Parallel Architecture for Hierarchical Optical Flow Estimation Based on FPGA

Article *in* IEEE Transactions on Very Large Scale Integration (VLSI) Systems · January 2011

DOI: 10.1109/TVLSI.2011.2145423

Parallel Architecture for Hierarchical Optical Flow Estimation based on FPGA

Francisco Barranco, Matteo Tomasi, Javier Diaz, Mauricio Vanegas, and Eduardo Ros

***Abstract*—The proposed work presents a highly parallel architecture for motion estimation. Our system implements the well-known Lucas & Kanade algorithm with the multi-scale extension for the computation of large motion estimations in a dedicated device (FPGA). Our system achieves 270 frames per second for a VGA resolution in the best case of the mono-scale implementation and 32 frames per second for the multi-scale one, fulfilling the requirements for a real-time system. We describe the system architecture, address the evaluation of the accuracy with well-known benchmark sequences (including a comparative study), and show the main hardware resources used.**

***Index Terms*—Field programmable gate arrays; Machine vision; Real time systems; Reconfigurable architectures.**

I. Introduction

The optical flow computation is the challenging task of estimating bidimensional motion fields projected by the scene on the image plane of a camera from a sequence of captured frames. For the computation of the optical flow, we assume the constancy of the intensity for each pixel in the image at one instant for the successive frame, which is fulfilled with a good approximation for the object with smooth and small displacements.

Optical flow is a low-level vision feature widely used for many applications, as for instance, for the computation of middle-level applications such as motion in depth [1], structure from motion [2], IMOs (independently moving objects), [3] or heading [4]. Furthermore, its potential applications encompass video stabilization [5], object tracking [6], video de-noising and restoration [7], segmentation [8], or active vision [9][10]. All of them are useful in a wide range of fields such as autonomous robot navigation [11], video surveillance [12], or driving assistance systems [13] [14].

In our work, we implement the Lucas & Kanade (L&K) algorithm [15], particularly the model described in [16][17]. We include an extension to the original algorithm, a hierarchical architecture, capable of working with an extended motion range much larger than the standard mono-scale approaches (with a typical range of a few pixels [13]). This hierarchical architecture has been previously extensively detailed [18] and many works deal with the multi-scale implementation [19][20][21][22]. It basically consists in the construction of a pyramid of images with the image resolution being reduced one octave at each pyramid level. Then, the motion estimation stage performs the search over a small number of pixels at the coarsest scale. The obtained estimation is used as a seed to search locally at the next finer scale and so on to the finest one. This process is extensively detailed on Section II.B and significantly allows us to increase the motion range of the algorithm, which is critical for many real-world applications.

Nowadays, optical flow estimation is the main topic of a large number of works in the literature and its real-time computation is one of the main related challenges. We can find works in the literature addressing this topic using dedicated hardware architectures [23], graphic processing units (GPUs) as accelerators [24], and even optimized software implementations [25], or clusters of processors [26]. Finally, in previous works [13][27], we can also find FPGA based solutions.

In this paper, we have selected FPGA devices because local optical flow algorithms are good candidates due to their potential for a high-performance massive parallelization. The results are obtained using a fine-pipeline based architecture designed in a modular way. Our objective is to achieve a throughput of one pixel per clock cycle along the whole processing scheme. The correct exploitation of these high performance characteristics allows us to achieve, in the case of the mono-scale approach, up to 270 fps (frames per second) for an image resolution of 640x480 thanks to the presented specific-purpose architecture. In the case of the multi-scale implementation, the frame rate reaches almost 32 fps for the same image resolution with a motion working range 30 times larger than the mono-scale approach. The error results are shown in Section III.

This paper is structured as follows: in Section II, we summarize the L&K algorithm and the multi-scale extension. Section III details the hardware implementation and presents the results comparing the proposed alternatives and the hardware resource utilization. Section IV shows the architecture of the final system and its benchmarking. Finally, Section V presents the conclusions of the paper.

II. Lucas & Kanade Algorithm for Optical Flow Computation

The optical flow estimation consists in the computation of the motion field that represents the pixel displacements between successive frames, which is defined for each pixel as a velocity vector (u, v) as in (1).We compute the optical flow estimation assuming the brightness constancy, i.e. the Optical Flow Constraint (OFC) (1). In (1), we assume that

F. Barranco, M. Tomasi, J. Diaz, M. Vanegas, and E. Ros are with the Department of Computer Architecture and Technology, CITIC, ETSIIT, University of Granada, C/Daniel Saucedo Aranda s/n, E18071, Granada, Spain. M. Vanegas is also with the PSPC Group, Department of Biophysical and Electronic Engineering (DIBE), University of Genoa, Via Opera Pia 11A, I-16145, Genoa, Italy. E-mails: {fbarranco, mtomasi, jdiaz, mvanegas, eduardo}@atc.ugr.es

Manuscript received April 14, 2011; revised month day, year.

the intensity of the pixel in the (x, y) position in the image at time t, and the pixel $(x+u,\ y+v)$ in the image of time $t+1$ does not change. As commented before, this is only a valid assumption for slow-moving objects between frames (it depends on the distance from the object to the camera, on the 3D object velocity, and on the camera frame-rate).

$$I(x, y, t) = I(x+u, y+v, t+1) \tag{1}$$

This equation is solved by linearization using the first order Taylor expansion and this leads to the subsequent linear one (2)

$$I_x u + I_y v + I_t = 0 \tag{2}$$

where subscripts represent the partial derivatives in each direction (x, y, and t). Due to the fact that from (2), we cannot determine a unique solution, we need another constraint to find the solution of the optical flow estimation (only velocity vectors which are normal to the image structure could be computed with this equation). There are different approaches to solve this problem. Many local methods assume that the flow is constant in the same local neighborhood. Thus, we compute the motion estimation for a pixel focusing only on its neighborhood and ignoring the rest and it is supported because close pixels are very likely to correspond to the same objects and, therefore, similar velocity vector values can be expected. This approach was firstly proposed by Lucas & Kanade [15]. In addition, other local approaches impose that not only luminance values remain constant but also image derivatives (see for instance second or higher order gradient methods [16]). A different approach consists in solving an optimization problem minimizing a global function depending on the OFC (1) and other constraints. This is the approach originally proposed by Horn & Schunk [28].

