RETIMAC: REal-TIme Motion Analysis Chip

Paolo Nesi, *Member, IEEE*, Fabrizio Innocenti, and Paolo Pezzati

***Abstract*—Motion estimation is relevant for applications of both motion-compensated image sequence processing and dynamic scene analysis of computer vision. Different approaches and solutions have been proposed for these two applicative fields. In some cases, parallel architectures and dedicated chips for motion estimation in real-time have been developed. In this paper, a low-cost REal-TIme Motion Analysis Chip, RETIMAC, is presented, which is suitable for dynamic scene analysis in computer vision applications. This chip is capable of estimating optical flow fields in real-time, and has been especially developed for project OFCOMP (Optical Flow for COunting Moving People) DIM 45 ESPRIT III MEPI. It can be profitably used also for autonomous navigation, tracking, surveillance, counting moving objects, measuring velocity, etc., and for several computer vision applications which require as a first processing step the estimation of the apparent velocity of each pixel in the image plane (e.g., optical flow, velocity field). RETIMAC implements a gradient-based solution which has been demonstrated to be more reliable and precise with respect to several solutions proposed in the literature.**

***Index Terms*—ASIC, computer vision, motion estimation, optical flow estimation, real-time implementation.**

I. Introduction

IN IMAGE-BASED systems, the moving objects in the observed scene project their three-dimensional (3-D) velocity on the image plane (i.e., apparent velocity). This is usually called "velocity field" or "motion field" [1]–[3]. Most of the techniques adopted for evaluating velocity fields consider changes in the image brightness features/areas. For this reason, the fields obtained are normally called "optical flow" or "image flow" field. Generally, optical flow field differs from the velocity field, which is a pure geometric concept. The latter is equal to the optical flow field only under specific physical conditions of illumination, reflectance, moving object texture, etc. [2]. As stated many times, the motion estimation is in general an ill-posed problem [4]; its feasibility depends on the additional constraints which are usually added to allow the definition of specific solutions [5]. For these reasons, the optical flow precision with respect to the true velocity field also depends on the estimation technique adopted [6]. On the other hand, the estimations of an approximated velocity field can be enough for many applications.

Optical flow estimation is very useful in two important areas: *motion-compensated image sequence processing* [7], [8], and *dynamic scene analysis* of computer vision [3], [9]. In particular, examples of the first area are coding, noise reduction, and image interpolation; while examples of the second area are robot vision, scene monitoring, object tracking, moving object recognition, 3-D motion estimation, 3-D structure reconstruction of moving scene/objects, model-based image sequence coding, autonomous navigation, obstacle avoidance, etc. For motion-compensated image sequence processing, the precision of motion estimation and the difference between the optical flow estimated and the true velocity field are not very important, since the motion estimated is used for coding changes produced on the image plane. For many applications of dynamic scene analysis, the precision of motion estimation is really relevant since small errors can produce severe errors in the final result; for example, in the cases of: 3-D shape reconstruction of moving objects, 3-D motion estimation and understanding, object tracking in 3-D, obstacle avoidance, and where it is important to obtain a true velocity field.

For these reasons, *motion-compensated image sequence processing* and *dynamic scene analysis* applications have adopted different estimation techniques and architectures. Most of the above-mentioned applications of both categories need to consume optical flow fields in real-time. Motion estimation must be performed in real-time, especially for image sequence coding and robot vision. The implementation of parallel architectures and/or specific VLSI chips are techniques to achieve this goal. In the literature, there exist several motion estimation architectures for image sequence coding in real-time, e.g., [10]–[13], while only few are capable of producing suitable results for dynamic scene analysis in real-time, e.g., [14]–[16].

In this paper, the mathematical bases and the implementation of RETIMAC (REal-TIme Motion Analysis) chip for the real-time production of optical flow fields suitable for dynamic scene analysis are presented. This paper is organized as follows. Before presenting the approach selected, a discussion about the motion estimation techniques and related parallel architectures is reported in Section II. Section III reports the basic notions for the optical flow estimation used in RETIMAC following the approach proposed in [6] and [14] by one of the authors. An overview of the environment in which RETIMAC is usually operating is reported in Section IV. In Section V, the functional description of RETIMAC is provided; in Section V-A, the datapath and the final performance of the chip are, respectively, discussed. Conclusions are drawn in Section VI.

II. Motion Estimation Approaches

This section reports the rationale that led us to build the RETIMAC (REal-TIme Motion Analysis) chip. The main requirement was the implementation of *a low-cost chip for optical flow estimation in real-time, suitable for dynamic scene*

Manuscript received June 24, 1996; revised February 13, 1997. This work was supported by SED s.r.l. of Certaldo, Florence, Italy, the main contractor of Project OFCOMP DIM45 ESPRIT III MEPI. This paper was recommended by Associate Editor O. K. Ersoy.

P. Nesi is with the Department of Systems and Informatics, University of Florence, Italy (e-mail: nesi@ingfil.ing.unifi.it, www: http://www.dsi.unifi.it/~nesi).

F. Innocenti and P. Pezzati are with the Hi-Tech Agency CESVIT s.p.a., Microelectronics Center, Florence, Italy.

Publisher Item Identifier S 1057-7130(98)00776-9.

analysis applications. The first reference application was the counting of passing people getting on/off public buses, i.e., the estimation and recognition of nonrigid moving object behavior [16], and optical flow estimations and spatio-temporal analysis of these fields. The implementation of the RETIMAC chip has been considered as the way to reduce costs and dimensions of an expensive DSP-based embedded system reported in [16]. This was built for the same purpose and, in general, for computer vision applications based on optical flow field estimation.

Therefore, according to the applicative context, such a chip had to provide the following.

1) Precise optical flow vectors suitable for computer vision applications such as obstacle avoidance, motion understanding, nonrigid object tracking, etc., where "precise" means at least 0.1 pixels by grabbing the scene on 25×25 image (higher resolutions can be obtained for compositions). This means that, theoretically, 1 pixel of resolution could be enough for extracting velocity vectors on images of 250×250 pixels of the same scene.
2) Velocity vectors having components less than ± 1.5 pixels per frame on a 25×25 image. This means that with a 250×250 image, a range of ± 15 pixels is needed.
3) Optical flow fields with a regular grid of 25×25 vectors per image, thus producing 625 velocity vectors per image. These can be estimated on a 25×25 or bigger image. In the first case, a dense optical flow is obtained, where "dense" means a velocity vector per 1 pixel in the whole image.
4) At least 10 optical flow fields per second, thus 6250 velocity vectors per second.
5) A low-cost architecture having, at most, 70 000 transistors and a reduced number of I/O's, strongly independent of other external chips for preparing input data, etc.

Moreover, the algorithm selected for optical flow estimation must be robust with respect to the major drawbacks of optical flow estimation techniques—discontinuities and the problem of aperture [5], [17].

In the literature, four main approaches for motion estimation in a regular grid of the image can be identified: spatiotemporal filtering-based approaches (FBA's), block-matching approaches (BMA's), pel-recursive approaches (PRA's), and gradient-based approaches (GBA's). Please note that since our goal was to estimate regular optical flow fields, then feature-based motion estimation techniques (e.g., corner tracking, line tracking, etc.) have been neglected.

A comparison among the above approaches is really difficult since each of them has been developed for different purposes and produces different results. For the same reasons, also from the computational point of view, the approaches can be compared with difficulty since the effort needed for producing optical flow fields in real-time is strongly influenced by the approach selected and, thus, by the estimation algorithm, data dependency, type of operations, etc., and by algorithm implementation on sequential and/or parallel architectures. Therefore, in the following, the approaches are reviewed in the light of the dynamic scene analysis application field, considering their potentiality about the computational effort needed, estimation precision, and reliability. Additional detailed reviews can be found in [3], [6], [7], [9], [18], [19].

A. Spatiotemporal Filtering-Based Approaches (FBA's)

The FBA's treat the complete image sequence with a set of filters in the temporal and/or the spatial frequency domain. Filtering is tuned in frequency/time and space in order to detect the components of the motion. This theory is derived from the analysis of vision systems in animals. This class of methods can be divided in two fundamental approaches: the spatiotemporal and the spatiotemporal frequency approach.

The first approach uses a set of 3-D bandpass filters tuned on the same spatial frequency, but with different spatial orientations and temporal frequencies. Filters such as 3-D Gaussian convolution are normally used. Van Santen and Sperling have presented an approach that uses spatial and temporal filters, expanding Reichardt's model [20]. Adelson and Bergen [21] have proposed an energy model in which the motion is estimated by integrating in time the output of a set of filters oriented in space–time and tuned in spatial–frequency. Heeger has used a collection of spatiotemporal bandpass Gabor-energy filters, which can be seen as a pair of 3-D Gaussian filters [22].

The second class of solutions works on the frequency domain. These methods are based on the fact that a translation and/or rotation of an object in the frequency domain can easily be detected, e.g., the translation is seen in the frequency domain as a shift, and the rotation is considered as a local translation. Watson and Ahumada have used a set of spatiofrequency tuned filters to extract the velocity field considering only translational movements [23]. Another point of view for this approach can be found in Jacobson and Wechler [24].

Typically, with FBA's, small displacements are estimated, thus these are unsuitable for motion-compensated image sequence processing. For FBA's, the estimation precision is critical since it depends on the dimension of the filters and on the mathematical approach for detecting peaks. The range depends on the dimension of the area in which the peaks are searched. To obtain the required precision, the optical flow field grid has to be much bigger than 1; a reasonable number can be 10.

Algorithms based on FBA's do not operate on local information, but use the global information contained in the whole image sequence. Although this approach is computationally complex, it fits well a pyramidal architecture, where a multiresolution set of images from the scene under observation can be easily obtained and analyzed [22], [25], [26]. At each level of the pyramid, the information related to one spatial–frequency band is extracted. Unfortunately, the cost of pyramidal architectures makes this approach somewhat impractical for hardware implementation.

The typical asymptotical complexity can be an $O(M^2B^3F)$, where M is the optical flow field dimension in pixels (equal to the image dimension if the grid is equal to 1), B is the

dimension of the 3-D filter (typically from 7 to 15), F the number of filters applied (typically 4–12, but at least 24 for the required precision). Considering $M = 25$, FBA's need from 51.4 to 506 MOPS (millions of operations per second) to evaluate 10 optical fields per second. Please note that most of these operations must be performed by considering the fractional part. In this case, to better evaluate the algorithm complexity, a scale factor from 10 to 20 should be considered with respect to approaches presenting only operations on small integers, but this depends on the specific hardware adopted.

The above number of operations per second, as those evaluated in the following, represents obviously a coarse estimation of the effort needed to obtain the results. These estimations can be useful for evaluating the scale of the computational effort needed.

For our purpose and for many computer vision applications, this approach is unsuitable, since precise estimations can be obtained only at the expense of a huge complexity.

