Fast Hardware Implementation of Gabor Filter Based Motion Estimation

A. SPINEI[1], D. PELLERIN[1,2], D. FERNANDES[1], J. HERAULT[1,2]

[(1)]Laboratoire des Images et des Signaux, Institut National Polytechnique, 46 Av. Félix Viallet, 38031 Grenoble Cedex, France

[(2)]Institut des Sciences et Techniques, Université Joseph Fourier, Grenoble, France

E-mail: pellerin@lis-viallet.inpg.fr

ABSTRACT: Motion estimation in image sequences is a fundamental issue in many applications as for instance in artificial vision and three-dimensional scene reconstruction. Among all the existing techniques, we are particularly interested in methods using Gabor filters, which are known to furnish quality results but usually require intensive calculation. A new, fast energy-based method is presented, which combines in a direct manner the energetic responses of Gabor spatio-temporal filters organized in triads. An implementation of this technique on a general purpose Digital Signal Processor (DSP) board is described and the advantages compared with Very Large Scale Integration (VLSI) and parallel machine approaches. Our hardware implementation attains a reasonably fast output rate (several images/second) for a better resolution than in the most recent VLSI implementations. These results open interesting perspectives for real-time implementations (such as in mobile robotics) and for the obtention of higher-level results by combining different Gabor filter techniques (for motion, edge detection, texture analysis, etc.).

INTRODUCTION

Image filtering using oriented pass-band Gabor wavelets is a well-known powerful tool in the field of image processing, used in various applications such as: edge detection (Merhotra et al., 1992), texture classification and segmentation (Manjunah and Ma, 1996), (Dunn et al., 1994), object detection and recognition (Wu and Bhanu, 1997), motion estimation (Heeger, 1987), (Fleet and Jepson, 1990). Some of these techniques are inspired by research concerning the vision of primates, a fast-developing domain which could lead in the future to new image processing architectures able to execute higher level tasks. These methods generally lead to good results, but at the price of complex algorithms and high computing

power requirements.

In this paper, we examine the problem of speed estimation in the image sequences, using Gabor filters implemented in a fast, novel approach. Our main motivation is that motion information extracted from a sequence of time-varying images plays a key role in image understanding and in the comprehension and perception of a scene. Motion analysis has inspired a lot of scientific work (Mitiche and Bouthemy, 1996) due to the high economic stakes involved, the number and the importance of the applications (multimedia, medicine, automated surveillance, etc.) but current hardware implementations are too slow and computing power too demanding or they fail to respond to high quality standards required by complex tasks. Therefore, real-time, cost-effective implementations of Gabor filtering techniques for motion analysis are usually not considered as viable.

The article presents a new method for speed estimation using triads of Gabor filters and its implementation on a general purpose Digital Signal Processor (DSP) board. The principles of energy-based methods for motion estimation are explained. We present our algorithm, its complexity and some simulation results. We enumerate the advantages of our DSP implementation compared to Very Large Scale Integration (VLSI) and parallel machine implementations. Furthermore, the architecture of our hardware implementation is described, along with some illustrative results.

PRINCIPLE OF THE ENERGY-BASED METHODS

In the last 20 years, a large number of methods for motion estimation have been proposed, which can be classified into three main categories: differential techniques, region-based and energy-based methods (Barron et al., 1992) and (Barron et al., 1994). The differential methods, which are the oldest, use the local spatio-temporal variation of luminance. They are based on the gradient equation constraint, which joins the temporal and spatial derivatives to the two speed components. The results obtained with these techniques are generally degraded if the image sequence presents spatio-temporal noise. The block-matching techniques consist in identifying the corresponding patterns (with or without actual significance) in successive images. These techniques are well-adapted to high amplitude translations and offer neither the required precision for slow speeds (less than 1 pixel/image) nor a dense optic flow. Finally, the energy-based methods, relatively recent and less well-studied, are based on motion analysis in the frequential domain. We have chosen these methods because they can be easily implemented in a homogenous and simple manner using successive filtering, offering good quality results and a natural physical interpretation.

Energy-based methods for velocity estimation in image sequences are based on the fact that motion induces specific energy patterns in the frequential domain (Adelson and Bergen, 1985). The power spectrum, associated with an image $i(x, y)$ in translation with a uniform velocity $\vec{v} = (v_x, v_y)$ occupies a plane in the spatio-temporal frequency domain whose equation is: $f_t + v_x f_x + v_y f_y = 0$ where f_x, f_y represent

spatial frequencies and f_t the temporal frequency. The principle of energy-based methods consists in localizing the non-zero energy plane or velocity plane using oriented spatio-temporal filters, in order to directly obtain the components of the velocity vector (v_x, v_y).

A few energy-based methods using spatio-temporal Gabor filters are described in the literature. The first such method belongs to (Heeger, 1987), with an extension proposed by (Spinéi et al., 1998a) which allows a great improvement of the results in the case of a fixed camera. In both cases, motion estimation is computed as a minimization of the quadratic distance between the measured filters' responses and the predicted responses to a white noise translating texture. The required computating time is prohibitive and is highly dependent upon the estimation precision desired. Finally, the principle of a more general architecture is presented by (Grzywacz and Yuille, 1990), but the number of filters required is prohibitive for actual implementation.

MOTION ESTIMATION ALGORITHM

Spatio-temporal pass-band filters such as Gabor energy filters are not selective to velocities, but rather are tuned to particular spatio-temporal frequencies: therefore, on their own, they do not allow velocity estimation. We shall demonstrate an effective way at combining their responses in order to estimate the motion (Spinéi, 1998).