The L&K approach is one of the most accurate, computationally efficient, and widely used methods for the optical flow computation. It is a local method that belongs to the gradient based ones, because its computation is based on the image derivatives [16][17] and because it solves the previous equation by assuming that the flow is constant in the same local neighborhood. This is the model which we have selected for our implementation. To solve the previous OFC (2), we estimate the optical flow as the value which minimizes the energy function building an over-constrained equation system as in (3)

$$E(u, v) = \frac{1}{2}\sum_{i\in\Omega}(W_i^2(I_x u + I_y v + It)^2) \tag{3}$$

where W_i stands for the weight matrix of the pixels in neighborhood ω. And then, for the resolution of the system of equations, we use a least squares-fitting procedure as shown in (3).

$$(u, v) = (A^T W^2 A)^{-1} A^T W^2 b \tag{4}$$

From (4), we solve the equation, obtaining a 2-by-2 linear system defined by (6) and (5).

$$(A^T W^2 b) = \begin{bmatrix} \sum_{i\in\Omega} W_i^2 I_{xi} I_{ti} \\ \sum_{i\in\Omega} W_i^2 I_{yi} I_{ti} \end{bmatrix} \tag{5}$$

$$(A^T W^2 A) = \begin{bmatrix} \sum_{i\in\Omega} W_i^2 I_{xi}^2 & \sum_{i\in\Omega} W_i^2 I_{xi} I_{yi} \\ \sum_{i\in\Omega} W_i^2 I_{xi} I_{yi} & \sum_{i\in\Omega} W_i^2 I_{yi}^2 \end{bmatrix} \tag{6}$$

Barron computes the confidence of the estimation using the minimum of the eigenvalues of Matrix (5). We compute the reliable values with the determinant of (5) to simplify the hardware implementation (in this case, it corresponds to the eigenvalues product) and the loss of accuracy is not significant [13][29].

The previous analysis requires that only a small displacement is presented in the scene in order for the first order Taylor expansion approximation to be valid. Therefore, the main disadvantage of the L&K algorithm is the accuracy for the estimations of large displacements, which is not possible with the previous schemes (in fact, there are some few exceptions [13] but they are not practical for real-world scenarios). For the resolution of the problem, we can adopt one of the following strategies: a) increasing the frame rate of the sequence to decrease the motion range (it depends on the capture device or the available sequences and is not always possible) or b) implementing a hierarchical approach. The second approach consists in a multi-scale implementation that computes the optical flow velocity components for each different resolution input simulating the use of filters of different sizes for a better tuning of the different range displacements.

A. *Multi-scale Implementation*

In this paper, the problem of the limited motion working range with the L&K algorithm is solved with a coarse-to-fine multi-scale implementation based on the hierarchical model proposed by [18].

The first stage is the image pyramid computation of the input frames, using a variable number of scales which mainly depends on the size of the sequence and on the range of the displacements. Then, the computation is performed in the coarse-to-fine scheme where motion computed at coarse scales is used as a seed that is refined at the next pyramid levels as described in the next paragraphs.

After the pyramid image construction, the next stage is the L&K motion estimation. A pair of estimations is obtained for the x and y velocity components $(u,\ v)$ for the current scale. At this point of the algorithm, the scale is the coarsest one.

The next step is the over-sampling of the velocity estimation to the resolution of the subsequent scale and which is also multiplied by 2 (the sub-sampling factor in order to increase one octave).

Then, the warping process is performed. This is a complex operation in terms of hardware and it consumes a significant amount of computational resources. It consists in warping the input frames with the estimations calculated in the previous

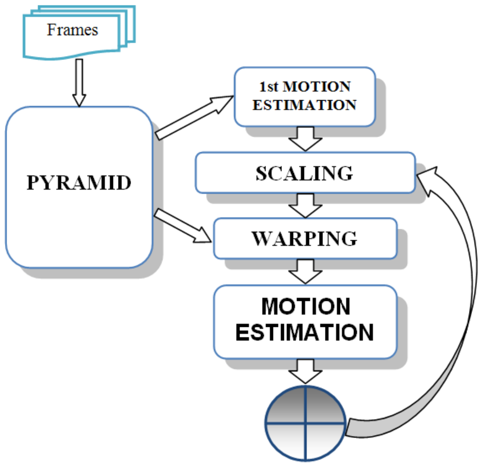

Fig. 1. Scheme of the multi-scale optical flow algorithm. Main stages: over-sampling from the previous motion estimation, warping with the new frames, new estimation computation, and merging between the two computed estimations. This is carried out by iterating from the coarsest to the finest scale.

steps to move each pixel to the previously estimated position. In such a way, we address a ”motion compensation”, reduce the range of the highest displacements, and keep local motion in the filter tuning range. The number of frames and the order of each one decide the warping of their pixels. In our case, we use 3 frames and thus, the pixels of the first and the last frames are warped (with different signs) to reduce the range of displacements with respect to the second (central) frame. Furthermore, for this sub-pixel warping, we implement a 2-by-2 bilinear interpolation to produce smoothed images, which is required for the accurate computation of the image derivatives.

The new warped frames are the input frames for the optical flow computation in the next stage, achieving a new velocity estimation (u, v).

The next stage performs the merging of the first optical flow estimation and the last one.

At this point, the algorithm iterates from the over-sampling of the velocity estimation to the merging stage. The number of iterations is defined by the number of scales set with the input parameters. The algorithm is described by the flow diagram in Fig. 1.

III. Hardware Implementation

The hardware implementation is performed in an FPGA, in our case, a Xilinx Virtex4 XC4vfx100 chip. The board, a Xirca V4 [30], provides a PCI express interface and four 8 MB SRAM ZBT memory banks and is able to work as a co-processing or stand-alone platform.

The hardware implementation has been performed using two different abstraction levels. The RTL language VHDL has been used for the Memory Controller Unit (MCU), the off-chip memory, and the PCI interfaces. We used the high level Handel-C language for the implementation of the optical flow core and the multi-scale extension. This language is much more suitable for the algorithmic descriptions with a low degradation in terms of performance or resource utilization [31].

A. Lucas & Kanade optical flow core

The implementation of the L&K core is based on a previous approach described in [13][32]. The most significant changes are the migration to the new platform, the connection with the new memory interface with the MCU, and the multi-scale implementation with warping. The multi-scale extension is a critical difference which leads to an expansion of the system working range (in terms of motion computation) at the cost of significant computing resources. This multi-scale extension is usually avoided in hardware approaches due to its architectural complexity in the framework of a massively parallel datapath. Furthermore, in this paper, instead of presenting a single implementation, we describe different alternatives at the design stage that lead to different performances vs. hardware cost trade-offs.

The input of the core consists of two channels, as is shown in Fig. 2: a) *Pixel In* is the channel of the input frame pixels (3 frames x 8 bits/frame) and b) *Control* is the channel for the parameters of the core (the number of scales, the image resolution, and the confidence thresholds). As an output, the system computes the optical flow estimation, i.e. the component velocities for the x and the y directions (12 bits per component).