B. Block-Matching Approaches (BMA's)

BMA's divide the image in blocks of sizes $B \times B$ and produce a velocity vector for each block. The pattern of the image block around the point under tracking at the previous time instant is used as the reference pattern for searching the displacement and, thus, the velocity of pattern center [7]. Each motion vector can be reliably obtained only on the basis of the large block matching (typically, 16×16, 32×32). Too small a block size can lead to instability due to the ill-posedness of the motion estimation problem [3], [5]. The block-matching technique is also sensitive with respect to noise and discontinuities. This problem is strongly attenuated by augmenting the dimensions of the reference block. Large blocks can produce wrong estimations when a block presents different motions due to deformations, rotations, presence of object boundaries, presence of more than one moving object with different motions, etc. BMA's are usually quite imprecise since most of them produce results with a resolution equal to 1 pixel. A better resolution can be obtained by interpolating results at the expense of computational cost, thus obtaining a precision of 0.5 or 0.25 pixels, or by increasing the number of pixels of the scene since BMA's are very suitable for estimating large displacements.

BMA's can be classified in two main categories: full search and nonfull search algorithms. The full search BMA's, FBMA's, are optimal in the sense that they are capable of finding the optimal solution among the possible ones. FBMA's search the best match of the $B \times B$ reference block in all points of the $S \times S$ search area; thus, the computational cost is an $O(M^2B^2S^2)$, where typically $S \geq B$. This complexity is addressed by evaluating a velocity vector by executing differences between pixel values and other operations in parallel for all/many pixels of the block. The regularity of the algorithm and the simplicity of mathematical operations have allowed the implementation of several VLSI architectures. BMA's have been used for many applications in real-time video coding systems such as H.261 standard for videophone and videoconferencing, MPEG for video communication and storage, high definition TV, etc. [27]. According to these standards, many architectures [28]–[32], and hardware implementations [10]–[12], [33]–[36] have been proposed. These hardware implementations are based on pipelined or systolic architectures since they have regular data access. Some of these architectures include an interpolator for increasing the estimation resolution.

Non-FBMA's (NFBMA's) reduce the computational cost by decreasing the number of candidate blocks during matching, accepting suboptimal solutions. Among these algorithms, there are: the Hierarchical BMA's (HBMA's); the two-dimensional logarithmic search-based algorithms; the three-steps search algorithms; and the conjugate direction search algorithms, etc. [27], [37]–[41]. The adoption of HBMA allows us to reduce some problems of classical BMA, such as those produced by the presence of blocks with more than one moving object; but the problems related to estimation precision have not been solved yet at low cost.

As regards parallel implementations for computer vision, a matching-based approach has been utilized in [42] as a first step for defining an algorithm for motion estimation on a Connection Machine in close-to-real-time. In [43], the matching approach has been implemented on a pyramidal architecture. The process starts at the coarsest level where the displacement components are shorter than one pixel, and then it passes to a finer level by using a matching-based algorithm. This architecture can avoid the convergence to local minima.

According to our requirements, to have a 25×25 velocity vector with 0.1 pixels of resolution and a range of ± 1.5 pixels, the BMA's could be theoretically applied on a 125×125 image (considering a half-pixel of resolution) and S must be at least 15 pixels (increasing the image dimensions implies increasing the input bandwidth and frequently the computational complexity). Reliable matchings are obtained only when $B \geq 16$. Then considering $M = 25$, for estimating 10 optical flow fields per second (three operations per pixel), 1080 MOPS are needed. BMA's are strongly regular in data, the operations to be performed are very simple (only additions, subtractions and comparisons) and, thus, they are highly parallelizable, as many times demonstrated. Usually, in BMA parallel implementations, the flow of the input data (images) has to be organized (especially for systolic architectures); while in some pipelined architectures there is a module for generating addresses for reading images from memory.

As a conclusion, BMA's could be used as a basis for dynamic scene analysis applications by increasing their estimation precision and robustness. To this end, several techniques have been proposed in the literature, but they are computationally very expensive: postregularization, hierarchical searching, prediction/correction, nondense optical flow fields estimation, etc. (see [3], [13], [42], [44]).

C. Pel-Recursive Approaches (PRA's)

PRA's have been proposed for the first time by Netravali and Robbins [45]. PRA's are based on an iterative process for finding the minimum of a frame difference function. The iterative process is a descendent technique in which

the step is usually constant. This approach is capable of estimating dense optical flow fields. PRA's are more suitable for estimating small displacements; this makes PRA's more suitable for dynamic scene analysis rather than for motion-compensated image sequence processing, since PRA partially solves the problems of BMA's when different movements are included in the same block. On the other hand, subpixel resolution is obtained by interpolating the estimation of the image brightness and of its gradient. In PRA's, the estimation of velocity vectors with 1/10 pixel of resolution needs several iterations (depending on the dimension of the step). Therefore, to obtain precise results, it is mandatory to evaluate precise interpolations which are usually computationally intensive. Otherwise, in order to improve precision, the image dimensions can be increased, increasing in this way the number of iterations if the step of the iterative process is constant. Moreover, large displacements to be measured also increase the probability of finding a local minima between the starting point and the optimal solution.

PRA's are very sensitive to noise which usually produces local minima in which the descendent technique can erroneously converge. For these problems, the PRA has been considered to be not reliable enough for estimating optical flow fields for computer vision.

From the computational point of view, the dominant operation is the interpolation; therefore, PRA asymptotical complexity is an $O(M^2 I_t P I_n)$, where: M denotes the dimensions of the image in pixels, I_t the number of iterations, P the number of pixel values on the basis of which the gradient of the image brightness is estimated (typically 4), I_n the number of operations for interpolating the image brightness (typically 12). By considering $M = 25$, and $I_t = 15$, the classical PRA algorithm can need approximately 4.5 MOPS. Most of these operations must be performed by considering the fractional part. The adoption of a step based on a 1/16, 1/8, etc., of pixel is a way to reduce the computational effort.

PRA's present an intricate data dependency graph to be implemented by means of systolic architectures; for this reason, they have been rarely considered for defining VLSI chips and parallel architectures for real-time motion estimation—e.g., Kim and Lee in [46] proposed a pipelined architecture implementing the iterative process. PRA was also extended to regions producing the block recursive algorithm (BRA) [45], for improving its robustness with respect to noise. In this case, the interpolation is made by using the information coming from the image block. A more complex approach has been used for defining a linear array of processors by Frimout *et al.* [47].

Region Recursive Approaches (RRA's) are a sort of BRA in which the motion is estimated for regions which are identified after a phase of segmentation [48]. It is very difficult to model the motion of a region with a unique velocity vector; thus, the motion is modeled in terms of invariants: translational, rotational, and divergence components. This fact makes the computational complexity higher than that of classical BMA's, but increases the precision of the estimated motion in the case of rigid objects.

As a result, PRA's are very sensitive to noise and local minima to be suitable used for dynamic scene analysis. BRA's and RRA's architectures are more reliable and complex than classical PRA's to be used for building a low-cost chip for estimating optical flow fields in real-time.

D. Gradient-Based Approaches (GBA's)

GBA's provide a solution to motion estimation from the observation of brightness changes in the image plane, thus leading to motion estimation of image brightness features [2], [6], [17], [49], [50]. GBA's are very suitable for producing dense optical flow fields and can be regarded as the generalization of PRA's. Gradient-based solutions consist of solving a partial differential equation based on the gradient of the image brightness. The algorithms produce solutions by using discrete differentiations; for this reason, they are suitable for working on small displacements between consecutive frames, from 0 to 2 pixels. In this range, a high precision and conformance with the respect to the velocity field can be obtained depending on the model used, the precision of selected algorithms is higher than 0.1 pixels [2], [6]. For this reason, GBA's are profitably adopted as the basis of many dynamic scene analysis elaborations.

In general, GBA's are suitable for parallel implementation, since they require us to access only local image information. On the other hand, the early solutions to GBA's such as [17], [49], [51] were too computationally intensive to be profitably used for building VLSI chips. An example of this is the fully pyramidal implementation presented by Enkelmann in [52]. As a first step, a pyramid of images of different resolutions is obtained by using a Gaussian filter. Then, starting from the lowest resolution images, processing moves to higher resolutions, thus improving the optical flow estimation at each step. More recently, faster solutions have been defined, see [3], [14] for comparisons. A parallel implementation on the Connection Machine-2 architecture of the algorithm presented by Tretiak and Pastor in [49] has been presented by Tistarelli in [53], providing quasireal-time estimations. A faster parallel implementation on Connection Machine-2 of the algorithm proposed by Del Bimbo, Nesi, and Sanz in [6] has been presented by Del Bimbo and Nesi in [14]. In [14], the parallel implementations of the algorithm of Horn and Schunck [17] and that of Tretiak and Pastor [49] were also shown for the sake of comparison. This last solution presents more severe problems with respect to the behavior of the algorithm of Del Bimbo and Nesi, as demonstrated in [14] paper. From this comparison, the solution of Del Bimbo, Nesi, and Sanz [6] was determined to be the fastest and the most reliable. As discussed in the following, after additional experiments, this solution has been used for implementing RETIMAC.

A hardware implementation of Tretiak and Pastors' algorithm [49] was proposed by Danielsson *et al.* in [15], where an SIMD machine comprised of 512 processors is proposed to estimate optical flow fields (512×512) 10 to 15 times per second. The architecture needs 4 input images executing 136 MOPS; thus, this architecture is also too expensive to be profitably used in real applications. Each processor must be capable of estimating several products and two divisions.

The asymptotical complexity of GBA's can be from an $O(M^2 I_t G)$ or an $O(M^2 N^2 G)$, depending on the solutions

selected (iterative and direct solutions, respectively), where I_t is the number of iterations (from 10 to 100), N is the dimension of the image block from which the solution adopted extracts the image brightness values (typically from 5 to 15), G is the number of operations performed for each pixel of the block (typically from 5 to 50 depending on the solution adopted). According to the initial requirements, GBA's need from 0.78 to 70 MOPS. The main problem consists in the fact that most of these operations must be performed by considering the fractional part or by using at least integer variables with 32 b.

E. Approach Comparison and Discussion

Among the approaches discussed, it can be observed that FBA's and PRA's are not capable of producing optical flow fields with the required precision and robustness with respect to noise and local minima at low cost. Reliable versions of these approaches are computationally intensive to be considered for implementing low-cost hardware.

BMA's (and some PRA reduced solutions) have been used for image sequence coding in real-time since they are suitable for coding schemes based on cosine transform (DCT) and those used in the recent standards MPEG-1, MPEG-2, and H.261 [27]. For this purpose, GBA's have not been chosen up until now since they are not capable of estimating large displacements. This problem could be circumvented by adding special hardware for reducing the image resolution or by acquiring the image scene at a lower resolution.