Gabor Energy Filters

A cosine (or even) phase 3D Gabor filter is represented as a spatio-temporal Gaussian window multiplied by a cosine wave:

$$G_c(x,y,t) = \cos\left[2\pi\left(f_{x_0}x + f_{y_0}y + f_{t_0}t\right)\right] \frac{1}{\sqrt{2\pi}^3 \sigma_x \sigma_y \sigma_t} \exp\left[-\left(\frac{x^2}{2{\sigma_x}^2} + \frac{y^2}{2{\sigma_y}^2} + \frac{t^2}{2{\sigma_t}^2}\right)\right]$$

where $(f_{x_0}, f_{y_0}, f_{t_0})$ is the central frequency of the filter and $(\sigma_x, \sigma_y, \sigma_t)$ is the standard deviation of the spatio-temporal Gaussian window. The sum of the squared output of the cosine-phase filter and the squared output of the sine-phase filter gives a measure of energy that is independent of the phase of the signal. The energy response to a moving random texture of such a 3D Gabor filter (Heeger, 1987) is:

$$R_{f_{t_0}} = \frac{k^2}{8\pi\sqrt{\alpha}} exp\left[-4\pi^2\sigma_x^2\sigma_y^2\sigma_t^2 \frac{\left(v_x f_{x_0} + v_y f_{y_0} + f_{t_0}\right)^2}{\alpha}\right]$$

with $\alpha = (v_x\sigma_x\sigma_t)^2 + (v_y\sigma_y\sigma_t)^2 + (\sigma_x\sigma_y)^2$ and k a normalization constant (the energy due to the local image contrast).

1D and 2D Motion Estimations

For simplicity, we shall first examine the case of a 1D movement into a monodimensional "image". In this case, we will use 2D Gabor filters whose energetic response is:

$$R_{f_{t_0}} = \frac{k^2}{4\sqrt{\pi}\sqrt{\sigma_x^2+(v_x\sigma_t)^2}} exp\left[-4\pi^2\sigma_x^2 \frac{(v_x f_{x_0}+f_{t_0})^2}{{v_x}^2+\left(\frac{\sigma_x}{\sigma_t}\right)^2}\right]$$

The essential issue of our approach (Spinéi et al., 1998b) is that the speed v_x is computed directly using the energy responses of three Gabor filters (a triad) with identical bandwidths σ_x and σ_y, centered on the same spatial frequency $f_{x_0} = 0.25$ but placed on three different temporal frequencies $f_{t_0} = -0.25; 0$ and 0.25 (Figure 1(a)). This allows us to avoid the time-expensive quadratic minimisation phase.

$$v_x = \frac{1}{6}\frac{L_{-0.25} - L_{0.25}}{L_0} \tag{1}$$

with the notation :

$$L_{f_{t_0}} = \ln R_{f_{t_0}} - \frac{\ln R_{-0.25}+\ln R_0+\ln R_{0.25}}{3}.$$

The 1D estimation is interesting in itself because some low cost robots are equipped with a linear 1D camera. We made several successful tests which proved that a 1D velocity estimation could be achieved using a unique triad of 2D Gabor filters. Robots using such a technique should be able to react and orient themselves in specially designed environments.

In the case of a 2D movement, two speed components are theoretically sufficient. As a matter of fact, the energy spectrum is not constant all over the motion plane, because the local contrast is different according to the direction. This could introduce estimation errors. For this reason we choose to estimate the 1D speed along four directions by using four triads of Gabor filters following the spatial orientations $\phi = 0, \pi/4, \pi/2$ and $3\pi/4$ (Figure 1(b)). Equation (1), which is used to compute the 1D speed in the case of a triad of 2D Gabor filters, is still valid in the case of a triad of 3D Gabor filters placed at the orientation $\phi = 0$ or $\phi = \pi/2$. The right member of the above equation is weighted by a factor of $\sqrt{2}$ in the case of the triads placed at $\phi = \pi/4$ and $\phi = 3\pi/4$.

The final motion vector is a combination of the four 1D estimations (the simplest way is to compute the mean value along two orthogonal directions as in Figure 1(c)). It is possible to use the values of the four components to detect the particular situation of edge motion (also called the aperture problem). In this case, the affixes of the four estimated vectors lie on a straight line which is parallel to the moving contour (Figure 1(d)) (Torralba and Hérault, 1997)).