The core computation is divided in 5 stages:

- S_0: The stage consists in the filtering of the input frames. The filtering uses a 5 tap Gaussian smoothing kernel. It reduces the aliasing effects and increases the accuracy of image derivatives.
- S_1: At this point, the algorithm computes the temporal derivative (Dt) and the spatio-temporal smoothing (St) of the three frames. This computation is carried out using a derivative and a 3 tap smoothing filter.
- S_2: It computes the spatial derivatives from the latter results: the It partial derivative is obtained by applying a Gaussian filtering to the temporal derivative; the I_x and I_y partial derivatives are achieved by differentiating the St term with a derivative filter in the respective direction following the Simoncelli complementary derivative kernels approach [33].
- S_3: This stage computes the coefficients of the linear system of (6) and (5). The weights W_i for the neighborhood are set by the 5-by-5 separable kernel used in [17][13][29]: $W = [1\ 4\ 6\ 4\ 1]/16$. It computes the $W_i^2 I_{ti}^2$ coefficient used as a threshold for the temporal noise too. This term is very useful for handling real-world sequences by helping to reduce the noise effects.
- S_4: The last stage calculates the resolution of the 2-by-2 system and uses the determinant of the resultant matrix to threshold the less confident results.

For this system, we use 199 parallel processing units: stage S_0 has 3 paths (one for each frame) for the Gaussian filtering, S_1 has 2 paths (for the temporal derivative and the temporal smoothing), S_2 has 3 paths (one for each derivative I_x, I_y, I_t), S_3 has 6 paths (one for each coefficient of (6) and (5)) and finally, S_4 has only one path. The number in brackets in each stage represents the micropipelined stages for each of them.

Our scheme implements four alternatives for the system

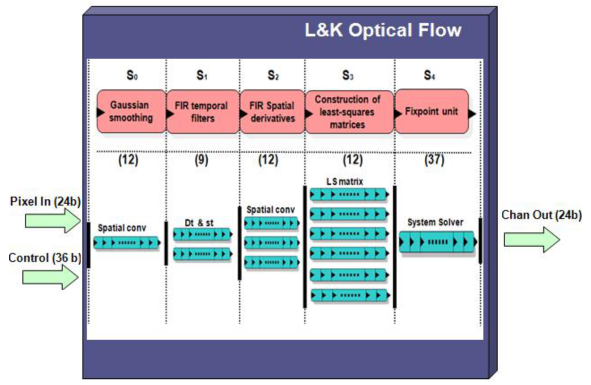

Fig. 2. Scheme of the pipelined stages for the optical flow core. It indicates the computation stages (from 0 to 4): the Gaussian smoothing to reduce aliasing effects; the computation of the spatio-temporal smoothing (St) and temporal derivative (Dt); partial derivatives (x, y, and t); the construction of the matrix for the least-squares resolution and the system solver. The numbers in parenthesis at each stage indicate the number of micropipelined stages and the parallel datapaths show the superscalar architecture level for fulfilling the high performance requirements.

solver. Two alternatives use the fix-point arithmetic and the other two, the floating-point arithmetic. They are listed below:

- *Low resources and fixed-point arithmetic (LRF)*: Implementation using fixed-point arithmetic with the lowest hardware utilization for the solver of the equation system (using as divider an IP core from the Xilinx CoreGenerator [34]).
- *High resources and fixed-point arithmetic (HRF)*: Implementation using a fixed-point arithmetic with the highest hardware utilization for the solver of the equation system (also using the IP divider)
- *Floating-point arithmetic (FLO)*: Using floating-point arithmetic and our customized and highly pipelined division circuit. This alternative is the most similar one to the alternative developed in [13].
- *Floating-point arithmetic and use of a core for the division (FCD)*: Implementation with a floating-point arithmetic and use of an IP core from the Xilinx CoreGenerator as float divider.

Table I shows the bit-width representation for the listed alternatives. In the previous work [13], the author carried out an extensive batch of simulations for the bit-width decision. Moreover, that work focused on a target implementation constrained by the hardware resources, the memory interface, and the bit-width of the embedded resources in that board. Now, our representations include more bits for the most constrained stages to improve the accuracy and the density. This is important because in the framework of multi-scale approach, the accuracy of the estimations at the coarsest scales needs to be high, because they drive the estimations of the finest ones. It means that we need a good accuracy at the coarsest scales in order to keep a high accuracy along the multi-scale approach as images are warped according to the velocity results of the coarser scales.

In Table I, the pair of values represents the number of bits of the integer and the fractional parts respectively. In the case of the use of the floating-point arithmetic (only for S_4), they represent the number of bits for the mantissa and the exponent. The *HRF* alternative presents a different bit-width configuration: this is the alternative with the highest resource use and fixed-point. The bit-width is incremented in S_1 and S_2 to exploit the full precision of the convolution operations without losing accuracy for the use of fixed-point arithmetic. In S_3, this effect is even more considerable reaching a word size of 20 bits to build the coefficients of the linear system and finally, the bit-width is incremented to 42 bits in the last stage, which performs the division, which is the stage with the highest loss of precision.

The resources used for the proposed alternatives are shown in Table II. All the results are obtained using the synthesis Xilinx ISE tools [34]. The choice with the lowest resource cost is the first one, which uses only 10% and has also the highest clock frequency (83 MHz). It is important to remark that the last alternative (which uses a floating divider core) saves resources with respect to the previous one but the frequency is 73 MHz. Although we have to add an interface to communicate the new divider core with the rest of the system and to parse the input and output parameters for the new format, the resources are lower than in the case of our pipelined division circuit (*FLO* choice).

Table III shows the achieved frame rate using the new proposed architectures (mono-scale), reaching the fastest one (*LRF*) about 270 fps.

On the other hand, the analysis of accuracy is also very significant for the optical flow estimation. Using the *"Marble"* sequence benchmark (available at [35]), we compute the AAE (average angular error, described in [17]), the SAE (standard deviation of the AAE), and the density (percentage of valid values). The five rows of Table IV represent the results for the software model and for the four alternatives explained in this section.

We obtain the best results with the floating-point approaches because of the higher density but with 3° more of AAE than the fixed-point one. It is important to remark that using the standard *"Marble"* sequence as benchmark prevents us to take full advantage of the high processing power (in terms of frames per second) to improve accuracy. This is due to its smaller inter-frame displacements or slower movements at high frame rates when the sequence is captured with high frame-rate cameras. Therefore, it should be noted that the errors of these mono-scale alternatives can be expected to be very low when using appropriate high frame rate cameras (better fitting the processing power of the mono-scale optical flow engine).