GBA's are those which guarantee the best estimation precision of the optical flow with respect to the velocity field [3], [6]. This is mandatory for several computer vision applications—such as the measure of time to impact for obstacle avoidance, 3-D object reconstruction, the 3-D motion estimation and understanding of nonrigid objects, etc. From the point of view of complexity, GBA's have been considered for many years to be too complex to be adopted. This justifies the lack of VLSI architectures based on that approach. In general, GBA's are computationally intensive since the algorithms use many floating point operations. Some new algorithms for the GBA are capable of estimating reliable optical flow fields on sequential machines in less time than the classical MBA, as demonstrated in [3] and [14].

Therefore, on the basis of the presented considerations and experiments, the GBA was selected as a basis for identifying the suitable estimation algorithm for the RETIMAC chip.

III. Gradient-Based Optical Flow Estimation

Gradient-based techniques provide a solution for the motion estimation problem starting from the observation of brightness changes in the image plane [2], [5], [17], [49], [50], [54]. The optical flow is the field of the image brightness feature velocities and, therefore, it can differ from the perspective projection of 3-D motion on the image plane (i.e., the velocity field) depending on image acquisition conditions and on motion [1], [2], [54].

Most of the GBA's described in the literature are based on the so-called Optical Flow Constraint (OFC) equation. The OFC is derived from the observation that changes in image brightness $E[x(t), y(t), t]$ of each point in the image are supposed to be stationary with respect to the time variable (i.e., $dE/dt = \nabla E \cdot \boldsymbol{V} + E_t = 0$)

$$E_{x(x,y,t)}u_{(x,y,t)} + E_{y(x,y,t)}v_{(x,y,t)} + E_{t(x,y,t)} = 0 \quad (1)$$

in which the abbreviation for partial derivatives of image brightness has been introduced; $u_{(x,y,t)}$, $v_{(x,y,t)}$ correspond to dx/dt, dy/dt, and represent the components of the local velocity vector $\boldsymbol{V}_{(x,y,t)}$ along the x and y directions on the image plane at the time instant t, respectively. Images are sampled on a fixed grid of points at a regular time interval; an image at time t is the collection of the irradiance measures $E_{(i,j,t)}$ for $i = 1, \cdots, M$, and $j = 1, \cdots, M$ along the x- and y-axes, respectively.

A. Optical Flow Estimation Problems

In general, optical flow estimations present two main problems independently of the approach used among those previously discussed.

The first is the presence of discontinuities in the optical flow field. These are due to image brightness discontinuities which are originated by the presence of noise, too-crisp patterns on the moving object surfaces, occlusions between moving objects, and object velocities which are too fast for the measuring system. Generally speaking, the presence of discontinuities can be overcome (or at least attenuated) by filtering the image with a two-dimensional (2-D) or 3-D Gaussian smoothing operator at the expense of the computational effort [3], [6].

The second problem is the so-called "problem of aperture," which also exists in human vision, and derives from the impossibility to recover the direction of motion univocally if the object is observed through an aperture that is smaller than the object size. In this context, the references of the object under observation (such as textures, e.g., patterns) are not sufficient to detect the transversal component of the object motion, and only the component of apparent velocity which is parallel to ∇E can be detected [5], [18], [55]. This is coherent with human perception which is not able to detect the true direction of the velocity of an object if it does not have enough references, such as a pattern or an edge curvature. The gradient-based model can be useful to model this problem. In such cases, only the component of the velocity field which is parallel to the spatial gradient ∇E

$$\boldsymbol{V}_{\perp} = -\frac{E_t}{\|\nabla E\|}\frac{\nabla E}{\|\nabla E\|} \quad (2)$$

can be estimated and perceived, where $dE/dt = 0$ and $\|\nabla E\| \neq 0$ are assumed.

In the literature, two main approaches for gradient-based optical flow estimation can be identified: the *regularization-* and the *multiconstraint-based* approaches [3].

The *regularization-based* approaches estimate the optical flow field by minimizing a functional where a smoothness constraint is appropriately weighted to regularize the solution. The functional is minimized by using calculus of variations, and leads us to define iterative solutions [5], [17].

The *multiconstraint-based* approaches for estimating optical flow fields are based on the fact that it is usually possible to define more than one constraint equation [56]. They can be used to define an overdetermined system of equations with u and v as unknowns evaluated at the same point in the image [49], [50], or by considering that all the constraint equations which can be defined in the neighborhood of the estimation point represent the same optical flow field [6], [51], [57]. The latter approach is commonly referred to as a "multipoint" approach. The overdetermined system of equations can be solved by using the least-squares technique or by other means [6]–[58], [60].

B. Motion Estimation Technique

In [3], [6], [14], and [19] several comparisons among GBA's were made. In most cases, the comparison was made by adjusting algorithm parameters in order to get comparable results in terms of precision and reliability. From these comparisons, the solution by Del Bimbo, Nesi, and Sanz [6] presents the most interesting compromise among precision, reliability, and complexity. Its robustness with respect to noise and behavior has been demonstrated in [6] by comparing its results to the corresponding results obtained by using other well-known techniques. The consistency of the solution with respect to the conditions of aperture has been discussed in [61], demonstrating that its behavior is coherent with human vision.

For these reasons, such a solution was tested with an early DSP-based prototype [16], and then selected as a basis for implementing RETIMAC chip [62].

In [16], a DSP version of a system for counting moving people has been described. In that system, the optical flow estimation phase was performed by an ADSP 21 020 (Analog Device, 100 MFLOPS) instead of the ASIC discussed in this paper. In that system, the DSP estimated the optical flow fields while a low power microprocessor executed the reasoning in time for detecting, tracking, and counting passing people; DSP utilization was about 95%, considering also the costs of loading, saving, copying, images, and optical flow. In that architecture, the main reasons for demanding the estimation of the optical flow fields to a VLSI chip for optical flow estimation are the reduction in costs and dimensions of the equipment. Also the DSP-based architecture was used in the approach discussed in Section V-B for estimating optical flow fields with a reduction in the computational cost.

Therefore, RETIMAC implements the multipoint-based technique for optical flow estimation proposed by Del Bimbo, Nesi, and Sanz [6]. This *multipoint* approach is based on the fact that, by considering that the optical flow changes following a law which is approximately linear, a smoothed solution for optical flow estimation can be obtained from a linear approximation of the OFC equation in the neighborhood of the point under consideration [6] (this assumption is valid only if the optical flow field under observation is smooth). Consequently, a set of similar constraints in the neighborhood of a pixel yields an overdetermined system of equations.

A multipoint solution based on the OFC equation (1) is obtained in the discrete domain at the finite differences. Thus, for the estimation of velocity components for the pixel under consideration, an overdetermined system of $N \times N$ constraint equations in two unknowns, is defined as

$$E_{x(i,j,t-1)}u_{(x,y,t)} + E_{y(i,j,t-1)}v_{(x,y,t)} + E_{t(i,j,t-1)} = 0$$

for all (i, j) in an $N \times N$ neighborhood of the estimation point, where N is the dimension of the image segment side of the neighboring pixels, and $N \geq 2$. In this technique, a large value of N will lead to smooth optical flow estimations, and a loss in resolution in the estimation of velocity vectors. It should be noted that using central differences for the estimation of $E_{t(i,j,t)}$, the knowledge of the $E_{(x,y,t-1)}$ and $E_{(x,y,t+1)}$ is needed; thus, the coefficient of (1) can be evaluated only one time instant later with respect to the current time.

The overdetermined system of equations is solved by using a least-squares technique. In particular, after the estimation of image brightness partial derivatives in each pixel, an overdetermined system of $N \times N$ OFC equations in two unknowns is defined as

$$\boldsymbol{AV} + \boldsymbol{K} = 0$$

where $\boldsymbol{V}$ is the optical flow vector with components u, v; $\boldsymbol{A} \in \mathcal{R}_{N^2 \times 2}$ matrix of coefficients, with $a_{r,0} = E_{x_r}$ and $a_{r,1} = E_{y_r}$; and $\boldsymbol{K} \in \mathcal{R}_{N^2}$ vector with known terms $k_r = E_{t_r}$ for $r = 1, \cdots, N^2$. An increase in N, the neighborhood size considered, leads to a significant increase in memory requirements to store the matrix and vector elements (for instance, with $N = 7147$ memory locations are required for storing the coefficients of matrix $\boldsymbol{A}$ and vector $\boldsymbol{K}$ above).

The solution of the overdetermined system of equations by means of the least-squares technique is obtained by using the pseudoinverse technique, which transforms the above system of equations into a determined system of equations

$$\hat{\boldsymbol{A}}\boldsymbol{V} + \hat{\boldsymbol{K}} = 0 \tag{3}$$

where $\hat{A} = \boldsymbol{A}^T\boldsymbol{A}$, and $\hat{\boldsymbol{K}} = \boldsymbol{A}^T\boldsymbol{K}$ (i.e., $\boldsymbol{A}^T$ is the transpose of $\boldsymbol{A}$). This system of equations can be solved by using traditional techniques such as LU decomposition, Gauss Jordan, etc. It should be noted that the coefficients of the matrix $\hat{\boldsymbol{A}}$ and those of the vector $\hat{\boldsymbol{K}}$ are estimated by using

$$\hat{a}_{i,j} = \sum_{r=1}^{N^2} a_{i,r}^T a_{r,j} = \sum_{r=1}^{N^2} a_{r,i} a_{r,j}$$

$$\hat{k}_i = \sum_{r=1}^{N^2} a_{i,r}^T k_r = \sum_{r=1}^{N^2} a_{r,i} k_r.$$

The estimation of the $\hat{a}_{i,j}$ and $\hat{k}_i$ (for $i = 0, 1$; $j = 0, 1$) can be performed by accumulating one term at a time, from the rth neighboring OFC equation (for $r = 1, \cdots, N^2$), to obtain the final sum and thereby avoiding the need to store the entire set of N^2 OFC coefficients.