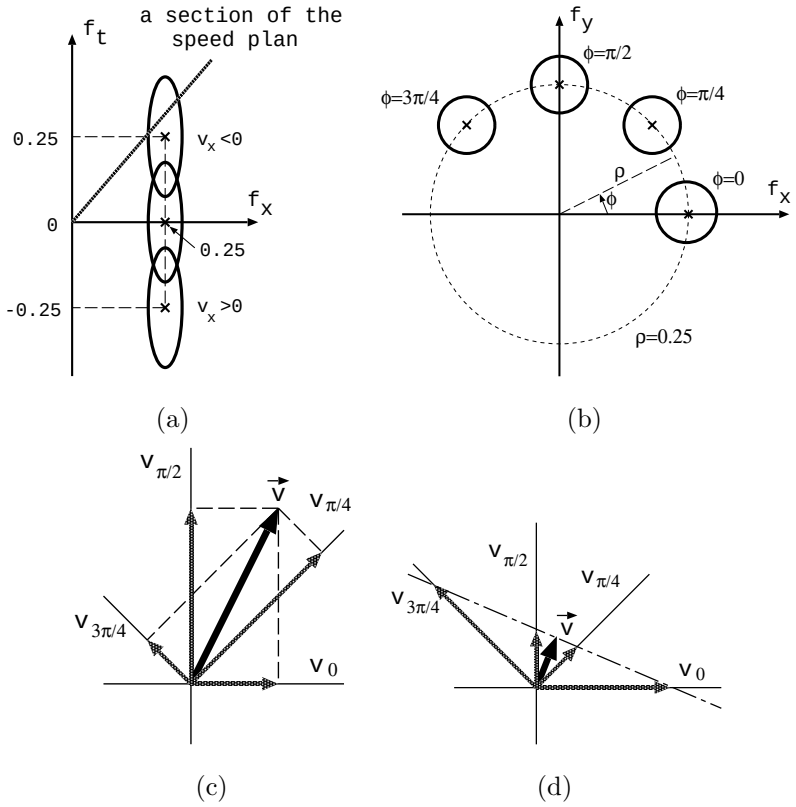

Figure 1 (a) A triad of Gabor filters in the plane (f_x, f_t), (b) four triads of 3D Gabor filters in the plane (f_x, f_y), (c) the 2D motion vector $\vec{v}$ as a mean value of the four components or (d) in the case of edge motion.

Algorithm Synthesis

The final filtering architecture consists of 12 Gabor filters, tuned to the spectral domain as described in (Heeger, 1987). The filters are organized in a particular way as they are divided into 4 triads. The choice of the position and the parameters of these filters is determined by the range of velocities desired. The filters are tuned to different spatial and temporal frequencies which will determine in a direct manner the range of valid velocity estimation. As a direct consequence, the estimation of rapid movements will imply filtering with Gabor filters tuned to a very low spatial frequency. These filters must have a large spatial standard deviation to avoid aliasing problems (which is equivalent to filtering at a superior level of the spatial pyramid). Therefore, a large range of velocities implies a loss of precision in the localization of moving regions. However, this problem can be overcome by estimating each velocity within the appropriate pyramid level (lower pyramid levels for lower speeds, higher pyramid levels for higher speeds).

Moreover, the choice of the standard deviations σ_t and $\sigma_s = \sigma_x = \sigma_y$ is a complex and important problem. The optimal values depend on a few parameters: the filter position, size of the speed domain and also, to a certain extent, the speed and the size of the moving object. In order to obtain an equivalent

performance to the human visual system on textured images with different contrasts according to the directions, Heeger merely indicates that the standard deviation σ_s must be greater than σ_t. He suggests the values $\sigma_t = 1$ and $\sigma_s = 4$. It is not possible to satisfy all the optimization criteria, so an empirical compromise is usually employed. We found that the value $\sigma_t = 1$ offers a good coverage of the frequential temporal domain if the three filters are tuned to the temporal frequencies $-0.25, 0$ and 0.25. The choice of the optimal value for the spatial bandwidth σ_s is influenced by two constraints, which represents the following dilemma:

- for larger spatial standard deviations, we obtain speed estimation of good quality inside the moving objects, but the smoothing phenomenon on motion boundaries is great.
- for smaller spatial standard deviations, the speed estimation within the moving objects deteriorates but the smoothing phenomenon is lesser.

We determined experimentally that a satisfactory compromise appears for σ_s in the range of 3.5 and 4.5 (we used images of size 128×128 and 256×256 pixels).

The hypothesis of white spectrum is rarely satisfied in the case of real images. In (Atick and Redlich, 1992), it is reported that the mean spectra of real-world natural images is of $\frac{1}{f}$ type and that a compensating whitening filter has a great resemblance to primary retinal pre-filtering. According to this hypothesis, our method is composed of three fundamental steps (Figure 2) which compares well with the physiological model of visual processing in the retina and the visual areas V1 and MT (Stoner and Albright, 1993):

- Prefiltering in order to generate a smoother energy spectrum (whitening) and to reduce the lower frequencies. This step is carried out using a pass-band spatial DOG filter (Difference Of Gaussians).
- Extracting the energies induced by the motion plane into a bank of oriented spatio-temporal filters and computing the 1D velocity estimation by combining the answers of three such filters (a triad).
- Estimating the 2D velocity by associating four triads.

Algorithmical Complexity

The algorithmic complexity is essentially due to the great computing power needed by the twelve 3D Gabor energy filters. We shall first explain how to decompose the three-dimensional filters into a combination of 1D filters and how to reduce the number of 1D filters necessary. Next, we show how a recursive implementation of 1D spatial filters accelerates the computing time.

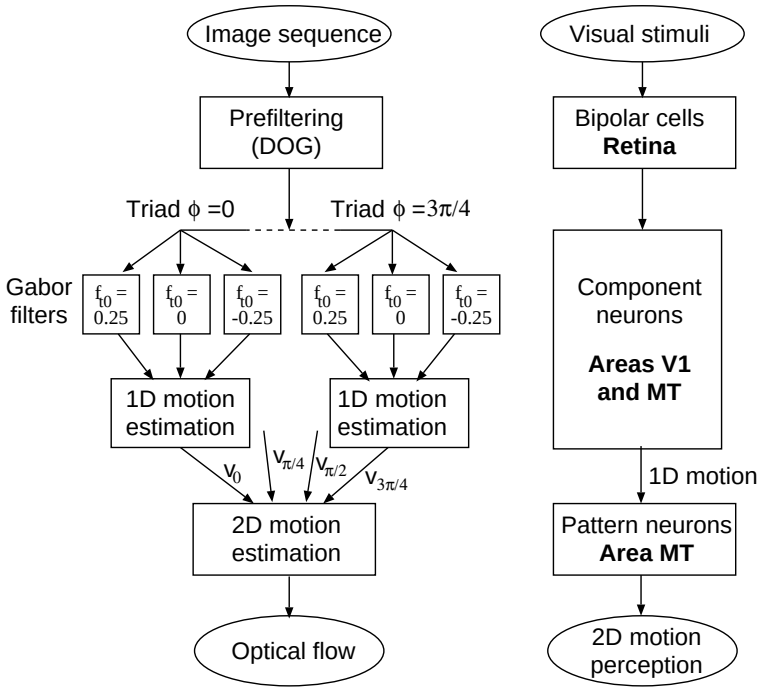

Figure 2 Comparing the model with physiology.