It is well-known in computer vision that the selection of the best alternative is executed depending on the target application, as a good trade-off between the required accuracy and the constraints about the maximum frequency and the resource utilization. In our case, we will present two systems for the multi-scale computation using the LRF and the FLO alternatives, because they have yielded up as the less expensive ones in terms of hardware resources with an affordable (very low) loss of accuracy.

TABLE I
BIT-WIDTH REPRESENTATION FOR THE PROPOSED ALTERNATIVES DEPENDING ON THE STAGE[a].

	S_0	S_1	S_2	S_3	S_4
LRF	[9 0]	[9 0]	[8 1]	[14 4]	[29 8]
HRF	[9 0]	[9 1]	[9 1]	[16 4]	[33 8]
FLO	[9 0]	[9 0]	[8 1]	[14 4]	[11 7][b]
FCD	[9 0]	[9 0]	[8 1]	[14 4]	[11 7][b]

[a]Integer representation: [integer part fractional part]; floating-point representation [mantissa exponent].
[b]Stage with floating point representation.

TABLE II
HARDWARE RESOURCE UTILIZATION FOR THE PRESENTED ALTERNATIVES USING A XILINX VIRTEX-4 FX100 FPGA.

	4 input LUTs	Slice Flip-Flops	Slices	Freq (MHz)
LRF	5039 (5%)	6622 (7%)	4224 (10%)	83
HRF	5399 (6%)	6851 (8%)	4803 (11%)	67
FLO	8865 (10%)	4715 (5%)	6551 (15%)	76
FCD	8203 (9%)	4838 (5%)	5916 (14%)	73

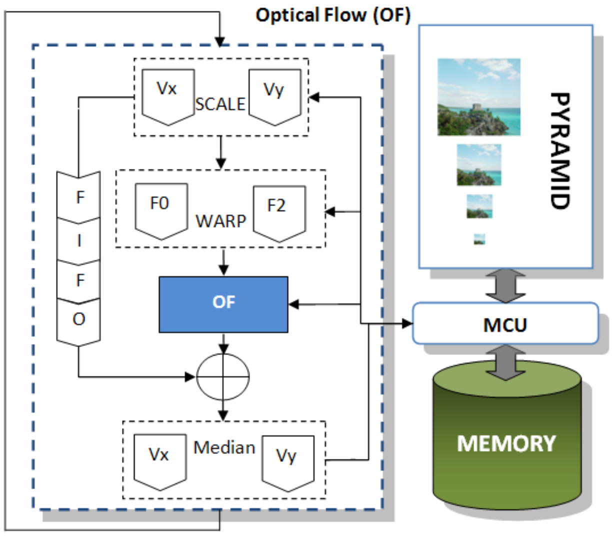

Fig. 3. Hardware system architecture. On the right side, we have the pyramid iteration and its communication with memory. On the left side, the multi-scale optical flow computation (scaling, warping, merging, median filtering, and the L&K optical flow computation) can be seen. The communication with memory is performed through the MCU.

B. Multi-scale Architecture

The components of the complete system are (see Fig. 3): the L&K optical flow core, the scaling module, the warping module, the merging module, and the median filter circuits, and they all work in parallel. For the smallest scale (the first iteration of the algorithm), the merging circuit is omitted and the optical flow block works with the pyramid output.

All the other blocks in the processing stages interact with memory through the MCU that multiplexes in time the huge amount of data which they have to read/store [36].

The data flow is displayed in Fig. 3 and can be summarized as follows:

- The Scaling circuit reads old partial optical flow values and scales them with a bilinear interpolation (the new values are multiplied by 2 to adapt the optical flow values to the next scale).
- The Warping circuit reads the pyramid images and displaces them using the expanded motion.
- The Median filtering stage removes the outliers homogenizing the partial and the final results. This stage consists in a median bidimensional filter and can be a cascade of them. This filter also contributes by incrementing the density of the final results removing the non-confident values and filling the holes with the filter results (in our case, the filter computes on a 3x3 neighborhood).
- The Merging stage allows the sum of the previous optical flow estimation and the current one in order to compute the final estimation.

The interaction with memory is a very critical problem and needs a dedicated circuit and a specific memory mapping. The parallel access to the RAM blocks is allowed using the multiple available banks and a sequential operation strategy.

- Pyramid: A pyramid is built by a smoothing and sub-sampling circuit (see Fig. 4). Each pyramid scale is obtained sequentially (mainly, due to the limitations of the sequential access to the external memory). Input and output images are directly read/stored into an external RAM memory. The main operations at this step are the 2D convolution with the 5 tap Gaussian filter (smoothing) and the sub-sampling. The kernel is a 5-by-5 matrix decomposed in two arrays $K = [1\ 4\ 6\ 4\ 1]/16$, as suggested in [37].
  The convolution of the input image is divided in the x and y operations to take advantage of the FPGA massive parallelism. Five different image lines are stored in an embedded multi-port BlockRAM used like a FIFO for the convolution.
  After the convolution computation, we send a pixel to the output (the external SRAM) every two clock cycles: one pixel is discarded (sub-sampling).
- Warping: It consists in a bilinear interpolation of the input images with the increment values that we have stored from the optical flow in the previous scale in a LUT.
  The computation of each warped pixel requires the reading of the pair $(\Delta x, \Delta y)$ from their correspondent matrices as well as the pixel P. The integer part of the pair

TABLE III
ACHIEVED FRAME RATE FOR THE MONO-SCALE ARCHITECTURE AT THE 640X480 RESOLUTION.

	LRF	HRF	FLO	FCD
Frame rate (fps)	270	218	247	238

TABLE IV
COMPARISON OF THE SOFTWARE AND HARDWARE ERROR MEASURES FOR THE *"MARBLE"* SEQUENCE FOR THE MONO-SCALE IMPLEMENTATION.

	AAE (°)	SAE (°)	Dens (%)
Software	29.14	24.31	80.90
LRF	22.84	21.47	59.71
HRF	22.84	21.48	59.70
FLO	25.98	24.77	68.17
FCD	26.03	24.86	68.15

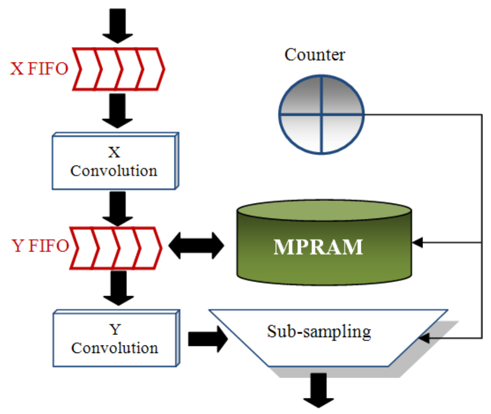

Fig. 4. Architecture for the image sub-sampling required for the pyramid building. The 2D convolution is split into two different 1D convolutions to take advantage of the inherent massive parallelism of the device.