Among the collected N^2 OFC equations, those which have the E_t under a chosen threshold are ignored and considered as insignificant constraints since no difference in the image brightness is registered. Moreover, the constraint equations

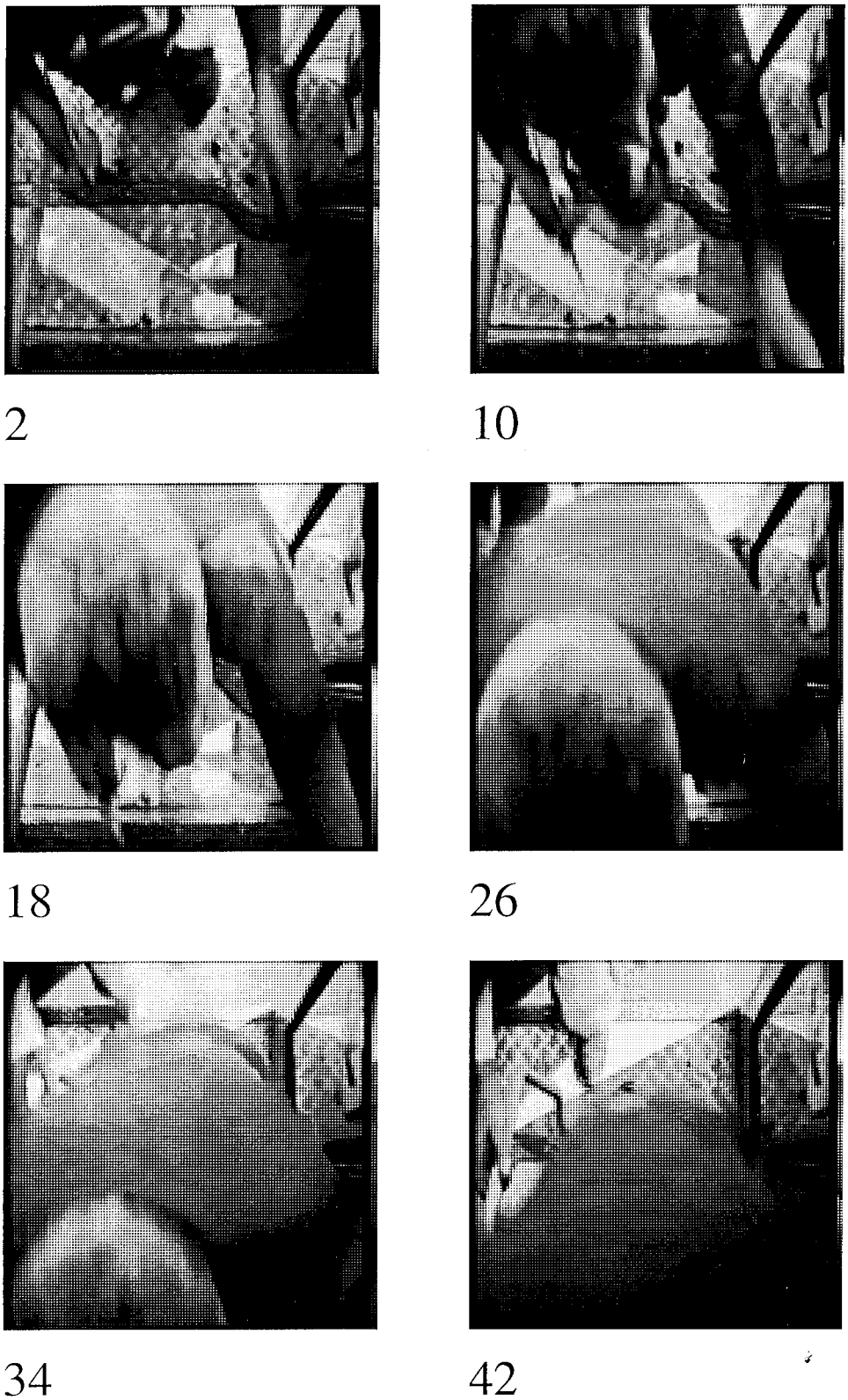

Fig. 1. Typical sequence of images where people are getting on a bus (frames: 2, 10, 18, 26, 34, 42, with resolution of 32 × 32).

which have values too large for E_x and E_y are also neglected, hypothesizing that there is a high probability that such large values are originated by noise.

In our case, system (3) is composed of two equations in two unknowns, and the direct solution is used: $u = U/div$, and $v = V/div$, where

$$U = \hat{a}_{0,1}\hat{k}_1 - \hat{k}_0\hat{a}_{1,1}$$
$$V = \hat{a}_{0,1}\hat{k}_0 - \hat{k}_1\hat{a}_{0,0}$$
$$div = \hat{a}_{0,0}\hat{a}_{1,1} - \hat{a}_{0,1}\hat{a}_{0,1}.$$

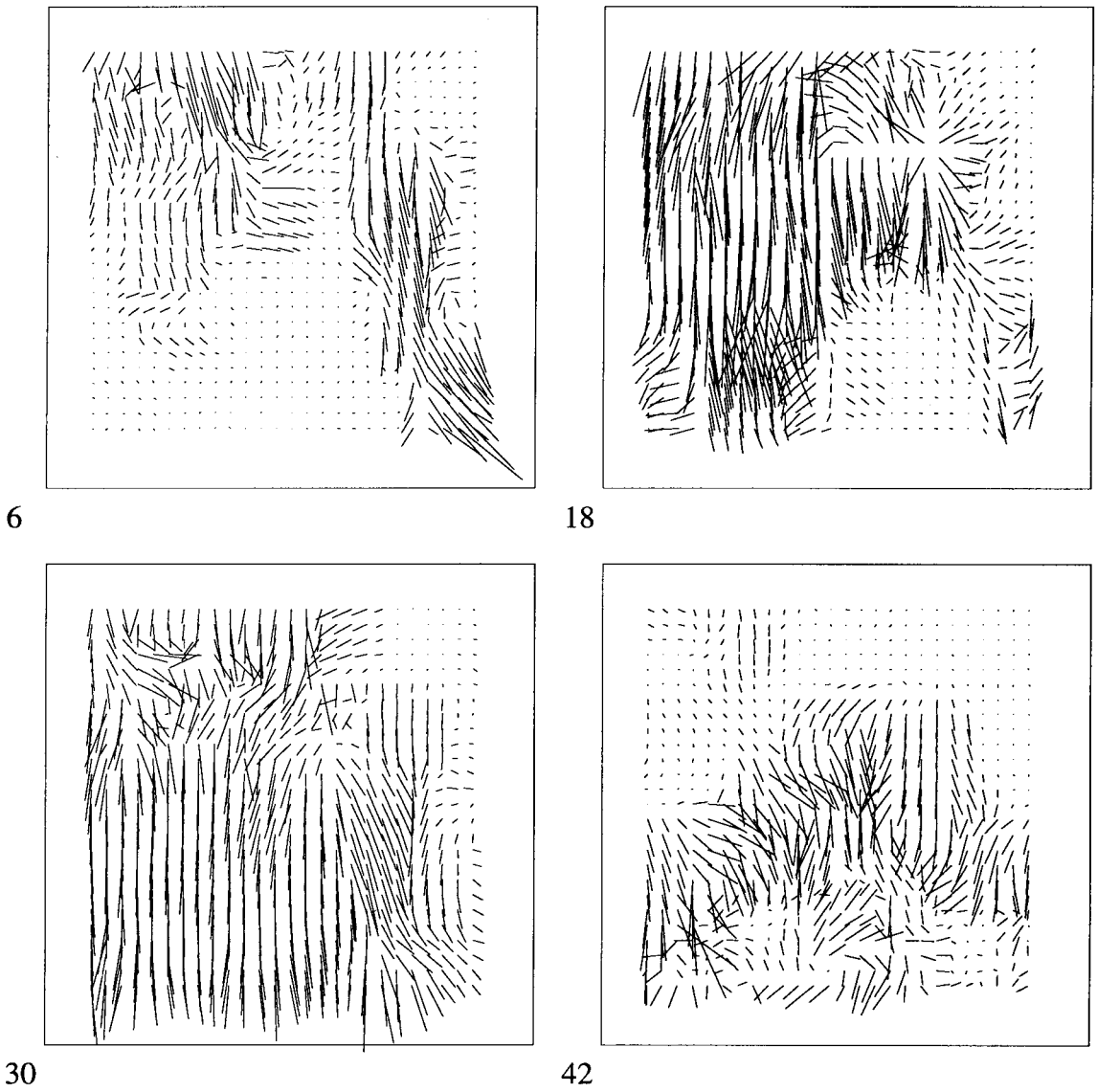

Fig. 2. Optical flow fields estimated from the image sequence reported with $N = 5$ (frames: 6, 18, 30, and 42).

1) A Simple Example: In order to test the robustness of the optical flow estimation technique used, several experiments were performed [6]. Since the chip has been especially implemented for counting moving people, particular attention was devoted to testing algorithm responses with respect to changes in illumination and weather, deformations due to nonrigidity of moving objects, and critical effects due to the out-of-focus and the incomplete vision of the moving object shape in the viewing area [16].

Fig. 2 shows optical flow fields estimated by using the method proposed on the image sequence of Fig. 1. The experimental results refer to the case of a person getting on/off a public bus. The image acquisition system is placed at the entrance of the bus over the stair steps, thus observing people from above.

As can be seen, the shape of the moving person is not immediately detectable from the optical flow fields. This is mainly due to the fact that the body is *noncompletely focused and included* in the viewing area of the image acquisition system. In addition, the moving object is *nonrigid* and its parts are moving at *different velocities* in intensity and direction. Obviously, optical flow estimation cannot solve the problems related to all moving object behaviors, as required by a system that performs people counting. These problems can only be solved by long-term analysis interpreting optical flow fields sequences in a temporal window. To this end, RETIMAC is capable of storing eight consecutive optical flow fields into its output memory. This allows spatiotemporal reasoning on optical flow fields. Flow of people is further interpreted as a flow of elementary moving particles (each of which has its own optical flow vector) and not as a sequence of single passages of large objects [16].

2) Computational Complexity: The explicit complexity of the solution proposed for estimating an optical flow field on an $M \times M$ image for a sequential machine is

$$C(\,) = 3 \ominus (M-2)^2 + (5 \oplus +5 \otimes) N^2 [M-d]^2 + (6 \otimes +3 \ominus +2 \varnothing)[M-d]^2 \tag{4}$$

where $d = 2[1 + (N - 1/2)]$ is due to the image boundaries.

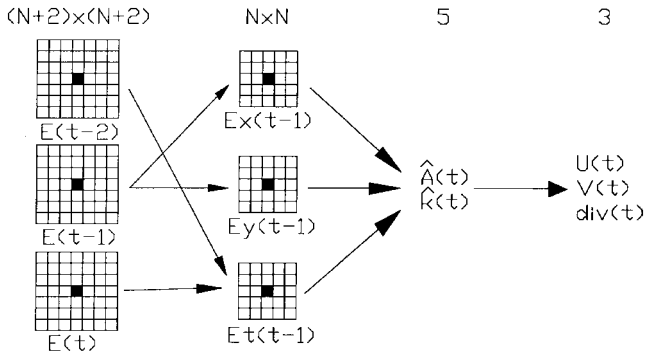

Fig. 3. Relationships and data transformations for estimating the optical flow vector corresponding to the dark square.

The first term is due to the estimation of the partial derivatives of image brightness, which are obtained by using central differences

$$E_{x(i,j,t-1)} = [E_{(i+1,j,t-1)} - E_{(i-1,j,t-1)}]/2$$
$$E_{y(i,j,t-1)} = [E_{(i,j+1,t-1)} - E_{(i,j-1,t-1)}]/2$$
$$E_{t(i,j,t-1)} = [E_{(i,j,t)} - E_{(i,j,t-2)}]/2.$$

where, in order to simplify the estimations the division by 2 can be omitted, since it appears in each OFC term.