Amount of 1D Gabor Filters

A pair of Gabor 3D filters (sinus and cosinus phase) implemented by separable convolution requires the calculation of 8 terms which are the 8 possible combinations of 1D convolutions with sinus and cosinus phase over t, x and y:

$$\begin{aligned}\cos(f_{t_0}t + f_{x_0}x + f_{y_0}y) = \\ &- \sin(f_{t_0}t)\cos(f_{x_0}x)\sin(f_{y_0}y) \\ &- \sin(f_{t_0}t)\sin(f_{x_0}x)\cos(f_{y_0}y) \\ &- \cos(f_{t_0}t)\sin(f_{x_0}x)\sin(f_{y_0}y) \\ &+ \cos(f_{t_0}t)\cos(f_{x_0}x)\cos(f_{y_0}y) \qquad (2)\end{aligned}$$

$$\begin{aligned}\sin(f_{t_0}t + f_{x_0}x + f_{y_0}y) = \\ &\sin(f_{t_0}t)\cos(f_{x_0}x)\cos(f_{y_0}y) \\ &- \sin(f_{t_0}t)\sin(f_{x_0}x)\sin(f_{y_0}y) \\ &+ \cos(f_{t_0}t)\sin(f_{x_0}x)\cos(f_{y_0}y) \\ &+ \cos(f_{t_0}t)\cos(f_{x_0}x)\sin(f_{y_0}y) \qquad (3)\end{aligned}$$

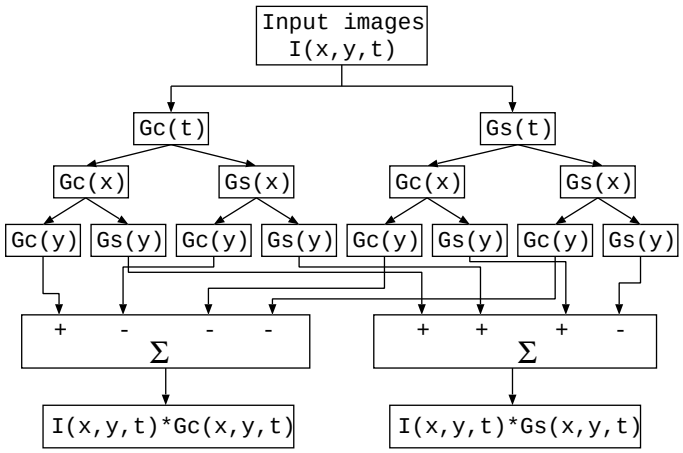

Figure 3 Computing the result of a 3D Gabor filtering through separable convolution with sinus and cosinus 1D Gabor filters (Gs and Gc) over the three dimensions x,y and t.

The number of convolutions required for a pair of filters is 2 over the temporal dimension t, $2 \times 2 = 4$ over x and $4 \times 2 = 8$ over y, which means 2 temporal filtering and 12 spatial filtering. The final result for the 3D Gabor filters (sinus, respective cosinus phase) is obtained by the algebraic addition of the four corresponding terms (Figure 3).

The four triads contain 12 pairs of Gabor filters. The number of convolutions required for the calculation of those filters using the "trivial" approach is of $2 \times 12 = 24$ temporal filtering operations and $12 \times 12 = 144$ spatial filtering operations. However, the number of 1D filters necessary could be reduced by re-using the intermediate results. As an example, the temporal convolutions are identical for all four triads, meaning that the temporal filtering operations could be done only once for the first triad and reused for the three others. The temporal and spatial symmetries between the terms of the equations (2) and (3) could be exploited using some simplifications based on elementary trigonometrical equalities. We found that the temporal filtering for $f_{t_0} = -0.25$ is easily deduced from the filtering accomplished at $f_{t_0} = +0.25$ and that for the temporal filter tuned to $f_{t_0} = 0$, only the cosinus filtering is useful. Therefore, only 3 temporal filters are necessary for each sequence processed. In a similar way, we reduced the number of spatial filtering operations used for the triads to $\phi = 0$ and $\phi = \pi/2$. Re-using the intermediate results for the triad at $\phi = \pi/4$ as part of the calculation for the triad at $\phi = 3\pi/4$ reduces it even further. Finally, the number of 1D filtering operations is of 3 temporal filtering operations and 30 spatial filtering operations.

Implementation of Recursive Filtering

We propose an efficient recursive implementation of the energy filters, inspired by (Unser, 1993). The output $o(n)$ of a 1D Gabor filter (expressed in a complex form) excited by an input signal $i(n)$ is written:

$$o(n) = i(n) \star \left[K \; exp\left(-\frac{n^2}{2\sigma^2}\right) \exp(2\pi j f_0 n) \right]$$

Or,

$$o(n) = \left([i(n) \exp(-2\pi j f_0 n)] \star K \exp\left[-\frac{n^2}{2\sigma^2}\right] \right) \exp(2\pi j f_0 n)$$

Thus, a Gabor filter could be broken down into three successive stages: modulation, low-pass filtering (Gaussian envelope) and demodulation (Figure 4).

