Real-time Estimation of Object Trajectories in Image Sequences

LÍRIDA ALVES DE BARROS

Département Electronique, ENST/Télécom-Paris
46, Rue Barrault, 75013 Paris, France
barros@elec.enst.fr

Abstract. This paper presents a new motion estimation architecture for large displacements. An efficient differential block recursive algorithm is used to orient searching for a match and save computation power. Multiresolution and multiprediction approaches accelerate the algorithm convergence. Data multiplexing and pipeline allow real-time processing for standard video frequencies such as CCIR 601.

1 Introduction

Motion compensation is well known to be useful to increase performance in image sequence processing as video data compression, TV standard conversion, temporal interpolation, filtering, etc [Limb-Murphy (1975)], [Barros (1993)]. Differential, transform and block matching based algorithms have been proposed and compared [Bergmann (1983), Bouthémy (1988)]. There are two important problems in motion estimation: computation power and I/O rate requirements. Because of the inherent complexity and the computation power requirements of the algorithms, only translation movements are generally considered.

Almost all the proposed motion estimation circuits are based on the block-matching algorithms [Komarek (1989)], [Kim-Maeng (1994)]. Full search for block-matching algorithms requires a power computation proportional to the square of the maximal displacement. Moreover, full search implies a very high I/O rate. Therefore, the architectures are limited to estimate small displacements.

In this paper, a new motion estimation architecture is presented. A significant feature of the proposed scheme is to estimate large displacements. A differential block recursive algorithm have been used [Sanson (1991)]. Unlike in the block-matching algorithms, searching for a match is not systematical, but oriented by local image gradient. The number of matches to be made is reduced and a circuit with moderate I/O rate is obtained. Then, access to an external memory may be performed directly, without restriction for displacement lengths. Data multiplexing and pipeline allow real-time processing for standard video frequencies such as CCIR 601. This architecture is suitable for image temporal interpolation and implements a half pixel precision vector estimation according to MPEG recommendation for image coding.

The text is organized as following: in section 2 motion estimation algorithms are briefly reviewed. The implemented motion estimation algorithm is explicited in section 3. Section 4 presents the proposed architecture. Discussion of this architecture and comparison with other approaches are seen in section 5. Finally, conclusions of this work are drawn in section 6.

2 Motion Estimation Algorithms

The principal idea of the motion estimation is that the movement in image sequence generates luminance changes in regions concerned by this movement. Then, this change of the luminance can be used to estimate the movement. It supposes that luminance of objets in the scene is invariant between two consecutive images.

Motion estimation may be described as a parameter estimation process. This process starts by the construction of motion models which describe the motion involved. The motion models are usually specified by a set of parameters called motion parameters. Many approaches are reported in the literature [Limb-Murphy (1975)], [Bergmann (1983), Bouthémy (1988)].

There are direct and indirect approaches. For the direct approach, the 2-D motion parameters are estimated directly from an image sequence. Differently, in the indirect approach, a first determination of an optical flow is carried out. The model parameters are extracted from this flow.

With respect of the models, the movements are classified in translational and complex. Most often, only translation movements are considered [Musmann et al (1985)]. The motion estimators may be differential, block matching or transform based.

Transform based algorithms estimate a movement of an object in front of an uniform background by using the following property of the Fourier transform: a change in the space domain causes a change in the phase in the transform domain. The phase difference evaluated in two fields gives the motion vector. The disavantage of this approach is that only one movement by field is evaluated.

Block-matching algorithms consist to find the best block estimator in the previous frame (*t-1*) according to a matching criterion. The image is splitted in blocks and, for each block in the image at time *t*, it is tried to find the best match small enough to satisfy the assumption of translational motion. The intensity level of pixels in the block is being matched (Figure 1).

The *differential methods* include any method which makes use of the spatial and temporal differentials of an image. It usually depends on the Taylor series expansion of an image, which is not necessarily restricted to the first order. This is the chosen method for the implementation proposed in this paper and it will be discussed in the next section.

3 The algorithm description

The architecture for the motion estimator in this paper is based on the algorithm proposed in [Sanson (1991)]. This is a block recursive differential algorithm based on multiprediction and hierarchical multiresolution approaches. These concepts are discussed below.

3.1 Recursive differential method

Differential method includes are based on the Taylor series expansion of the spatio-temporal differences. The Taylor series and the motion model define a constraint which relates the locally available information luminance of pixels and the local motion.