The second additive term of (4) is due to the least-squares technique for calculating coefficients $\hat{a}_{i,j}$ and $\hat{k}_i$ (for $i = 0, 1$, and $j = 0, 1$), where the distance between two spatially consecutive estimation points has been assumed equal to one pixel (dense optical flow), and $[x]$ is the greatest integer number lower than x.

The third term is determined by the method for solving the final system of (3). In this phase, two divisions are also used for estimating u and v optical flow vector components from U, V and div which can be estimated by using integer mathematics. It should be noted that the magnitude of the optical flow components is from 0 to 2; thus, the divisions can be performed in fixed point only if a strong expansion of the scale is performed. Therefore, in order to simplify the RETIMAC chip, it has been decided to produce, as precise results, components U, V and div.

As can be seen from (4), the asymptotical complexity of the solution proposed is an $O(M^2N^2)$.

In Fig. 3, the relationships and data transformations among the phases of optical flow vector estimation, corresponding to the dark square, are reported. The algorithm implementation has been performed by using $N = 5$; hence, the estimation of each velocity vector needs an area of 7×7 pixels coming from three images. $N = 5$ has been used since this value represents a good compromise between estimation quality, noise robustness, and computational cost.

The computational cost in terms of the number of operations is equal to 0.178 MOP, with $N = 5$, $M = 32$ [where for the boundaries only $(M-6)^2$ velocity vectors are estimated]. From these values, the number of operations per second can be easily obtained, e.g., if 10 estimations per second with $M = 32$ are taken, a power of 1.78 MOPS is needed. On the other hand, by reusing partial results, the actual power can be reduced as discussed in Section V-B.

IV. RETIMAC Environment

On the basis of the above algorithm, the low-cost RETIMAC chip has been designed, implemented, and tested. This chip is capable of estimating dense optical flow fields in real-time, i.e., 10 times per second; in particular, producing U, V and div at 32 bits from which the velocity components u, v can be simply obtained. RETIMAC operates on images of 32×32 pixels; but if these dimensions are not sufficient, a composition of RETIMAC's can be adopted for covering the size required, thus defining a parallel architecture having the maximum of resolution equal to that of the sensor chip. RETIMAC is capable of grabbing images, estimating optical flow vector, and storing results in parallel. Being capable of estimating optical flow fields, RETIMAC can be used as a basis for several applications of dynamic scene analysis where a real-time motion estimation is needed.

RETIMAC has been designed for operating in a well-defined environment (see Fig. 4) comprised of the following.

1) *An Image Acquisition Sensor called Polifemo:* It consists of an array of 128×128 photodiodes [63], which is capable of capturing gray level images with 8 bits of resolution per pixel. This chip presents several options such as the possibility of selecting the window of interest (in our case a window of 32×32 is selected); and setting the integration time and, thus, the grabbing rate. If a set of RETIMAC chips is used for covering the whole image, different image windows should be selected (possibly partially covered) for considering boundary conditions. By using Polifemo, it is possible to regulate sensor sensitivity simply by changing the integration time. This is particularly useful for implementing an automatic regulation of sensor sensitivity directly on the microprocessor.
2) *4 Dual-Port SRAM's (1Kbyte per 8 bits):* It is used for storing 32×32 image data at 8 bits from the Polifemo sensor. These banks are the input memories of RETIMAC. This kind of architecture has been used for allowing the estimation of optical flow fields in parallel, with respect to the image acquisition. When RETIMAC transfers the image at time $t+1$ into the dual-port RAM0 with /CSB0A, at the same time it uses the other three chips of dual-ports for reading the images at time $t, t-1$, and $t-2$. By using this information, RETIMAC estimates the optical flow field corresponding to time $t-1$. At the next time instant, the dual-port containing the image at time $t-2$ (the oldest) is assigned to the image acquisition interface and, thus, RETIMAC estimates the next optical flow field by using the images at time $t+1, t$, and $t-1$. In this way, the transfer of data between different image memories is avoided, since the shifting of images is only performed by the dynamic addressing of the dual-port chip selects (/CSB0B-/CSB3B, /CSB0A-/CSB3A), with a savings in computational effort. It should be noted that port A of

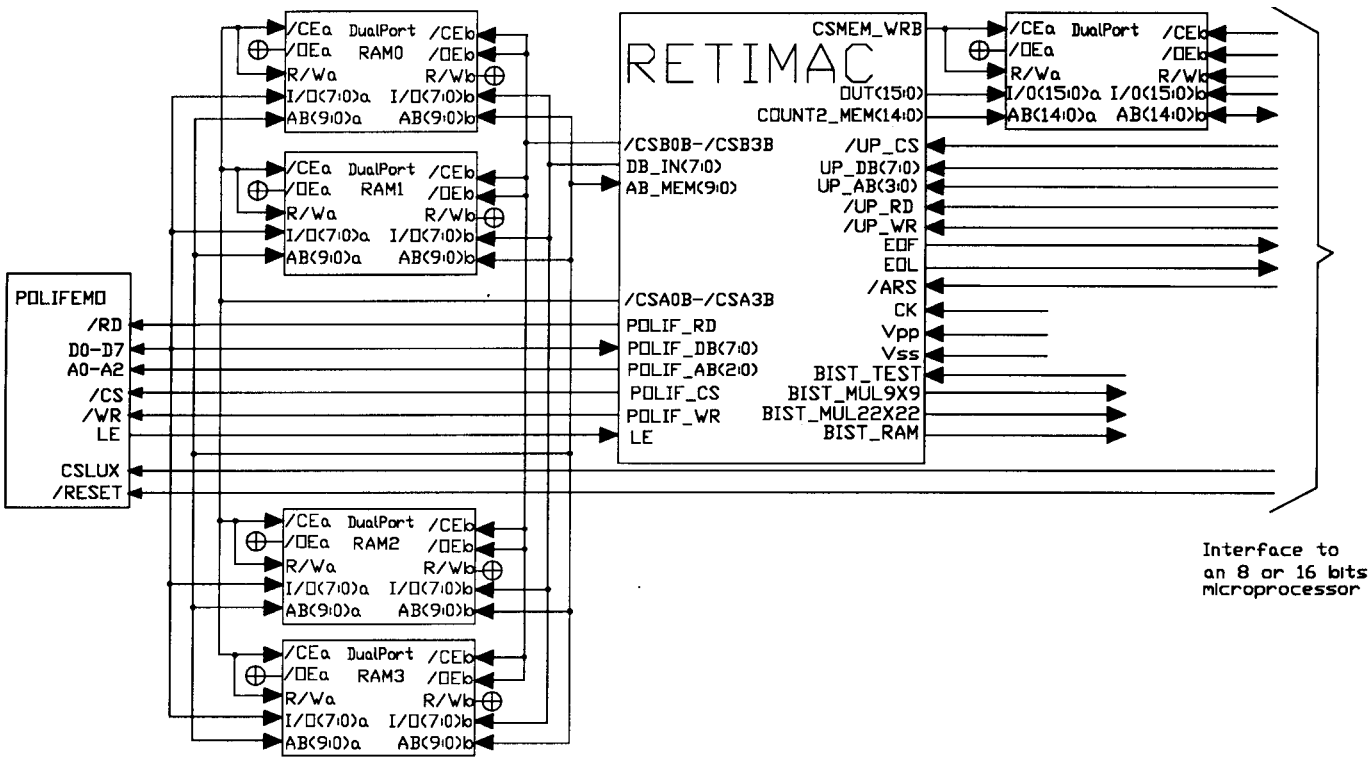

Fig. 4. Architecture of a system based on RETIMAC grabbing the images with the Polifemo chip.

input memories is always used for writing, while port B is only for reading the stored images.

3) *A Dual-Port Memory for the RETIMAC Results:* This bank has been designed to be read/written at 16 bits and is called the output memory of RETIMAC. In particular, the bank is written by the RETIMAC chip, while the same memory is accessed from the external CPU to read the results produced by RETIMAC, i.e., the optical flow vector components at 32 bits. In particular, the 32k $\times$ 16 bits output memory stores 32 bits results (16 bits at a time) in subsequent cells. It contains eight complete fields of 26 $\times$ 26 (26 and not 32 for the boundaries) U, V, div results. This allows the reasoning in time on the evolution of optical flow fields, without spending time to copy optical flow fields from the RETIMAC output memory on the microprocessor memory. The components of each single vector are stored with the following format as shown at the bottom of the page.
4) *A Microprocessor:* The microprocessor can read the results produced by RETIMAC directly on port B of the output memory (dual-port). As already pointed out, the results are the values U, V and div, from which the components of the actual optical flow vectors can be easily obtained. Moreover, the microprocessor can set and reset Polifemo and RETIMAC configurations. For these operations, it must write into the Polifemo internal registers (resolution, frame size, integration time, wait time, start, etc.) and into the RETIMAC control register to impose the operating modes. In addition, the microprocessor can perform higher-level processing on the optical flow fields estimated, e.g., interpreting the optical flow in time for counting moving people such as in our main application [16], identifying moving objects, etc. Usually, this processing is much less complex and costly than the optical flow estimation; thus, the microprocessor can even have low power.

In the following, the typical sequence of the main phases for optical flow processing and sensor setup are reported:

ARS;
SENSOR SET-UP 1;
START O.F.Estimation 1;
WAIT;
SENSOR SET-UP 2;
RESET;
START O.F.Estimation 2.

If the microprocessor needs to modify the sensor setup, it can keep RETIMAC in a wait mode for changing the contents of Polifemo internal registers and then reset and restart the optical flow estimation processing with a new configuration.