Figure 4 Decomposition of Gabor filtering into three stages.

For the energy filters, the final demodulation stage is redundant. As a matter of fact, if $q(n)$ is the intermediate output before the final filtering stage, we can easily prove that:

$$|q(n)|^2 = |o(n)|^2$$

Gaussian low-pass filtering is carried out using a recursive filter, which offers the advantage of having a size that does not depend on the filter's selectivity. We chose third order filters, which have a small root mean square error ($RMSE = 0.15\%$ for $\sigma_s = 4$) when compared to a Gaussian filter. The coefficients of these filters were computed using the relations given in (Young et al., 1995). These filters, with their coefficients as a function of the selectivity, are expressed as a product between a causal and a noncausal term:

$$H(z) = \frac{A_2}{b_0 - b_1 z^{-1} - b_2 z^{-2} - b_3 z^{-3}} \times \frac{A_2}{b_0 - b_1 z^{1} - b_2 z^{2} - b_3 z^{3}}$$

The recursive implementation is very efficient in the spatial domain, where the size of the initial convolution filter is 23 coefficients. In the temporal domain, the recursive filtering is not a viable solution because of the noncausal requirements. Instead, we will use the initial convolution filter, which is reasonably small due to its narrow temporal bandwidth. In fact, for $\sigma_t = 1$, a 7 coefficients filter provides a $RMSE$ smaller than 1%.

Sequence name		Barron and al.	3 pyramid levels	Second pyramid level only
Translating tree	ANE	4.79^o	4.06^o	2.42^o
	SD	2.39^o	2.8^o	5.79^o
Diverging tree	ANE	4.95^o	4.6^o	8.28^o
	SD	3.09^o	3.5^o	6.79^o
Yosemite	ANE	11.97^o	14.9^o	14.95^o
	SD	19.0^o	17.3^o	12.6^o

Table 1 Comparison of motion estimation errors for three image sequences, for the three pyramid levels and for the second pyramid level only (as used in our DSP implementation). ANE=Average Normal Error, as defined by Barron, SD=Standard Dispersion of errors

Simulation Results

The software implementation uses recursive filtering and a pyramidal multi-resolution decomposition. In order to compare our results with those obtained by Barron in his reference paper (Barron et al., 1992), we processed three image sequences with known optical flow and we used a similar confidence measure which allows us to eliminate, via simple thresholding, the less reliable vectors. In order to compare computation time, we also implemented the estimation method through minimization as suggested by Barron (using a rough speed quantification for a coarse estimation, followed by a second minimization with a smoother resolution).

The estimation errors obtained using the two approaches are very close (c.f. table 1), but our method is much faster. For instance, the estimation time for an optical flow for images of size 128×128 pixels (using an UltraSparc 1 workstation) is of 8.4 seconds in our case, as opposed to approximately 184 seconds in the case of the estimation through minimization. In figure 5, we present the results for the image sequence "Yosemite".

In order to drastically reduce the computation volume, we computed a non-dense optical flow only at the second level of the spatial pyramid. In these conditions, computing time went down to 0.4 seconds. For the tested sequences, the results presented a good quality.

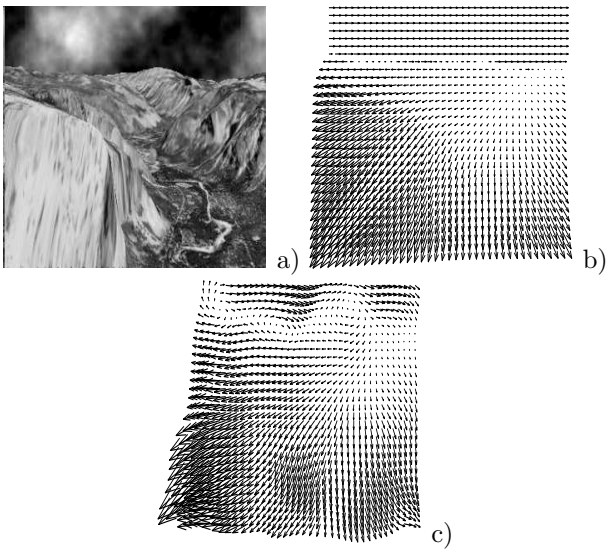

Figure 5 (a) An image from the "Yosemite" sequence, (b) real optical flow, (c) estimated optical flow.

DSP BOARD IMPLEMENTATION

Hardware Choice

Next, we examine different solutions that can be used for an implementation of our algorithm, and justify the choice that we have made.

Today, the most widely studied approaches are VLSI implementations and parallel implementations. VLSI implementations present some indisputable advantages: reduced computing time (especially in the case of analog circuits), small dimensions and ease of integration in real-world applications. However, certain reasons incited us to ignore this solution for the moment. The making of integrated circuits requires a very long testing phase and heavy financial investment. In our case we had two additional technical problems: the large amount of memory required by the algorithm and the difficulty of making a good 3D Gabor filter approximation on silicon (Shi, 1998).

The parallelism embedded in low-level vision processes can be exploited by certain parallel machines, usually Single Instruction Multiple Data (SIMD) architectures. Two such vision-targeted architectures have been reported, the CNAPS machine from Adaptive Solutions Inc. and the systolic/cellular array processors from Hughes Research Lab. Some specially designed architectures were designated for motion detection (Lee et al., 1993) and estimation using massively parallel neural networks (Zhou and Chellappa, 1988). The major problem of parallel machines is the actual paradigm which sees them more as a "calculation accelerator" linked to a host machine than as an independent system. Accordingly, parallel machines dedicated to vision processing are heavily penalized by the small video bandwidth between the video source and the computing device, inappropriate for real-time applications.