$(\Delta x, \Delta y)$ is used for retrieving from memory the four pixels of the original image. Then, the warped pixel is calculated with the fractional part performing a weighted bilinear interpolation.
The warping process needs to execute four memory accesses per clock cycle to calculate one warped pixel to achieve the maximum throughput. This is one of the reasons for the choice of a specific MCU to manage data with different Abstract Access Ports (AAP) [36].
The warping architecture uses a reading AAP of the MCU for accessing the original image. Two reading AAPs are used by the two warping blocks: one for the first and one for the third frame in the temporal sequence. The MCU provides a 36-bit bus allowing to access four pixels per memory read. The X-matrix and Y-matrix which store the new locations of the pixels are provided from the over-sampling circuit through two blocking FIFOs.
The warping requires a neighborhood of four pixels and the number of data available per memory access is limited to four in a same line. Thus, one access brings two pixels of the same line in the best case. Nevertheless, it is impossible to access the four-pixel window in a single memory access. In the worst case, we access four different memory words (all four consecutive memory accesses). Therefore, performance is constrained to up to 4 pixels every 10 memory accesses.

- Merging: This module computes the addition of the previous optical flow estimation and the current one. The result is stored for the next iteration. The non-valid values are propagated from the coarsest scales to the finest ones. These non-confident values are obtained at each scale applying the threshold mentioned before as the eigenvalues product (5). At the last scale, the finest one, we make the logical *"and"* operation between its non-valid values and the propagated ones for the final estimation. The propagation for the other scales is implemented using an *"or"* logical operation; this difference in the computation is performed to weight more the non-valid values of the finest scale because they are the more exact ones in terms of non-confidence. The main problem at this module is the synchronization between the current and the stored results.
- Median Filtering: The stage filters the output of the L&K computation using a 3-by-3 median filter module parameterized by a threshold. The threshold controls the computation of the filter depending on the number of unreliable values. This stage can be enabled or disabled by an input parameter too. Furthermore, the module can be a cascade of two 3-by-3 median filters. Our data (Table V and Table VI) are extracted from a system with a single median filter at this stage.

C. Multi-scale System Performances and Resource Utilization

In this subsection, we show the results for the implementation of the system including the frame rate and the quantitative and qualitative results. In addition, we also present here the tables with the resource information utilization. For this implementation, we use the *LRF* core (see Section IIIA for details).

As proposed in the introduction, our objective is the implementation of a high-performance system, which means a system with a high frame rate to work in real time. In Table V, we can see the information about the frame rate related to the corresponding resolution. With an image resolution of 512 x 512 pixels, the processing of the system reaches more than 37 fps, which is significantly higher than the commonly accepted frame rate for real-time performances (25 fps). Fig. 5 shows some qualitative results for the *"Marble"* sequence.

The hardware results (Table VI) for this sequence are, in the best case (using 4 scales), very similar in AAE, about 9.5°, to the software model (which has 9.6°) losing only 9% of density with a very similar standard deviation too.

For the well-known *"Yosemite"* sequence (ignoring the clouds), the hardware AAE results with the multi-scale approach reach 4.55° with a density of 58.49%, (using the *LRF* hardware version), while the software version achieves 3.72° with 63.63% of density (in this case, the AAE increment is of 0.83°). This comparison is carried out using similar representations for hardware and software implementations (using the same regularization filter size, the same median thresholds, without a final thresholding or similar sizes for image boundaries).

In Table VII, we compare our implementation performances with previous works in different architectures and with different algorithms for the optical flow estimation. The advantages of the mono-scale version are obvious; it achieves the best results in terms of computing power (frames per second for the given resolutions) compared to some state of the art works. The multi-scale version does not achieve the same performances, but the accuracy in this case is significantly better as explained previously. Moreover, this table also includes the error measures (AAE and density) for all the alternatives (when provided by the authors). Table VII shows that our implementation achieves a good position in the rank for the Yosemite sequence taking into account that this sequence does not take full advantage of multi-scale performance improvements, because shift ranges are very small ($[-4.28, 3.25]$ for u

TABLE V
ACHIEVED FRAME RATE RELATED TO THE INDICATED SPATIAL RESOLUTION.

Resolution	512 x 512	640 x 480	800 x 600	1024 x 1024
Frame rate (fps)	37.39	31.91	20.42	9.34

TABLE VI
COMPARISON OF THE SOFTWARE AND HARDWARE ERROR (USING *LRF* VERSION) FOR THE *"Marble"* SEQUENCE USING A DIFFERENT NUMBER OF SCALES.

	Software results			Hardware results		
No. scales	AAE (°)	SAE (°)	Dens. (%)	AAE (°)	SAE (°)	Dens. (%)
1	29.14	24.31	80.90	23.08	22.23	71.51
2	13.39	15.38	79.80	11.12	14.41	70.69
3	9.64	12.26	79.63	9.73	13.03	70.68
4	9.62	13.08	79.49	9.48	12.41	70.58
5	9.16	11.46	79.48	9.99	13.07	70.56

and $[-4.19, 5.19]$ for v). There are also several L&K software implementations (real-time driven) that present lower errors than ours as the one of Marzat et al. [38], with an AAE of 2.34° (density is not provided). There are other approaches, as the OpenCV implementation [19][39], which achieves 6.10° for a fully dense motion field. In both these latter cases, they are L&K hierarchical and iterative implementations (and the error values are obtained using the Yosemite sequence without clouds). In any case, in the framework of on-chip implementations, it is important to remark that the accuracy of our approach is one of the highest of those described in the literature, which validates our design.

Table VIII shows the information about the hardware resource utilization. Table VIII presents the percentages and the total amount of used elements: the total 4 input LUTs, the Slice Flip Flops, and the Slices, used DSPs, Block RAMs and, finally, the maximum frequency (MHz) which is reached is shown.