Table I shows the RETIMAC timing performance related to the estimation of optical flow and the range for the integration time of Polifemo.

div(31:16)	div(15:0)	U(31:16)	U(15:0)	V(31:16)	V(15:0)

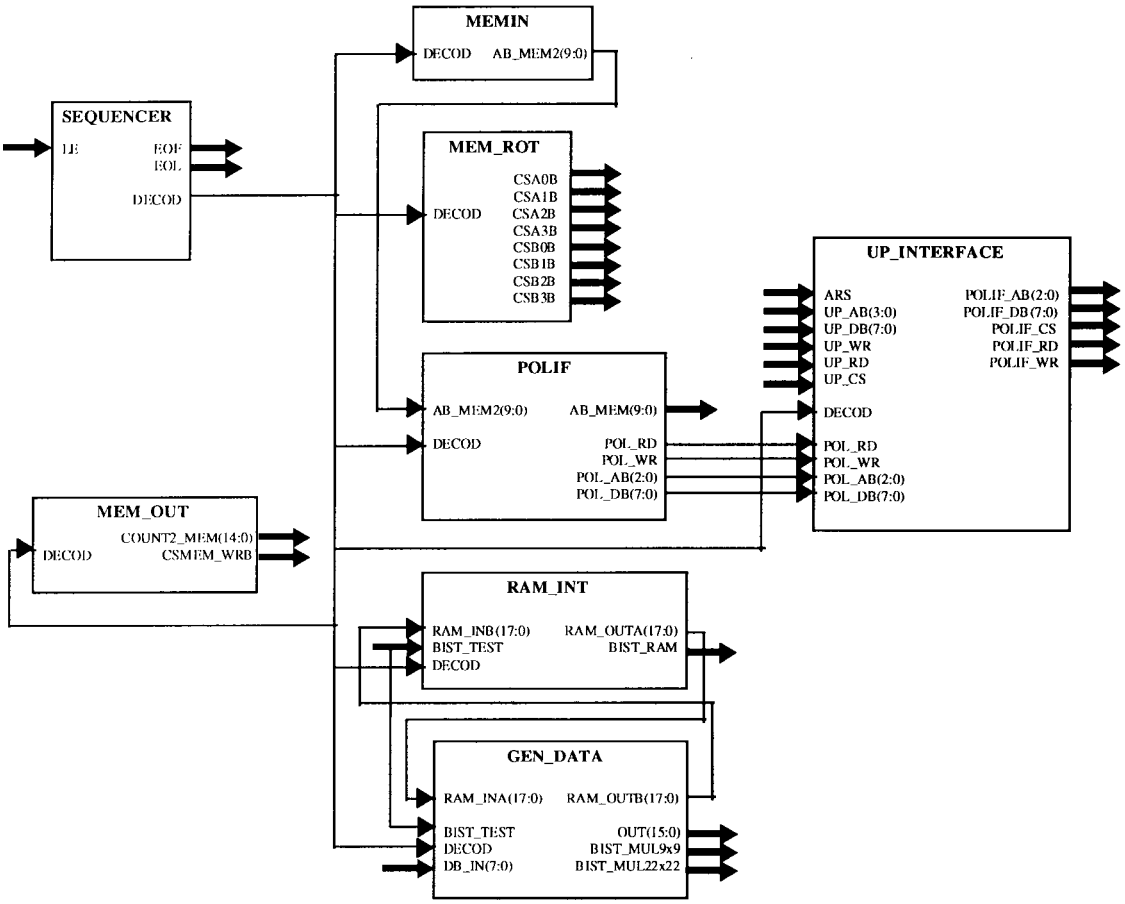

Fig. 5. RETIMAC block diagram.

TABLE I
RETIMAC PERFORMANCE

RETIMAC system clock	$Fmax = 10MHz$
OF row processing time	$T_{ROWOF} = 189.9\mu s$
OF processing time for a frame of 32 × 32	$T_{OFframe} = 6.076ms$
Min integration time for a row	$T_{IROWmin} = 191\mu s$
Max integration time for a row to perform $10frame/s$	$T_{IROWmax} = 3.117ms$

V. RETIMAC FUNCTIONAL DESCRIPTION

RETIMAC is a digital CMOS ASIC (1 μm ECPD10 ES2 technology) that acts as a coprocessor dedicated to estimating optical flow fields by using the previously presented algorithm. Internally, it is comprised of 8 main subsystems as shown in Fig. 5. The subsystems, with their respective roles, are as follows.

- **SEQUENCER** is comprised of three parts: a state generator, a state decoder, and a control generator for data path. The sequencer constitutes the state machine which manages all the sequential operations of the device.
- **MEMIN** is the interface toward the 4 dual-ports input memories. It generates a special addressing ("cross" addressing) to get directly samples of 7 × 7 pixels of submatrix, that is, the basic data for the algorithm for optical flow estimation. The sequence of addresses is strongly simplified by the fact that four dual-ports are used instead of a single one.
- **MEM_ROT** performs the dynamic selection ("rotation technique") of input memories at the end of each frame. At the same time, the fourth memory is configured for copying the image from the Polifemo chip. The mechanism implemented has been described in the previous section.
- **POLIF** consists of the interface between RETIMAC and Polifemo optical sensor. This block is capable of transferring image segments from the Polifemo chip to input memories line by line, 32 lines per frame.
- **MEM_OUT** is the interface toward 32k × 16 bits memory for the output of results. This block also manages the rotation of optical flow fields position in the output memory. Each optical flow needs 26^2 cells of 6 bytes, that is, 4056 bytes; thus, every 4096 bytes, an optical flow field is stored.
- **UP_INTERFACE** is the interface with respect to the microprocessor. It allows the chip reset and the accessing to the internal register of RETIMAC (see Section V-A).
- **RAM_INT** is a 125 × 18 bits internal RAM, which is used to perform read–modify–write operations on intermediate results $(E_x^2, E_y^2, E_xE_y, E_xE_t, E_yE_t)$ for estimating coefficients of matrix $\boldsymbol{A}$ and vector $\boldsymbol{K}$, during the optical flow estimation. This memory allows us to achieve a special data path architecture that calculates the velocity vectors very quickly, as discussed in the following.

TABLE II
CONTROL REGISTER: DECODING AND OPERATING MODES

up_db(1:0)	Control Register (Write only) decoding
00	RESET MODE
01	START Optical Flow Estimation MODE
10	WAIT MODE

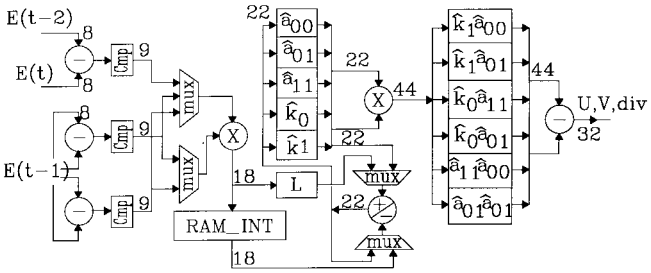

Fig. 6. The main blocks of RETIMAC data path.

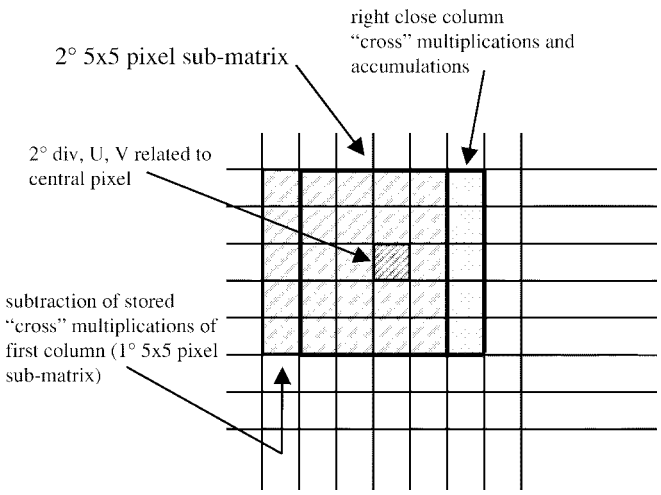

Fig. 7. Technique for estimating the next velocity vector by using read–modify–write operation of internal RAM.

- **GEN_DATA** is the data path with a pipeline structure that leads to the estimation of the optical flow vectors (see in Section V-B for details).

A. Operating Modes

RETIMAC presents some operating modes which are set by selecting the RETIMAC internal control register that is written by the microprocessor using address *up_ab(3:0) = 8*, data bus *up_db(7:0)* and write signal *up_wr*; Table II shows control register decoding, while in the following a brief discussion of each operating mode is reported. The operating modes of RETIMAC are not limited to those reported in the table; the complete set is discussed in the following, together with the operating modes of a system based on RETIMAC.

1) *SENSOR SETUP:* Before performing other operations, the microprocessor must take control of the sensor to begin its setup: to this end, the micro must reset RETIMAC by activating the asynchronous reset ARS or by writing the RESET MODE code into the RETIMAC internal control register, so that this releases the control of the sensor to the microprocessor. The sensor setup can consist of imposing the image window to be grabbed, as well as setting the integration time and/or other registers [63].
2) *START Optical Flow Estimation MODE:* After sensor setup, the microprocessor writes "START Optical Flow Estimation MODE" (see Table II) into RETIMAC control register. As a consequence, RETIMAC waits the line ready, LE (synchronism signal), from Polifemo and, then, it begins the optical flow field estimation through digital processing in four phases.

   - *Input memory loading*: RETIMAC stores 8-bit sensor data of an image line (32 samples) into the selected port A of the assigned input dual-port memory.
   - *Asking for the next row and optical flow estimation processing*: the chip writes to Polifemo the next row to be grabbed by addressing a specific register; *at the same time*.
   - *Optical flow estimation processing*: the chip executes the optical flow estimation algorithm getting data from the other three input RAM's by means of their ports B.
   - *End of line and end of frame*: at the end of line processing RETIMAC activates the EOL signal; after 32 EOL's, the EOF (end of frame) signal goes low indicating that a complete set of frame results is ready in the output memory.
3) *WAIT and RESET:* When RETIMAC is in one of these states, as well as when RETIMAC is between EOF and the next LE, the microprocessor can directly manage the Polifemo sensor. The microprocessor can write the internal registers of Polifemo and/or directly read the grabbed image. This allows the adoption of more than one RETIMAC chip on the same Polifemo chip.
4) *TESTING MODE:* The BIST_TEST pin is capable of starting a built-in testing mode on silicon compiled macrocells (MUL9×9 and MUL22×22 internal multipliers, and RAM125×18 internal memory): note the characteristic signature of testing structure at the three dedicated outputs: BIST_MUL9×9, BIST_MUL22×22, BIST_RAM.

Once the silicon compiled macrocells are tested, the chip behavior can be tested by using a set of test vectors including sequences of images and generated optical flow fields. To this end, during the first task of project OFCOMP, a simulator of RETIMAC chip has been implemented [62].

B. Data Path Description

It is very interesting to analyze in detail the complete data flow and technique used to estimate optical flow vectors row by row (see Fig. 6). The algorithm is mainly contained inside block GEN_DATA.