Taking into account the serious problems presented by VLSI and parallel machine set-ups, we chose to implement our algorithm on a DSP board. This approach offers great flexibility for shorter development time and lower hardware cost.

Hardware Implementation

The board was designed in our laboratory with the aim of providing a platform for real time dynamic video applications. The board associated with a camera and a B/W video monitor constitutes a complete processing system which includes all the processing steps from image sampling to result display. The development and testing of a new algorithm are done on a host PC. After the initialization (which could be done with a ROM memory in real applications) the board is totally autonomous.

The DSP card was designed for a maximum speed of 25 Hz, the real-time video framerate, with both static (edge detection using a Canny operator) and dynamic (motion detection using Markov fields) image processing applications. The second application was able to perform a simple detection of moving regions in the case of fixed camera, and it achieved a framerate of 12 images/second (Dumontier et al., 1996).

The architecture of the board is composed of 5 principal stages (Figure 6): an input stage for AD conversion (image size 512×512 pixels coded on 8 pixels), a preprocessing stage for elementary operations (Field Programmable Gate Arrays (FPGAs), non-interlaced odd frame of size 256×256 pixels), a processing stage for complex operations (Digital Signal Processor (DSP) with static memory (SRAM)), a buffer stage for asynchronous data accesses (Video RAM (VRAM) component), an output stage for DA conversion. The three main steps of our algorithm (DOG filtering, Gabor filtering and 1D estimation, 2D estimation: Figure 2) are carried out the DSP. We managed to obtain computing time reduction (about 20%) and memory reduction (16 Kbytes) by implementing the first low pass filtering for the pyramidal decomposition on one of the FPGAs included on the board. The remaining operations needed for the generation of the second spatial pyramid level, less intensive, are entirely the task of the DSP. Estimated speed values are resampled and rescaled before being written into the VRAM for conversion and visualization.

The board is built around a general purpose DSP Motorola 96002 clocked at 24 MHz (Figure 7), with a computing performance of about 12 MIPS and 36 MFLOPS. It contains a frame grabber and a video conversion module (Brooktree AD and DA converters), an FPGA (Xilinx XC4003) circuit as well as an interface circuit for the ISA bus allowing the programming and the testing via a PC. The memory available is composed of 32 Kwords of SRAM and of 2 banks of 256 Kbytes each of triple-port fast page mode VRAM. Two supplementary FPGAs implement the synchronization and the control logic. All these modules are synchronized by the video clock (fpixel=7.5MHz).

.

We had to respect certain strict hardware constraints, concerning especially the small amount of program memory available: only 1024 words of 32 bits, almost half of which is occupied by the initialization

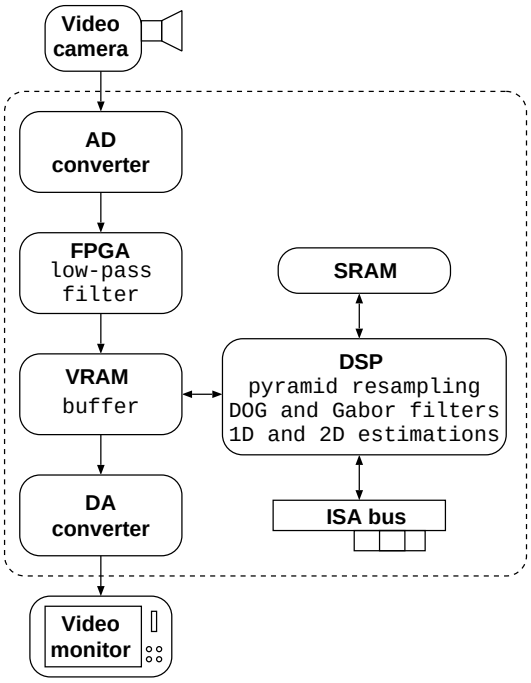

Figure 6 PC board block diagram

and video refresh subroutines. Though it is possible to use the extra but much slower external memory, we avoided this by minimizing the length of code in order to maximize processing speed.

Results

The test environment consists of a PC (with an i486 SX processor) which supervises the ISA board, a standard video input (CCD camera) functioning at 25 images/second and a B/W monitor to visualize directly the motion estimation. The results are shown separately for each triad, using a small image which represents in shades of grey the modulus of the estimated motion following the orientation considered.

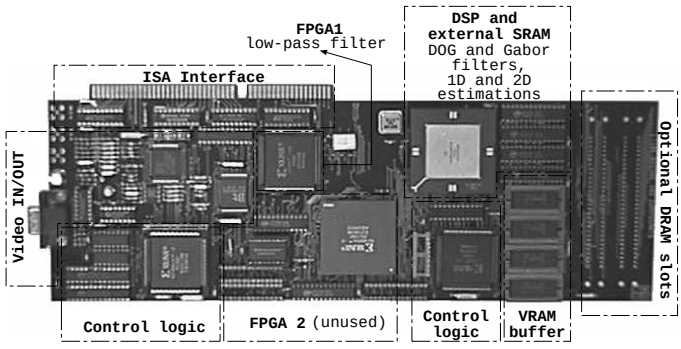

Figure 7 Photo of the PC board

The estimation is made in the speed domain of $[-5; 5]$ pixels/frame within images of size 128×128 pixels. The filtering is made at the pyramid level 2, therefore the optical flow consists of a matrix of 32×32 vectors (1 speed vector for a group of 4×4 pixels, using the second level of a spatial pyramid). This is a reasonable size for the resulting optical flow in order to limit memory needs and to produce a fast estimation framerate. Two demonstration programs are currently available: one of them computes the speed using four triads (approximately 3.3 estimations/second) and the other one uses only two triads (6 estimations/second with a minor degradation of the results).