It is assumed that the change in luminance of a pixel is only due to motion and that the luminance funtion is linear. So, only small displacements may be measured. To overcome this problem, recursive approaches have been developed [Boroczky et al (1990)]. In recursive algorithms, it is assumed that an initial estimate d^K for the real displacement d is available. This prediction d^K is used to produce a new estimate d^{K+1}. That is, the Taylor series expansion is iteratively used to update a motion displacement prediction. In the *pixel-recursive method*, an initial estimate, usually obtained from the neighbouring pixels, is updated by the information available from the local constraint. The estimate can only be updated in the direction of the image gradient.

The only information available from a single pixel is the velocity component in the direction of the image gradient at that point. So, information from different pixels have to be combined in order to obtain the total velocity. It can be done by applying the constraint to a block of pixels. This carries to a measure of weighted average motion of the block. This is the *block-recursive method* (Figure 2). Note that, as in the block-matching methods, in differential methods, the intensity level of individual pixels or a collection of pixels is what is being matched. However, searching for a match is directed by local image gradient. For the implemented algorithm, the updating of the displacement vector $\boldsymbol{d} = (d_x, d_y)$ is carried out by calculating:

$$d^{K+1} = d^K - \frac{DFD(x,y,d^K)\,G(x,y,d^K)}{\sum_{x,y \in block} [G(x,y,d^K)]^2} \qquad (1)$$

where:

- $DFD(x,y,\boldsymbol{d})$ is the image temporal differential and corresponds to the difference between luminances in current image and previous image displaced by $\boldsymbol{d}$
- $G(x,y,\boldsymbol{d})$ is the spatial differential evaluated in the reference image displaced by $\boldsymbol{d}$
- K exponent indicates the K^{th} iteration
- Σ indicates that temporal and spatial differentials are evaluated for all the pixels in the block
- spatial and temporal differentials are approximated by spatial and frame differences, respectively.

3.2 Hierarchical multiresolution approach

Hierarchical motion estimation is a technique which allows acceleration for algorithm convergence. The idea is to split the image in large regions (large blocks called "father blocks") and to perform the motion estimation over these regions. After this processing, the image is splitted again in smaller regions (smaller blocks called "son blocks"). The motion estimator is applied again, enhancing the estimation. The previously obtained results are used as initial predictions (Figure 3).

To improve the convergence speed, multiresolution may be associated to a hierarchical approach. This is shown in Figure 4. The image is subsampled and a pyramid is generated. Motion estimation begins at the top level of the pyramid. The results obtained at a level *l* are used as initial predictions for the *l-1* level. This allows to quickly estimate large displacements. The number of resolution levels is a parameter which may be adapted to a specific application. Vectors from level *l* need to be multiplied by two before to be used in level *l-1* to compensate the subsampling of the image.

3.3 Multipredictions approach

Problems may occur because of the image segmentation in blocks and multiresolution. For example, prediction obtained at level *l* may include the motion of more than one object. Also, bad predictions should be obtained if the object have been splitted through the segmentation.

These problems are minimized by using a multiprediction scheme. For each block at level *l*, five initial predictions are used: four predictions are those corresponding to the neighbour blocks in the uplevel *l+1* (Figure 5).

Note that the split of the image is considered in a quadtree with overlap. A fifth prediction corresponding to zero value for motion vector allows a reset of the algorithm.

3.4 Temporal continuity

In the original version of the algorithm, processing at the pyramid toplevel supposes only one initial prediction. This prediction corresponds to a zero displacement vector. Nevertheless, other predictions should be easily added. Taken into account values for displacements in other frames enhances the performance of the algorithm and accelerate the convergence.

In the version implemented with the proposed architecture, the initial predictions for a block at the toplevel at time *t* are (Figure 6): the equivalent block in the same position at the time *t-1*, the up neighbour of this block, the up-right neighbour of this block, and the right neighbour of this block.or the four vectors obtained at the time *t-1* in a level with more resolution.