The listed options include the complete systems with the implemented cores using either the fixed-point or the floating-point arithmetic; it also includes the interface with the resources of the selected board and the MCU, the implemented modules for the multi-scale computation, the over-sampling module, the warping module (which includes two warping modules, one for each direction, and the interface with the MCU), and the merging module. It is important to remark that the systems use between 61% and 64% of the available resources of a Virtex4 FX100. Therefore, in the framework of real applications, it is possible to add some more cores to the multi-scale architecture to build in new on-chip engines capable of computing other vision features. This would lead to a real-time on-chip engine for the computation of low-level vision features which are significantly powerful for systems in the middle and high-level.

IV. THE TESTING PLATFORM

In our work, the FPGA has been used as a co-processing board and it is connected to a computer. In this system, images are written by the computer into the available memory banks at the board and the communication is carried out using the PCI express interface. We also implemented a hardware-software protocol with a double-buffer architecture for the communication between the board and the computer.

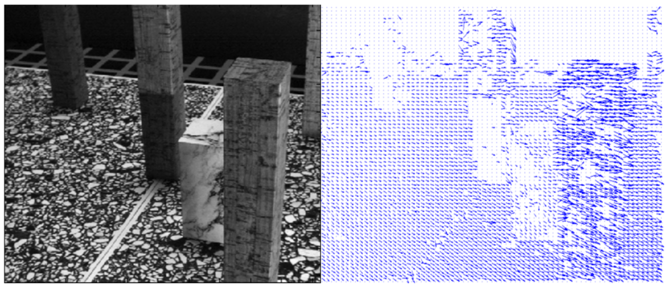

Fig. 5. Captures of the hardware generated qualitative results for the *"Marble"* sequence. The left image shows the central and the right frame of the motion estimation. The arrows have the direction and magnitude of the computed estimation.

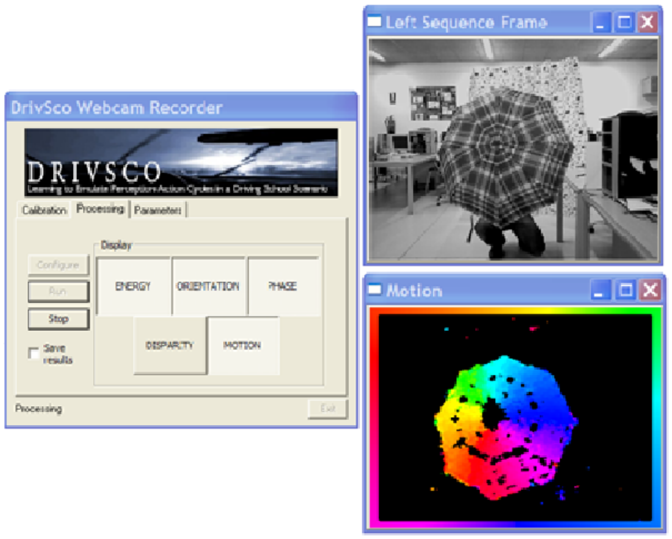

Fig. 6. Capture of the software environment in a real-time execution of the optical flow. The input is an umbrella rotating counter clockwise with a color-coding describing the direction of the motion according to the frame of the picture.

The use of a simple and efficient user interface is a crucial tool for the development of complex systems. In our case, the hardware implementation was tested using a software environment which has the role of an interface between the capture of the images from the real world using different kinds of cameras and the co-processing board. The developed software allows us to execute the implemented algorithms with the FPGA board for their display and the storing of the obtained results. It also allows the debugging of the hardware algorithms and the comparison with the rest of alternatives and vision algorithms or implementations (hardware or software).

The software platform is freeware (open source) and was released in 2009 by the GNU LGPL license, thus the source files, the documentation, the user's manual, the tutorial, and the video-tutorial are available at [48]. In Fig. 6, we can see a screenshot of this software platform.

TABLE VII
PERFORMANCE COMPARISON WITH PREVIOUS WORKS (SORTED BY PUBLICATION DATE) AND ERROR MEASURE FOR *"Yosemite"* SEQUENCE.

Implementation	Algorithm Family	Max. Image Resolution	Frame rate (fps)	Throughput (Mpixels/s)	Architecture	AAE (°)	Dens. (%)	Cloud handling
Our mono-scale[c] core	L&K	640x480	270	82.9	Xilinx V4 (83 MHz)	5.97	59.88	Cloudless
Our multi-scale work	L&K	640x480	31.91	9.8	Xilinx V4 (44 MHz)	4.55	58.50	Cloudless
Tomasi [40] (2010)	Multiscale Phase-based	640x480	31.5	9.6	Xilinx V4 (45 MHz)	7.91	92.01	Cloudless
Botella [41] (2010)	Multi-channel Gradient	128x96	16	0.2	Xilinx V2	5.5	100	UNK[d]
Mahalingam [42] (2010)	L&K (mono-scalar)	640x480	30	9.2	Xilinx V2 Pro (55 MHz)	6.37	38.6	UNK
Anguita [25] (2009)	L&K (mono-scalar)	1280x1026	68.5	90.0	Core2 Quad Q9550 (2830 MHz)	3.79	71.8	Cloudless
Gwosdek [43] (2009)	Variational	316x252	210	16.5	Cell Processor (PS3)	5.73	UNK	With clouds
Pauwels [24] (2008)	Phase-based	640x512	48.5	15.9	NVIDIA GeForce 8800 GTX	2.09	63	Cloudless
Diaz [13] (2008)	L&K (mono-scalar)	800x600	170	81.6	Xilinx V2 (82 MHz)	7.86	57.2	Cloudless
Chase [44] (2008)	Tensor-based (mono-scalar)	640x480	64	19.7	Xilinx V2 (187MHz)	12.9	UNK	With clouds
Chase [44] (2008)	Tensor-based (mono-scalar)	640x480	150	46.1	NVIDIA GeForce 8800 GTX	12.9	UNK	With clouds
Wei [45] (2007)	Tensor-based	640x480	64	19.7	Xilinx V2 (100 MHz)	12.7	UNK	Cloudless
Bruhn [46] (2006)	Variational	160x120	63	1.2	Intel Pentium4 (3.06 GHz)	5.77	100	Cloudless
Niitsuma [47] (2005)	Region-based	640x480	30	9.2	Xilinx V2	UNK	UNK	UNK

[c]This implementation only refers to the best mono-scale optical flow core (*LRF* core).
[d]*UNK* means Unknown (data is not provided by the authors).

TABLE VIII
HARDWARE RESOURCE UTILIZATION FOR THE PRESENTED COMPLETE ARCHITECTURE USING A VIRTEX-4 FX100 FPGA.