When using 2 triads instead of four, the results are less accurate, but it could be enough for certain cases. As an example, it was suggested to us that a possible application would be to control traffic lights at an intersection according to vehicle flow. In this particular case, the approximated speed for each vehicle, along with its direction, are sufficient even if the vehicle is not very well localized or the vector field on the vehicle has small discontinuities.

A small sample of the results in the case of an estimation with two triads is shown in Figure 8. We directed the camera over a street and we estimated the speed of a pedestrian. The computed optical flow is reconstructed from a screen caption containing the results displayed for the two triads. These results prove the validity of our method and open up interesting perspectives for real-time applications.

Our implementation is limited by the size of the optical flow (32×32 vectors). This resolution corresponds to the resolution of most recent VLSI implementations for velocity estimation, such as the "FTC sensor" (Deutschmann et al., 1997) which provides a 15x14 optical flow with the output vectors in a limited range of available 2D directions of motion, by steps of $\pi/4$. Several other working chips were reported, but they cannot be extended in 2D because of the algorithm used (Koriuchi et al., 1992) or they are tuned to a specific velocity (Delbruck, 1993). Certain chips provide only a measure of the time-to-contact to a certain area in the scene (Etienne-Cummings et al., 1997),(Kramer et al., 1997). The only advantage of dedicated VLSI chips over our DSP implementation is the higher output framerate they provide.

Working at the second level of the spatial pyramid implies degraded estimations for small objects (3-6 pixels) and less precise motion boundaries. Our method allows a dense and better localized optical flow to be computed if needed, using multiple levels of an image pyramid, but the necessary memory would then attain 20 times the present amount of memory, and the algorithm would presumably be 20 times slower for an output of 128×128 vectors. A newer, up-to-date DSP on a board with a larger amount of memory could provide the computing power needed for a faster frame rate and eventually a better resolution.

There is a certain degree of parallelism in the algorithm (computing the four triads) which can be exploited for example by using a processor from the Texas Instruments TMS320C8x family, which includes 4 DSP cores on the same chip. A faster frame rate and a denser optical flow would then enable more complex applications such as moving object tracking, mobile robotics and scene supervision.

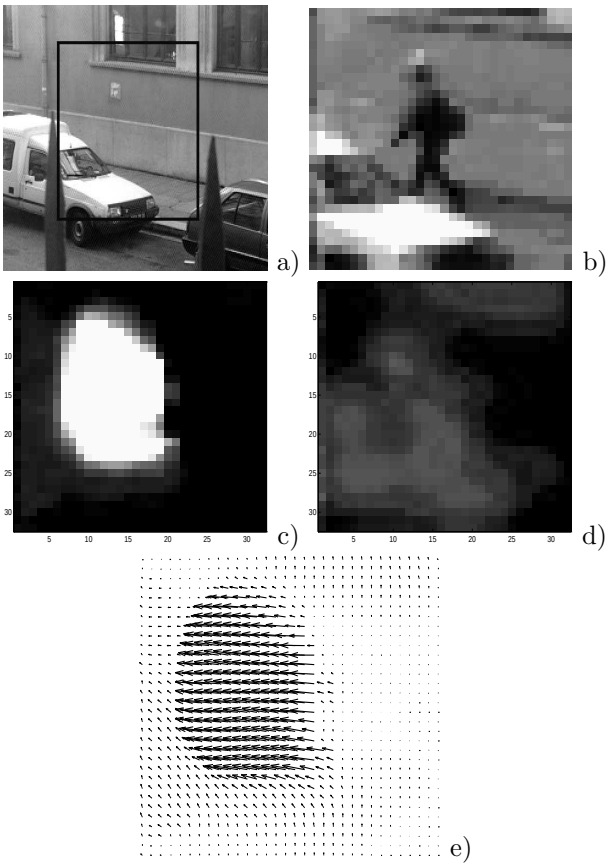

Figure 8 The results of the DSP implementation for the "Pedestrian" sequence: a) the original image containing the fixed scene, b) the processed image region and the modulus of the estimated motion c) v_x and d) v_y, e) the reconstructed optical flow using c) and d).

CONCLUSION

We have presented an energy-based method for motion estimation, using a very powerful tool in image processing: Gabor filters. The central element of our filtering architecture is the triad of Gabor filters which provides a 1D speed estimation over a preferred direction. 2D estimation is carried out by associating the results of several triads (2 or 4). In order to accelerate the process, we proposed the reduction of necessary 1D filters and the approximation of a Gabor filter by a recursive third order filter. This allows good-quality results within a reasonable computing time, according to the results of our simulations.

We proved that a DSP implementation of the algorithm is possible using a low-cost PC extension board, making the development process simpler than in the case of VLSI or parallel machine applications. This card has a simple and flexible architecture, which allows its adaptation to different image processing algorithms. Though the present generation of DSPs is much faster than the chip which is present on the PC board, we attain a reasonably short processing time (several images/second) for better resolution

than that of most recent VLSI implementations. The board is entirely autonomous and it could be easily integrated into an embarked application. These results open interesting perspectives for real-time implementations on dedicated architectures, such as in mobile robotics applications. Moreover, we firmly believe that this class of energy methods could lead to impressive results if combined with other results obtained with Gabor filtering, such as edge detection, texture and object recognition.