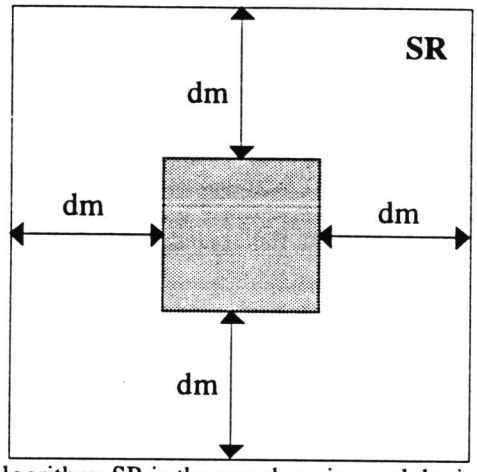

Figure 1: Block matching algorithm: SR is the search region and dm is the maximal displacement.

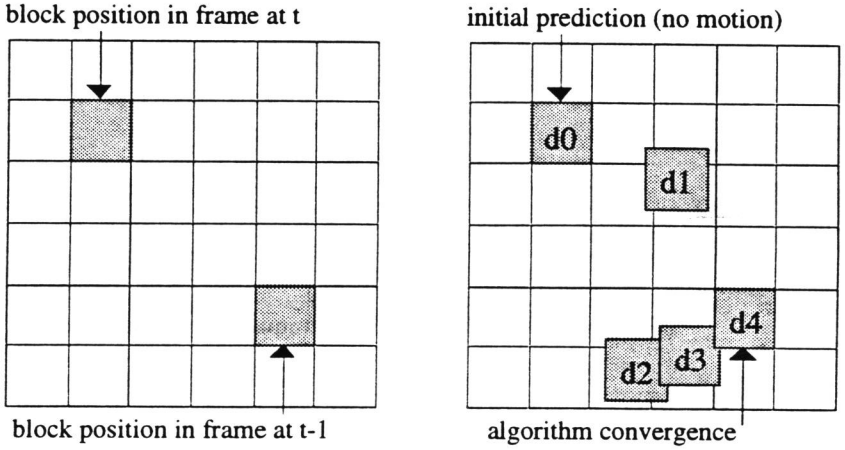

Figure 2: Block recursive motion estimation.

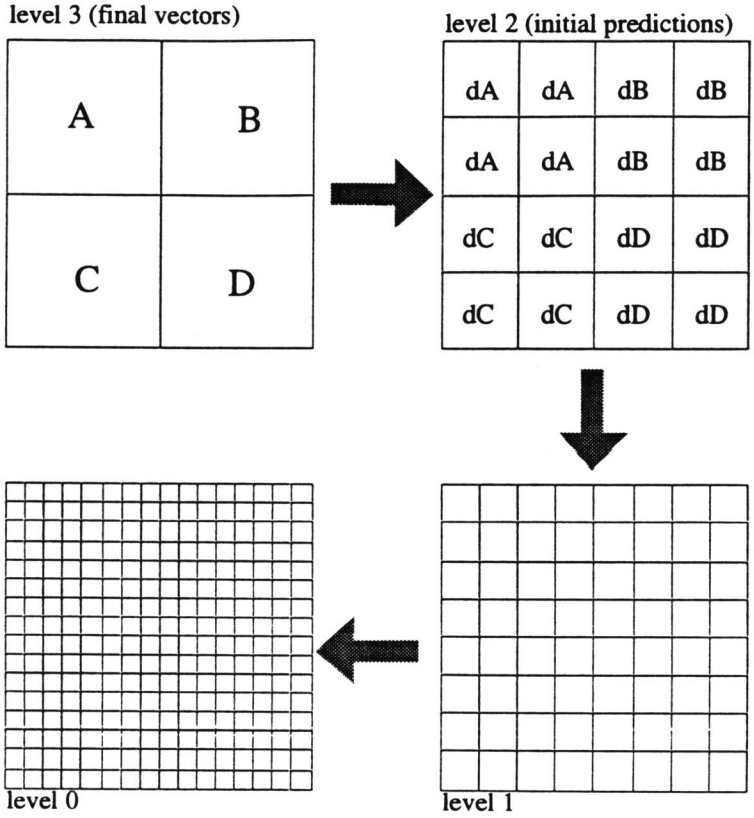

Figure 3: Hierarchical motion estimation: initial predictions for level *l-1* are the obtained vectors at level *l*.