The first operation consists of calculating partial derivatives of the image brightness (E_x, E_y, E_t) for each element of the

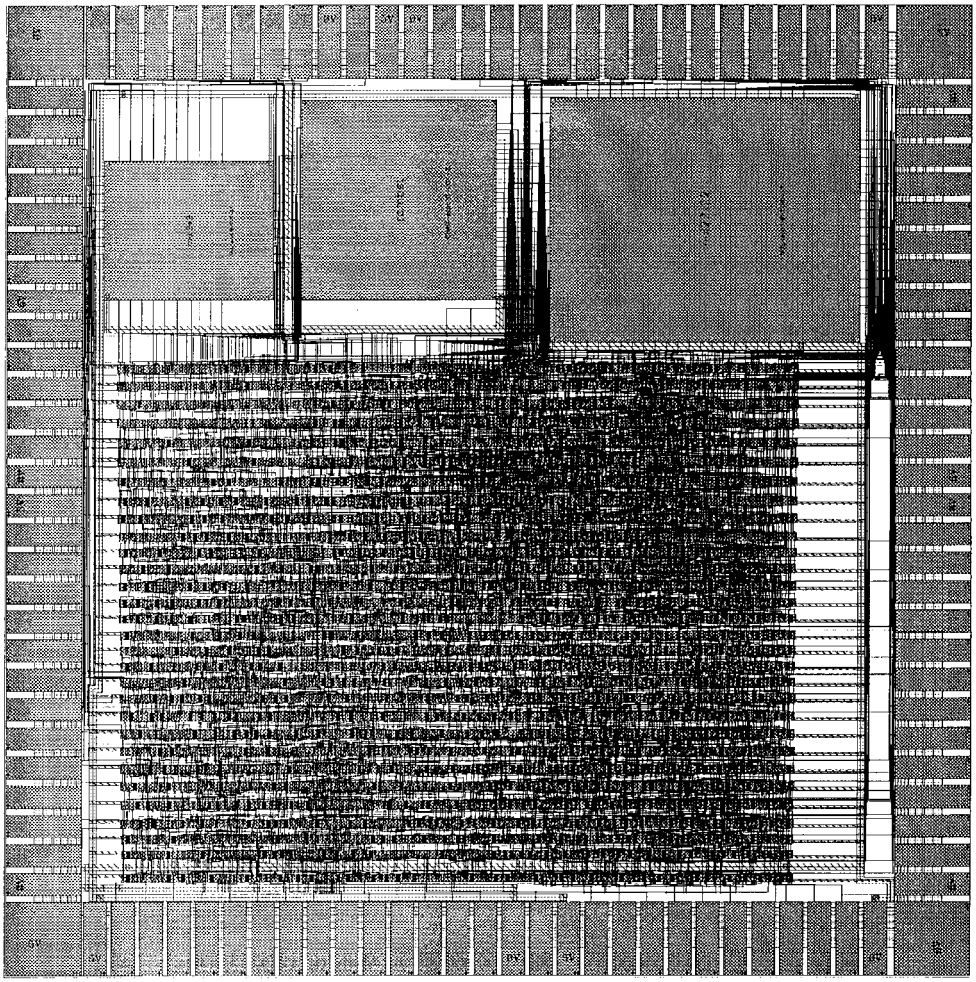

Fig. 8. RETIMAC chip layout.

first 5×5 pixels submatrix. Each partial derivative is passed to a comparator. According to Section III-A, the OFC equations which have 1) $E_t = 0$ or 2) $|E_x|$ and $|E_y|$ larger than a fixed threshold are neglected. Thus, if the comparator produces a negative result, the corresponding constraint equation is not considered; this allows us to eliminate unsuitable OFC's. This also avoids overflow in the last multiplicator, and is the reason why only the first 32 bits of U, V and div are saved into the output memory instead of 44 bits. After several experiments, we have fixed the threshold to 180.

The second step is to execute multiplications among the partial derivatives of the image brightness (E_x^2, E_y^2, E_xE_y, E_xE_t, E_yE_t) associated with each element of the submatrix. These results are accumulated in the internal memory (RAM_INT) of $5 \times 5 \times 5$ elements of 18 bits each. At the same time, these values are used for estimating coefficients: $\hat{a}_{00}$, $\hat{a}_{01}$, $\hat{a}_{11}$, $\hat{k}_0$, $\hat{k}_1$ by accumulating those of each pixel of the neighborhood according to the above-reported equations (this is done by using register L as reported in Fig. 6. On the basis of these coefficients, the solution of the system of two equation in two unknowns can be obtained by estimating the multiplications (with a 44 bits output) between coefficients and, thus, estimating the differences. These results are produced on 45 bits, from which the components U, V and div are obtained taking the 32 LSB's. This can be done since the subtraction makes a strong reduction in magnitude.

It should be noted that for calculating the first vector components U, V, div of an optical flow field, a scan by column of the first 5 columns is needed. Then, proceeding to the next pixel, only the next column on the right must be processed in order to estimate partial derivatives and their corresponding multiplications (see Fig. 7). In particular, before collecting that data, the coefficients $\hat{a}_{00}$, $\hat{a}_{01}$, $\hat{a}_{11}$, $\hat{k}_0$, $\hat{k}_1$ represent the data relative to the previous pixel. For estimating the new set of coefficients, it is enough to subtract from these values the multiplications of the image brightness (5 for each pixel, 25 per column) relative to the oldest column on the left of the neighborhood, and to add the new multiplications of the new column on the right (5 for each pixel, 25 per column). This mechanism is repeated for each element of the new column: estimating multiplications of the image brightness, extracting the old value from the memory, subtracting its value from the corresponding coefficient, adding the new multiplications to the coefficients; these are also stored in the memory once they are produced.

In this way, it is sufficient to calculate 25 accumulations on the right column close to the first submatrix in order to obtain the next vector, by removing at the same time

25 accumulations related to the first calculated column with a read–modify–write operation on internal memory. This is performed by using the direct connection of the output of block sub/add to one of its inputs as reported in Fig. 6. This mechanism reduces the computational effort to 0.067 MOP, and thus RETIMAC executes 0.67 MOPS, without considering the time for grabbing images, loading image pixel values, saving optical flow fields, etc.

VI. Discussion and Conclusions

RETIMAC has been implemented with 1-μm CMOS ECPD10 ES2 process, has 28 inputs, 61 outputs, 8 tristate, 3 NC. It presents 77 602 transistors, for about 19 400 equivalent gates, and a silicon area equal to 32.715 mm^2 (34.453 mm^2 with 150-μm margin for die cut). The layout of RETIMAC is shown in Fig. 8, where the area on top right is the 22×22 multiplicator, and that on the top left the 18×18 multiplicator, while the area in the center is the memory of 125 registers of 18 bits each.

The RETIMAC chip is capable of estimating optical flow fields in real-time and has been developed for project OF-COMP (Optical Flow for COunting Moving People) DIM 45 ESPRIT III MEPI [62]. A system based on RETIMAC for counting people is an improvement of a preliminary system in which a high performance DSP was used for the same purpose [16].

It should be noted that the counting of people getting in/out of a bus is only one of the various real-time applications in which RETIMAC could be used, e.g., autonomous navigation, surveillance, tracking of moving objects, etc. Therefore, the adoption of RETIMAC could allow the implementation of low-cost real-time systems with a stage of optical flow field estimation, and particularly suitable for the applications in which the spatiotemporal reasoning about flow fields is mandatory.

References

[1] A. Verri and T. Poggio, "Motion field and optical flow: Qualitative properties," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 11, pp. 490–498, May 1989.

[2] A. Del Bimbo, P. Nesi, and J. L. C. Sanz, "Analysis of optical flow constraints," *IEEE Trans. Image Processing*, vol. 4, pp. 460–469, Apr. 1995.

[3] P. Nesi, "Real-time motion analysis," in *Real-Time Imaging: Theory, Techniques, and Applications*, P. Laplante and A. Stoyenko, Eds. New York: IEEE, May 1996, pp. 27–57.

[4] M. Bertero, T. A. Poggio, and V. Torre, "Ill-posed problems in early vision," *Proc. IEEE*, vol. 76, pp. 869–889, Aug. 1988.

[5] P. Nesi, "Variational approach for optical flow estimation managing discontinuities," *Image Vis. Computing*, vol. 11, no. 7, pp. 419–439, 1993.

[6] A. Del Bimbo, P. Nesi, and J. L. C. Sanz, "Optical flow computation using extended constraints," *IEEE Trans. Image Processing*, vol. 5, pp. 701–719, 1996.

[7] H. G. Musmann, P. Pirsh, and H.-J. Grallert, "Advances in picture coding," *Proc. IEEE*, vol. 73, pp. 523–548, Apr. 1985.

[8] P. Dufaux and F. Moscheni, "Motion estimation techniques for digital tv: A review and a new contribution," *Proc. IEEE*, vol. 83, pp. 858–876, June 1995.

[9] J. K. Aggarwal and N. Nandhakumar, "On the computation of motion from sequences of images—A review," *Proc. IEEE*, vol. 76, pp. 917–935, Aug. 1988.

[10] A. Artieri and F. Jutand, "A versatile and powerful ship for real-time motion estimation," in *Proc. ICASSP'89*, Glasgow, Scotland, 1989, pp. 2453–2456.

[11] K.-M. Yang, M.-T. Sun, and L. Wu, "A family of VLSI design for the motion compensation block-matching algorithm," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 1317–1325, Oct. 1989.

[12] L. DeVos and M. Schöbinger, "VLSI architecture for a flexible block matching processor," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, pp. 417–428, Oct. 1995.

[13] A. Borri, G. Bucci, and P. Nesi, "A robust tracking of 3d motion," in *Proc. Euro. Conf. Computer Vision, ECCV'94*, Stockholm, Sweden, May 2–6, 1994, pp. 181–188.

[14] A. Del Bimbo and P. Nesi, "Optical flow estimation on connection machine-2," in *Proc. Int. Workshop Computer Architecture Machine Perception, CAMP'93*, New Orleans, LA, Dec. 15–17, 1993.

[15] P. Danielsson, P. Emanuelsson, K. Chen, P. Ingelhag, and C. Svensson, "Single-chip high-speed computation of optical flow," in *Proc. Int. Workshop Machine Vision Appl. MVA'90 IAPR*, Tokyo, Japan, Nov. 28–30, 1990, pp. 331–335.

[16] P. Nesi and A. Del Bimbo, "A vision system for estimating people flow," in *Image Technology: Advances in Image Processing, Multimedia and Machine Vision*, J. L. C. Sanz, Ed. New York: Springer-Verlag, 1996, pp. 179–201.

[17] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, pp. 185–203, 1981.

[18] A. Singh, *Optic Flow Computation: A Unified Perspective*. Los Alamitos, CA: IEEE Comput. Society Press, 1991.

[19] J. L. Barron, D. J. Fleet, and S. S. Beauchemin, "Performance of optical flow techniques," *Int. J. Comput. Vis.*, vol. 12, no. 1, pp. 43–77, 1994.

[20] J. P. H. van Santen and G. Sperling, "Temporal covariance model of human motion perception," *J. Opt. Soc. Amer. A*, vol. 1, pp. 451–473, May 1984.

[21] E. H. Adelson and J. R. Bergen, "Spatiotemporal energy models for the perception of motion," *J. Opt. Soc. Amer. A*, vol. 2, pp. 284–299, Feb. 1985.

[22] D. Heeger, "Model for the extraction of image flow," *J. Opt. Soc. Amer. A*, vol. 4, pp. 1455–1471, Aug. 1987.