	4 input LUTs (out of 84352)	Slice Flip-Flops (out of 84352)	Slices (out of 42716)	DSP (160)	Block RAM (378)	Freq. (MHz)
L&K LRF sys	31796(37%)	24694 (29%)	26036 (61%)	62 (38%)	112 (29%)	44
L&K FLO sys	35753(42%)	22586 (26%)	27359 (64%)	44 (27%)	107 (28%)	41
L&K LRF core	4589 (5%)	6622 (5%)	4128 (9%)	30 (18%)	48 (12%)	83
L&K FLO core	8160 (9%)	4715 (5%)	6457 (15%)	12 (7%)	48 (12%)	76
Board Interface	4774 (5%)	5195 (6%)	5388 (12%)	0	36 (9%)	112
Over-sampling	413 (1%)	270 (1%)	367 (1%)	0	1 (1%)	86
Warping + Int.	9943 (11%)	9097 (10%)	9894 (23%)	32 (20%)	43 (11%)	51
Merging	364 (1%)	244 (1%)	235 (1%)	0	4(1%)	107

V. CONCLUSIONS

In this work, we have designed and implemented an embedded system for the optical flow estimation. Among the numerous alternatives, we selected a multi-scale implementation of the well-known L&K method due to its good trade-off between accuracy and efficiency. However, we have also described the mono-scale L&K approach as a valid alternative when using high frame-rate cameras.

One of the main objectives which we achieved is the real-time optical flow computation capable of dealing with large displacements. The system achieves 270 fps for the mono-scale approach and 32 fps for the multi-scale one for VGA resolution taking full advantage of the massive parallelism of the FPGA, also obtaining good accuracy rates. Moreover, the hardware resource utilization is a key point too. In our case, for the proposed system, the mono-scale implementation corresponds to 10% - 15% of the available resources in a Xilinx Virtex4 XC4vfx100 device while the multi-scale takes about 60%. For our system, this increase in resources (roughly 45%) allows us to improve the accuracy results in about 3x while maintaining the density results but leading to almost a 10x decrease of the frame rate (Table VI). The resource increment is mainly due to the warping stage of the hierarchical implementation, which roughly consumes 23% of the total resources. Furthermore, the maximum working frequency is decreased from 83 MHz for the mono-scale core to 44 MHz for the complete system, being constrained by the warping module whose frequency is 51 MHz. This is a key point to understand that our system bottleneck is the warping computation (in terms of cost and performances). Moreover, as shown in Table VII, our AAE of 4.55 degrees with almost 60% of density is one of the best implementations of the listed ones and actually, the best of the hardware implementations.

The fine pipelined architecture benefits the high system performances and the low power consumption [13] (crucial for industrial applications). Rather than a specific system design, the described implementation shall be seen as a versatile approach that can be easily adapted to different performance versus hardware resource trade-offs, depending on the target application requirements. The adopted design strategy allows an easy sharing of hardware resources (using more or less pipelined stages and superscalar units). This makes the system definition easy to re-use in different application domains.

The characteristics which have been listed before make the system suitable for the implementation of more complex systems by adding new computation engines of visual features as depth, orientation, and phase. Moreover, we can even add new layers in order to compute IMOs detection, heading, or structure from motion, as indicated in the introduction section.

As future works, we will address the inclusion of additional on-chip image features in order to be able to develop generic and more complex image applications.

Acknowledgment

This work has been supported by the Spanish project DINAM-VISION (DPI2007-61683), the Spanish Grant (AP2007-00275), and by the regional projects (P06-TIC02007, TIC-3873).

References

[1] S. P. Sabatini, F. Solari, and G. M. Bisio, "Spatiotemporal neuromorphic operators for the detection of motion-in-depth," pp. 874–880, 2000.

[2] P. C. Merrell and D. Lee, "Structure from motion using optical flow probability distributions," in *Intelligent Computing: Theory and Applications III*, K. L. Priddy, Ed., vol. 5803. SPIE, 2005, pp. 39 – 48.

[3] K. Pauwels, N. Kruger, M. Lappe, F. Worgotter, and M. M. V. Hulle, "A cortical architecture on parallel hardware for motion processing in real-time," *Journal of Vision*, vol. 10, no. 18, pp. 1– 21, 2010.

[4] K. Pauwels and M. M. V. Hulle, "Optimal instantaneous rigid motion estimation insensitive to local minima," *Computer Vision and Image Understanding*, vol. 104, no. 1, pp. 77 – 86, 2006.

[5] Y. Matsushita, E. Ofek, W. Ge, X. Tang, and H.-Y. Shum, "Full-frame video stabilization with motion inpainting," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 7, pp. 1150 –1163, 2006.

[6] T. Brox, B. Rosenhahn, D. Cremers, and H. Seidel, "High accuracy optical flow serves 3-D pose tracking: exploiting contour and flow based constraints," in *European Conference on Computer Vision (ECCV)*, vol. 3952, 2006, pp. 98 – 111.

[7] A. Kokaram, "On missing data treatment for degraded video and film archives: a survey and a new bayesian approach," *Image Processing, IEEE Transactions on*, vol. 13, no. 3, pp. 397 –415, 2004.

[8] J. M. L. K. Nakayama, "Optical velocity patterns, velocity-sensitive neurons and space perception," *Perception*, vol. 3, pp. 63– 80, 1974.

[9] J. Aloimonos, I. Weiss, and A. Bandyopadhyay, "Active vision," *Int Journal of Computer Vision*, vol. 1, pp. 333 – 356, 1988.

[10] F. Barranco, J. Diaz, E. Ros, and B. del Pino, "Visual system based on artificial retina for motion detection," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 39, pp. 752 – 762, 2009.

[11] A. Giachetti, M. Campani, and V. Torre, "The use of optical flow for the autonomous navigation," in *Proceedings of the third European conference on Computer vision*, ser. ECCV '94, vol. 1. Springer-Verlag New York, Inc., 1994, pp. 146 – 151.

[12] A. Wali and A. M. Alimi, "Event detection from video surveillance data based on optical flow histogram and high-level feature extraction," in *Database and Expert Systems Application, 20th International Workshop on*, 2009, pp. 221 – 225.

[13] J. Diaz, E. Ros, R. Agis, and J. Bernier, "Superpipelined high-performance optical-flow computation architecture," *Computer Vision and Image Understanding*, vol. 112, no. 3, pp. 262 – 273, 2008.

[14] H. Frenz, M. Lappe, M. Kolesnik, and T. Bhrmann, "Estimation of travel distance from visual motion in virtual environments," *ACM Transactions on Applied Perception*, vol. 4, pp. 419 – 436, 2007.

[15] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," pp. 674 – 679, 1981.