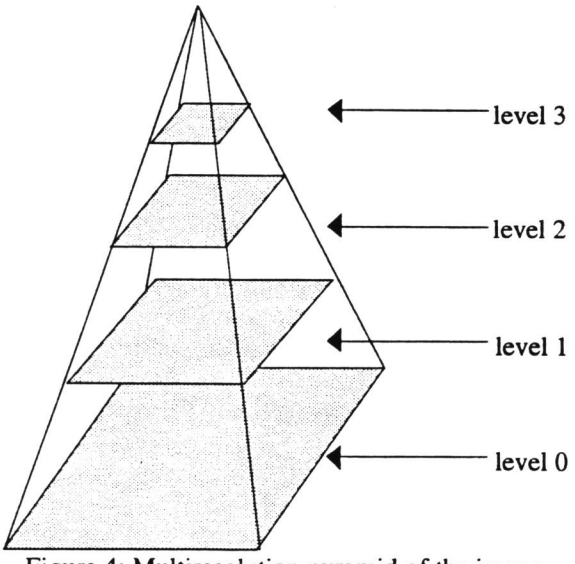

Figure 4: Multiresolution pyramid of the image.

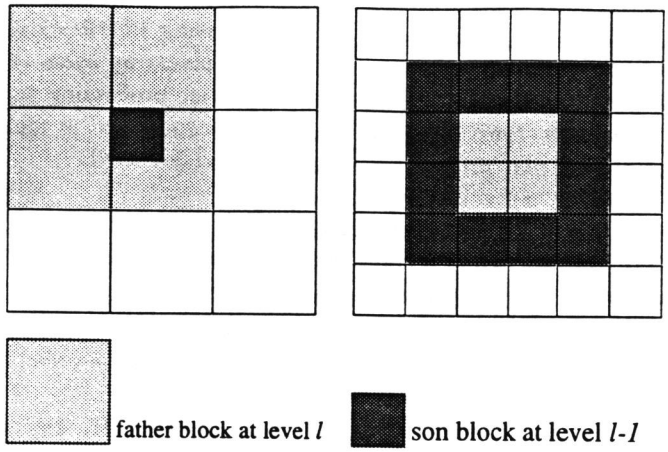

Figure 5: Quadtree with overlap. A "son block" at level *l-1* has 4 "father blocks" at level l. A "father block" at level *l* has 16 "son blocks" at level *l-1*.

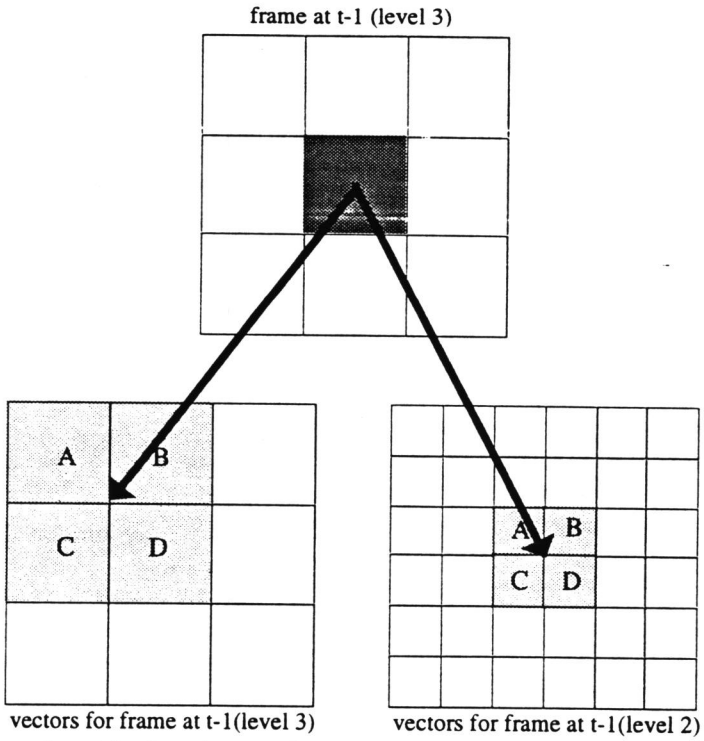

Figure 6: Initial predictions for a block at time t at level 3 (toplevel) take into account vectors obtained at time *t-1* at level 3 or level 2.

4 The proposed architecture

Figure 7 gives the general scheme of the motion estimator system. It supposes four levels in the pyramid and two iterations per prediction. The simulations performed have shown that it carries to the algorithm convergence. The full resolution image corresponds to the level 0. The processing is completly carried out at level 3 before to begin the processing on level 2 and so on. There are five initial predictions for each block. This processing is carried out *Ni* times for each initial prediction, where *Ni* is the number of iterations.

