grenoble Polytechnic National Institute, Grenoble, France, in 1970 and 1980, respectively.

He has been an Assistant Professor at Joseph Fourier University, Grenoble, since October 1972. He is currently with the Signal and Image Laboratory. His research interests are in the area of electronics design for video acquisition and processing.