[23] A. B. Watson and A. J. Ahumada, Jr., "Model of human visual-motion sensing," *J. Opt. Soc. Amer. A*, vol. 2, pp. 322–341, Feb. 1985.

[24] L. Jacobson and H. Wechsler, "Derivation of optical flow using a spatiotemporal-frequency approach," *Comput. Vis., Graph., Image Processing*, vol. 38, pp. 29–65, 1987.

[25] P. Anandan, "A unified perspective on computational techniques for the measurement of visual motion," in *Proc. 1st IEEE Int. Conf. Computer Vision ICCV'87*, London, England, June 8–11, 1987, pp. 219–230.

[26] P. J. Burt, C. Yen, and X. Xu, "Multiresolution flow-through motion analysis," in *Proc. of the IEEE Conf. Computer Vision Pattern Recognition, CVPR'83*, Washington, DC, June 19–23, 1983, pp. 246–252.

[27] B. Furht, "A survey of multimedia compression techniques and standards. Part II: Video compression," *Real-Time Imaging*, vol. 1, pp. 319–337, 1995.

[28] L. DeVos and M. Stegherr, "Parametrizzable VLSI architectures for the full-search block-matching algorithm," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 1309–1316, Oct. 1989.

[29] T. Komarek and P. Pirsch, "Array architectures for block matching algorithms," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 1301–1308, Oct. 1989.

[30] C.-H. Hsieh and T.-P. Lin, "VLSI architecture for block matching motion estimation algorithm," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 2, pp. 169–175, June 1992.

[31] H. Yeo and Y. H. Hu, "A novel modulae systolic array architecture for full-search block matching motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, pp. 407–416, Oct. 1995.

[32] H.-N. Kim, M. J. Irwin, and R. M. Owens, "Motion analysis of the micro grained array processor," Special Issue on *Real Time Imaging*, P. Nesi, Ed. *J. Real-Time Syst.*, vol. 3, no. 2, pp. 101–110, 1997.

[33] ———, "L64720 motion estimation processor," LSI Logic Corp., Tech. Rep., 1990.

[34] ———, "Data book: Sti3220 motion estimation processor," SGS-Thompson, Tech. Rep., 1991.

[35] S. Chang, J.-H. Hwang, and C.-W. Jen, "Scalable array architecture design for full search block matching," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, pp. 332–343, Aug. 1995.

[36] M.-J. Chen, L.-G. Chen, T.-D. Chiueh, and Y.-P. Lee, "A new block-matching criterion for motion estimation and its implementation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, pp. 231–236, June 1995.

[37] T. Komarek and P. Pirsch, "VLSI architectures for hierarchical block matching algorithm," in *Proc. IFIP Workshop*, Grenoble, France, Dec. 1989, pp. 168–181.

[38] L. DeVos, "VLSI-architecures for the hierarchical block-matching algorithm for hdtv applications," *SPIE Visual Commun. Image Processing*, vol. 1360, pp. 398–409, 1990.

[39] B.-M. Wang, J.-C. Yen, and S. Chang, "Zero waiting-cycle hierarchical," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, pp. 18–28, Feb. 1994.
[40] H.-M. Jong, L.-G. Chen, and T.-D. Chiueh, "Parallel architecture for 3-step hierarchical search block-matching algorithm," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, no. 4, pp. 407–416, 1994.
[41] G. Gupta and C. Chakrabarti, "Architectures for hierarchical and other block matching algorithms," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, pp. 477–489, Dec. 1995.
[42] H. Bülthoff, J. J. Little, and T. Poggio, "A parallel algorithm for a real-time computation of optical-flow," *Nature*, vol. 337, pp. 549–553, Feb. 9, 1989.
[43] F. Glazer, G. Reynolds, and P. Anandan, "Scene matching by hierarchical correlation," in *Proc. IEEE Conf. Computer Vision Pattern Recognition, CVPR'83*, Washington, DC, June 19–23, 1983, pp. 432–441.
[44] K. M. Nam, J.-S. Kim, R.-H. Park, and Y. S. Shim, "A fast hierarchical motion vector estimation algorithm using mean pyramid," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, pp. 344–351, Aug. 1995.
[45] A. N. Netravali and J. D. Robbins, "Motion-compensated television coding: Part I," *Bell Syst. Tech. J.*, vol. 58, pp. 631–670, Mar. 1979.
[46] R. C. Kim and S. U. Lee, "A VLSI architecture for a pel recursive motion estimation algorithm," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 36, pp. 1291–1299, Oct. 1989.
[47] E. D. Frimout, J. N. Driessen, and E. F. Deprettere, "Parallel architecture for a pel-recursive motion estimation algorithm," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 2, pp. 159–168, June 1992.
[48] F. Charot, C. Labit, and P. Lemonnier, "Architectural study of a block-recursive motion estimation algorithm," Special Issue on Real Time Imaging, P. Nesi, Ed. *J. Real-Time Syst.*, vol. 3, no. 2, pp. 111–128, Apr. 1997.
[49] O. Tretiak and L. Pastor, "Velocity estimation from image sequences with second order differential operators," in *Proc. 7th IEEE Int. Conf. Pattern Recognition*, Montreal, P.Q., Canada, July 30–Aug. 2, 1984, pp. 16–19.
[50] A. Verri, F. Girosi, and V. Torre, "Differential techniques for optical flow," *J. Opt. Soc. Amer. A*, vol. 7, pp. 912–922, May 1990.
[51] C. Caffario and F. Rocca, "Tracking moving objects in television images," *Signal Process.*, vol. 1, pp. 133–140, 1979.
[52] W. Enkelmann, "Investigation of multigrid algorithms for the estimation of optical flow fields sequences," *Comput. Vis., Graph., Image Processing*, vol. 43, pp. 150–177, 1988.
[53] M. Tistarelli, "Computing optical flow: A real time application of the connection machine system, tr v89-1," Thinking Machines Corp., DIST Genova, Cambridge, MA, Tech. Rep., June 1989.
[54] H.-H. Nagel, "On a constraint equation for the estimation of displacement rates in image sequences," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 11, pp. 13–30, Jan. 1989.
[55] A. Del Bimbo, P. Nesi, and J. L. C. Sanz, "Optical flow estimation by using classical and extended constraints," in *Proc. 4th Int. Workshop Time-Varying Image Processing Moving Object Recognition*, Florence, Italy, June 10–11, 1993.
[56] H.-H. Nagel, "On the estimation of optical flow: Relations between different approaches and some new results," *Artif. Intell.*, vol. 33, pp. 299–324, 1987.
[57] M. Campani and A. Verri, "Computing optical flow from an overconstrained system of linear algebraic equations," in *Proc. 3rd IEEE Int. Conf. Computer Vision ICCV'90*, Osaka, Japan, Dec. 4–7, 1990, pp. 22–26.
[58] B. G. Schunck, "Image flow segmentation and estimation by constraints line and clustering," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 11, pp. 1010–1027, Oct. 1989.
[59] D. Ben-Tzvi, A. Del Bimbo, and P. Nesi, "Optical flow from constraint lines parametrization," *Pattern Recognition*, vol. 26, pp. 1549–1561, Nov. 1993.
[60] P. Nesi, A. Del Bimbo, and D. Ben-Tzvi, "A robust algorithm for optical flow estimation," *Comput. Vis., Graph., Image Processing: Image Understanding*, vol. 61, pp. 59–68, 1995.
[61] A. Del Bimbo, P. Nesi, and J. L. C. Sanz, "Innovative multipoint solutions for optical flow estimation with different constraints," in *Proc. Int. Conf. Image Anal. Processing, IAPR*, Bari, Italy, Sept. 20–23, 1993.
[62] P. Nesi, "Project OFCOMP: Optical flow for counting moving people, deliverable of task 1.0 (chip RETIMAC specification) of DIM 45 Esprit iii Project MEPI n.7500, rt 25/94," Dip. Sistemi e Informatica, Facolta' di Ingegneria Univ. Studi di Firenze, Florence, Italy, Tech. Rep., Feb. 1994.
[63] M. Gottardi, A. Sartori, and A. Simoni, "Polifemo: An addressable CMOS 128×128-pixel image sensor with digital interface; Electrical specifications," IRST, Istituto per la Ricerca Scientifica e Tecnologica, I 38100 TRENTO, Loc. Pante' di Povo, Tech. Rep., Dec. 1992.

Paolo Nesi (M'92) received the doctoral degree in electronic engineering from the University of Florence, Italy, and the Ph.D. degree in from the University of Padoa, Italy, in 1992.

In 1991, he was a visitor at the IBM Almaden Research Center. Since November 1991, he has been with the Department of Systems and Informatics of the University of Florence, Italy, as a Researcher. Since 1987, he has been active on several research topics, including real-time systems, machine vision, physical models, parallel architectures, object-oriented technology, and formal languages. He is the scientist responsible at the CESVIT (High-Tech Agency) regarding High Performance Computer Networking technology and projects.

Dr. Nesi is a member of the program committees of several international conferences (e.g., ICECCS'96 and ICECCS'97, "IEEE International Conference on Engineering of Complex Computer Systems; AQUIS'96, CSMR'97, etc.; and Guest Editor of special issues of international journals. He is an editorial board member of the *Journal of Real-Time Imaging* and of Academic Press. He has been General Chair of Objective Quality Symposium 1995, LNCS n.926, Springer. He is a member of the IEEE Computer Society and the IEEE Systems, Man, and Cybernetics Society, IAPR (International Association for Pattern Recognition), TABOO, and AIIA (Italian Association on Artificial Intelligence).

Fabrizio Innocenti received the doctoral degree in electronic engineering from the University of Florence, Italy, in 1990.

He joined the SMA S.p.A. in 1989 as a digital ASIC designer, where he worked for five years. In 1994 he joined the constitution of CESVIT (High-Tech Agency, Florence, Italy) Centro di Microelettronica, where he is currently working as the Executive Manager, supervising all the designs carried out by the Center on different technologies ranging from microcontroller and FPGA to ASIC, MCM, and microsystem. He is the lecturer in charge at the University of Siena for digital ASIC design. He serves as an expert evaluator for two pan-European EC Projects.

Paolo Pezzati received the doctoral degree in electronic engineering from the University of Florence, Italy, in 1993.

From 1994 to 1996, he was an ASIC engineer at CESVIT, High Tech Agency in Florence. His main research activities included digital signal custom processor (RETIMAC), Cassini Project (aiHDL description and synthesis, chip assembly management). Since 1996, he has been with ULSI as a physical engineer at SGS-Thomson Microelectronics based in San Jose, CA and St. Genis, France. His main activities include Super Integration Program engineer, PC system-on silicon chip assembly flow, clock tree synthesis for deep-submicron design, and data path architecture.