In the proposed architecture, concurrence and parallelism of the algorithm are exploited. Pipeline of operating in ARITHMETIC UNIT enables a real time work for standard video frequencies. Because this is a recursive algorithm, the pipeline between PRODUCTS and UPDATING units is performed by multiplexing initial predictions. During the processing of a prediction by UPDATING, PRODUCTS realizes the processing reporting to another initial prediction.

4.1 PIXELS CONTROL

This unit receives the pixels from the external memories containing the blocks in the frames *t* and *t-1* for several resolutions and gives them to the ARITHMETIC UNIT. PIXELS CONTROL contains three block memories to allow multiplexing and saving for I/O rate. Therefore, this unit provides parallel format for the pixels (8 pixels/cycle) to the ARITHMETIC UNIT and those neighbours useful for interpolating.

4.2 ARITHMETIC UNIT

The ARITHMETIC UNIT computes the equation (1). That is, it evaluates a new motion vector $\boldsymbol{d}^{K+1}$ for a given prediction $\boldsymbol{d}^{K}$. It is subdivided in two blocks: PRODUCTS and UPDATING units.

4.2.1 PRODUCTS UNIT

This unit receives the prediction vector displacement $\boldsymbol{d}^{K}$ and evaluates

- $G_x(x,y,\boldsymbol{d})DFD(x,y,\boldsymbol{d})$
- $G_y(x,y,\boldsymbol{d})DFD(x,y,\boldsymbol{d})$
- $[G_x(x,y,\boldsymbol{d})]^2$
- $[G_y(x,y,\boldsymbol{d})]^2$
- $DFD(x,y,\boldsymbol{d})$

for all pixels in the block. For non integer displacement values, an interpolation is performed by using a bilinear filter. As shown in Figure 8, the processing in PRODUCTS UNIT is carried out for eight pixels by cycle. These values are accumulated. So, after processing for all pixels in the block, the values contained in the registers are delivered to the UPDATING UNIT.

4.2.2 UPDATING UNIT

UPDATING receives the accumulated values of PRODUCTS and, once per block, carries out the division and the addition in equation (1), giving the value $\boldsymbol{d}^{k+1}$ of the vector displacement.

4.3 CONTROL UNIT

CONTROL UNIT is the estimator manager. It selects the best vector to associate to a given block by holding the vector corresponding to the smallest accumulated *DFD*. For that, CONTROL UNIT contains registers to stock intermediate values of displacement vectors, multiplexers and comparators.

5 Architecture considerations and discuss

Nevertheless the evaluation of equation (1) represents more complex operations when compared to those for one vector test for block-matching based estimators, the gradient approach is globally less expensive. An overall reduction of the power computation and I/O rate because less vectors are tested (Table 1).

In fact, simulations have shown that using a four levels pyramid and two iterations per prediction the algorithm convergence is obtained. For these conditions and usual maximal displacements (bigger than 5 pixels) the power computation and the I/O requierements for the processor presented in this work are less expensive as seen in Table 2.

For TV video type applications, the frequency requirements are:

Fp (the source pixel rate) = 13.5 Mpixels/s,

Fi (the vector iteration rate) = 180 x 10^6 evaluations of products and accumulations for equation (1) per second.

Because of the value of *Fi*, a parallelisation approach has been used. In the proposed architecture, operators are multiplied to carry out eight *DFD* and gradient operations simultaneously in PRODUCTS. This allows real-time operation with a chip frequency *Fc* = 22 MHz, moderating by the way power consumption. On the other hand, in the UPDATING unit no parallelism is necessary. Furthermore, the motion vectors may be delivered in bit serial format, saving I/O number of pads. Table 3 summarizes some comparisons with other implementations for motion estimation.