ACKNOWLEDGMENT

We wish to thank our colleagues Jean Pierre Charras, Christophe Dumontier, Anne Gurin-Dugu and Nabil Maria for helpful discussions. This work was partly supported by the French "Groupement d'Intérêt Scientifique" "Science de la Cognition" and the Région Rhônes-Alpes project "ACTIV".

References

Adelson, E.H. and Bergen, J.R. (1985) "Spatio-temporal energy models for the perception of motion", *Journal of the Optical Society of America A.*, 2(2), pp. 284-299.

Atick, J. and Redlich, A. (1992) "What does the retina know about natural scenes ?", *Neural Computing*, 4, pp. 196-210.

Barron, J.L., Fleet, D.J. and Beauchemin, S.S. (1992) "Performances of optical flow techniques", *Proc. Conf. on Computer Vision and Pattern Recognition*, Champaign, pp. 236-242.

Barron, J.L., Fleet, D.J. and Beauchemin, S.S. (1994) "Performance of optical flow techniques", *Int. Journal of Computer Vision*, 12(1), pp. 43-77.

Delbruck, T. (1993) "Silicon retina with correlation-based, velocity-tuned pixels", *IEEE Transactions on Neural networks*, 4, pp. 529-541.

Deutschmann, R.A., Higgins, C.M. and Koch, C. (1997) "Real-time analog VLSI sensors for 2-D direction of motion", *Proc. of 7th Int. Conf. on Artificial Neural Networks*, Lausanne, Switzerland, pp. 1163-1168.

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

Dunn, D., Higgins, W.E. and Wakeley, J. (1994) "Texture segmentations using 2-D Gabor elementary functions", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2), pp. 130-149.

Etienne-Cummings, R., Van der Spiegel, J. and Mueller, P. (1997) "A focal plane visual motion measurement sensor", *IEEE Transactions on Circuit and Systems I*, 44(1), pp. 55-66.

Fleet, D.J. and Jepson, A.D. (1990) "Computation of component image velocity from local phase information", *Int. Journal of Computer Vision*, 5, pp. 77-104.

Grzywacz, N.M. and Yuille, A.L. (1990) "A model for the estimate of local image velocity by cells in the visual cortex", *Proc. of the Royal Society London*, B, 239, pp. 129-161.

Heeger, D.J. (1987) "Model for the extraction of image flow", *Journal of the Optical Society of America A.*, 4(8), pp. 1455-1471.

Koriuchi, T.K., Bair, W., Bishofberger, B., Moore, A. and Koch, C. (1992) "Computing motion using analog VLSI vision chips: an experimental comparison among different approaches", *Int. Journal on Computer Vision*, 8, pp. 203-216.

Kramer, J., Sarpeshkar, R. and Koch, C. (1997) "Pulse-based analog VLSI velocity sensors", *IEEE Transactions on Circuit and Systems II*, 44, pp. 86-101.

Lee, J.C., Sheu, B.J., Fang, W.C. and Chellappa, R. (1993) "VLSI neuroprocessors for video motion detection", *IEEE Transactions on Neural Networks*, 4(2), pp. 178-190.

Manjunah, B.S. and Ma, W.Y. (1996) "Texture features browsing and retrieval of image data", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8), pp. 837-842.

Merhotra, R., Namaduri, K. and Ranganathan, N. (1992) "Gabor filters edge detection", *Pattern Recognition*, 25(12), pp. 1479-1494.

Mitiche, A. and Bouthemy, P. (1996) "Computation and analysis of image motion: A synopsis of current problems and methods", *Int. Journal of Computer Vision*, 19(1), pp. 29-55.

Shi, B.E. (1998) "Gabor-type filtering in space and time with cellular neural networks", *IEEE Transactions on Circuit and Systems I*, 45(2), pp. 121-132.

Spinéi, A. (1998) "Estimation du mouvement par triades de filtres de Gabor: Application au mouvement d'objets transparents", PhD thesis, Institut National Polytechnique de Grenoble, France.

Spinéi, A., Pellerin, D. and Hérault, J. (1998a) "Spatiotemporal energy-based method for velocity estimation", *Signal Processing*, 65(3), pp. 347-362.

Spinéi, A., Pellerin, D. and Hérault, J. (1998b) "Motion estimation based on triads of Gabor filters: DSP board implementation", *European Signal Processing Conference*, Rhodes, Greece, pp. 1569-1572.

Stoner, G.R. and Albright, T.D. (1993) "Image segmentation cues in motion processing: implications for modularity in vision", *Int. Journal of Cognitive Neuroscience*, 5(2), pp. 129-149.

Torralba, A. and Hérault, J. (1997) "From retinal circuits to motion processing: a neuromorphic approach to velocity estimation", *European Symposium on Artificial Neural Networks ESANN '97*, Brussels, Belgium, pp. 47-54.

Unser, M. (1993) "Fast Gabor-like windowed Fourier and continuous wavelet transforms", *National Institutes of Health, NCCR Report*, 143/93.

Wu, X. and Bhanu, B. (1997) "Gabor wavelet representation for 3-D object recognition", *IEEE Transactions on Image Processing*, 6(1), pp. 47-64.

Young, I.A. and Vliet, L.J. van (1995) "Recursive implementation of the Gaussian filter", *Signal Processing*, 44, pp. 139-151.

Zhou, Y. and Chellappa, R. (1988) "Computation of optical flow using a neural network", *IEEE Proc. Int. Conf. Neural Networks*, 2, pp. 71-78.