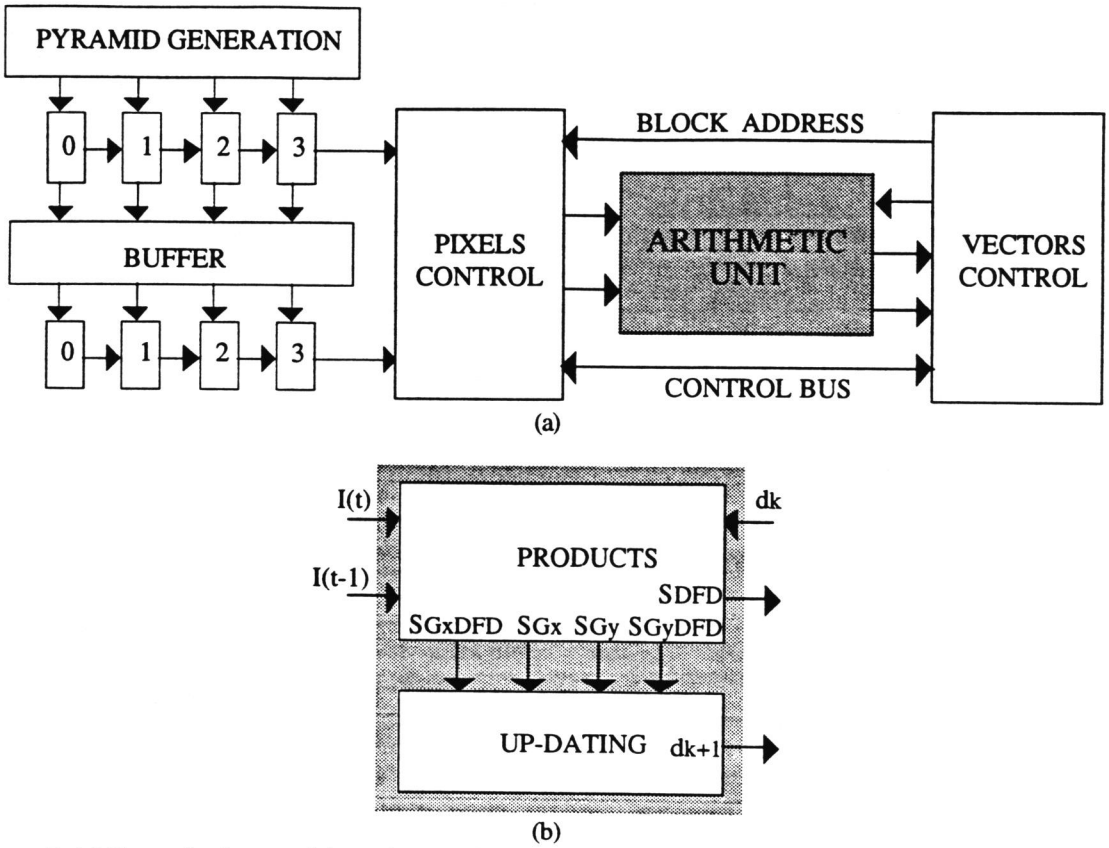

Figure. 7. (a) General scheme of the estimator. (b) Detail for ARITHMETIC UNIT: processing is pipelined.

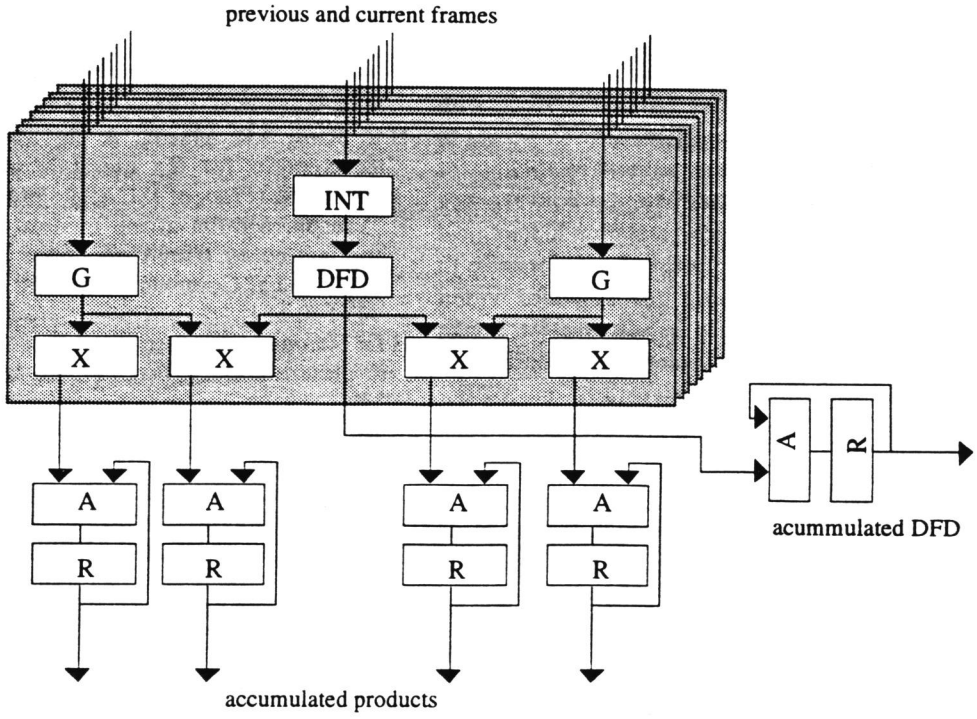

Figure 8. Internal scheme for PRODUCTS.

Algorithms Complexity	Number of Matches	Operations for each Match	Total Number of Operations
block-matching	$(2dm+1)^2$	1 interpolation + 1 DFD	$4\,(2dm+1)^2$
differential	Nix 5 x 4/3	1 interpolation + 1 DFD + 2 gradients + 4 multiplications	254 Ni

Table 1: Power computation for full-search block-matching and differential based algorithms. The results are given in 8 bits addition per pixel.

Differential	Block-Matching		
$\forall dm$, $Ni = 2$	$dm = 8$	$dm = 16$	$dm = 32$
508	1156	4346	16900

Table 2: Total number of 8 bits additions for usual displacements.

Circuit	A (mm^2)	Technology	Fc	Nb of chips	dm	Comments
COLAVIN91	85*	1,2 µm	13.5 MHz	3	-16/+15	full-custom
URAMOTO93	125**	0,8 µm	40 MHz	1	-16/+15	half-pel precision
PROPOSED	80***	0,8 µm	22 MHz	1	-128/+127	half-pel precision

Table 3: Comparisons of some block matching circuits and proposed motion estimator architecture. These values are obtained for real-time operation with 16x16 pixel block and CCIR 601 resolution. *[Colavin (1991)], **[Uramoto et al (1993)] and ***estimated.

6. Conclusions

In this paper, a new motion estimator architecture have been presented. Unlike in the block-matching algorithms, searching for a match is not systematical, but oriented by local image gradient. The number of matches to be performed is reduced and a circuit with moderate I/O rate is obtained. Then, directly access to an external memory is possible, without restriction for displacement lengths. To improve the convergence speed, the algorithm uses a multiresolution hierarchical approach. Multiplexing and pipeline are necessaries to allow a real time processing for standard video frequencies. This architecture is suitable for image temporal interpolation and implements a half pixel precision vector estimation according to MPEG recommendation for image coding.

7 References

J.O. Limb and J.A. Murphy, Measuring the speed of moving objects from television signals, IEEE Trans. on Communications (1975) 474-478.

L.A. Barros, Architecture intégrée pour le ré-échantillonnage d'images Animées, Technical Report, ENST (1993).

H.C. Bergmann, Analysis of different displacement estimation algorithms for digital television signals, in Image Sequence Processing and Dynamic Scene Analysis, Ed. T.S. Huang (1983) 215-233.

P. Bouthémy, Modèles et méthodes pour l'analyse du mouvement dans une séquence d'images, Technique et Sciences Informatiques, (1988) 527-545.

T. Komarek and P. Pirsch, Array architectures for block matching algorithms, IEEE Trans. on Circuits and Systems (1989) 1301-1308.s

H.C. Kim, S.R. Maeng, A pipelined systolic arrays architecture for the hierarchical block-matching algorithm, (Proc. of ISCASS 1994) 4221-4224.

H. Sanson, Motion affine models identification and application to television image coding, (Proc. of SPIE Visual Communication and Image Processing '91) 570-581.

H.G. Musmann, P. Pirsch, and H-J. Grallert, Advances in picture coding, Proc. of the IEEE (1985), 523-548.

L. Boroczky, K. Fazekas, and T. Szabados, Analysis of a pel-recursive Wiener-based motion estimation algorithms for general 2D motion, Signal Processing V (1990) 785-788.

O. Colavin, A. Artieri, J-F. Naviner, and R. Pacalet, A dedicated circuit for real time motion estimation (Proc. of the EUROASIC 91) 2453-2456.

S-I Uramoto, A. Takabatake, M. Suzuki, H. Sakurai and M. Yoshimoto, A half-pel precision motion estimation processor for NTSC-resolution video, (Proc. of IEEE Custom Integrated Circuits Conference 1993) 11.2.1-11.2.4.