[16] S. S. Beauchemin and J. L. Barron, "The computation of optical flow," *ACM Computing Surveys*, vol. 27, no. 3, pp. 433 – 466, 1995.

[17] J. L. Barron, D. J. Fleet, and S. S. Beauchemin, "Performance of optical flow techniques," *Int Journal of Computer Vision*, vol. 12, pp. 43 – 77, 1994.

[18] J. Bergen, P. Anandan, K. Hanna, and R. Hingorani, "Hierarchical model-based motion estimation," in *Computer Vision ECCV'92*, ser. Lecture Notes in Computer Science, G. Sandini, Ed., vol. 588, 1992, pp. 237 – 252.

[19] J. Y. Bouguet, "Pyramidal implementation of the lucas-kanade feature tracker: description of the algorithm," OpenCV Document, Intel Microprocessor Research Labs,, Tech. Rep., 2000.

[20] S. Baker and I. Matthews, "Equivalence and efficiency of image alignment algorithms," in *Computer Vision and Pattern Recognition, CVPR. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1, 2001, pp. 1090 – 1097.

[21] F. Dellaert and R. T. Collins, "Fast image-based tracking by selective pixel integration," in *ICCV Workshop on Frame Rate Processing*, 1999.

[22] G. Hager and P. Belhumeur, "Efficient region tracking with parametric models of geometry and illumination," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, no. 10, pp. 1025 –1039, 1998.

[23] Y. Murachi, Y. Fukuyama, R. Yamamoto, J. Miyakoshi, H. Kawaguchi, H. Ishihara, M. Miyama, Y. Matsuda, and M. Yoshimoto, "A vga 30-fps realtime optical-flow processor core for moving picture recognition," *IEICE Transactions on Electronics*, vol. 91, pp. 457–464, 2008.

[24] K. Pauwels and M. M. V. Hulle, "Realtime phase-based optical flow on the GPU," in *Computer Vision and Pattern Recognition Workshops, CVPRW '08. IEEE Computer Society Conference on*, 2008, pp. 1 – 8.

[25] M. Anguita, J. Diaz, E. Ros, and F. J. Fernandez-Baldomero, "Optimization strategies for High-Performance computing of Optical-Flow in General-Purpose processors," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 19, no. 10, pp. 1475 – 1488, 2009.

[26] T. Kohlberger, C. Schnorr, A. Bruhn, and J. Weickert, "Domain decomposition for variational optical-flow computation," *Image Processing, IEEE Transactions on*, vol. 14, no. 8, pp. 1125 – 1137, 2005.

[27] J. Diaz, E. Ros, F. Pelayo, E. M. Ortigosa, and S. Mota, "FPGA-based real-time optical-flow system," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 16, no. 2, pp. 274 – 279, 2006.

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

[29] J. W. Brandt, "Improved accuracy in Gradient-Based optical flow estimation," *Int. J. Comput. Vision*, vol. 25, pp. 5 – 22, 1997.

[30] Sevensols, "Seven solutions." [Online]. Available: http://www.sevensols.com/

[31] E. Ortigosa, A. Canas, E. Ros, P. Ortigosa, S. Mota, and J. Diaz, "Hardware description of multi-layer perceptrons with different abstraction levels," *Microprocessors and Microsystems*, vol. 30, no. 7, pp. 435 – 444, 2006.

[32] J. Diaz, "Multimodal bio-inspired vision system. high performance motion and stereo processing architecture," Ph.D. dissertation, Universidad de Granada, 2006.

[33] E. P. Simoncelli, "Design of multi-dimensional derivative filters," in *Image Processing, 1994. Proceedings. ICIP-94., IEEE International Conference*, vol. 1, 1994, pp. 790 – 794.

[34] I. Xilinx, "FPGA and CPLD solutions from xilinx, inc." [Online]. Available: http://www.xilinx.com/

[35] M. Otte and H.-H. Nagel, "Optical flow estimation: advances and comparisons," in *Proceedings of the third European conference on Computer vision*, ser. ECCV '94. Springer-Verlag New York, Inc., 1994, pp. 51 – 60.

[36] M. Vanegas, M. Tomasi, J. Diaz, and E. Ros, "Multi-port abstraction layer for FPGA intensive memory exploitation applications," *Journal of Systems Architecture*, vol. 56, no. 9, pp. 442 – 451, 2010.

[37] P. J. Burt, Edward, and E. H. Adelson, "The laplacian pyramid as a compact image code," *IEEE Transactions on Communications*, vol. 31, pp. 532 – 540, 1983.

[38] D. A. Marzat J., Dumortier Y., "Real-time dense and accurate parallel optical flow using cuda," in *Proceedings of the 17th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*, 2009.
[39] M. Heindlmaier, L. Yu, and K. Diepold, "The impact of nonlinear filtering and confidence information on optical flow estimation in a lucas &#38; kanade framework," in *Proceedings of the 16th IEEE international conference on Image processing*, 2009, pp. 1573 – 1576.
[40] M. Tomasi, M. Vanegas, F. Barranco, J. Diaz, and E. Ros, "High-Performance Optical-Flow architecture based on a multiscale, Multi-Orientation Phase-Based model," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 20, no. 12, pp. 1797 – 1807, 2010.
[41] G. Botella, A. Garcia, M. Rodriguez-Alvarez, E. Ros, U. Meyer-Baese, and M. C. Molina, "Robust bioinspired architecture for optical-flow computation," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 18, pp. 616 – 629, 2010.
[42] V. Mahalingam, K. Bhattacharya, N. Ranganathan, H. Chakravarthula, R. Murphy, and K. Pratt, "A VLSI architecture and algorithm for LucasKanade-Based optical flow computation," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 18, no. 1, pp. 29 – 38, 2010.
[43] P. Gwosdek, A. Bruhn, and J. Weickert, "Variational optic flow on the sony playstation 3," *Journal of Real-Time Image Processing*, vol. 5, pp. 163 – 177, 2010.
[44] J. Chase, B. Nelson, J. Bodily, Z. Wei, and D.-J. Lee, "Real-time optical flow calculations on fpga and gpu architectures: A comparison study," in *Field-Programmable Custom Computing Machines. FCCM '08. 16th International Symposium on*, 2008, pp. 173 – 182.
[45] M. Martineau, Z. Wei, D.-J. Lee, and M. Martineau, "A fast and accurate tensor-based optical flow algorithm implemented in fpga," in *Applications of Computer Vision, 2007. WACV '07. IEEE Workshop on*, 2007, pp. 18 – 18.
[46] A. Bruhn, J. Weickert, T. Kohlberger, and C. Schnrr, "A multigrid platform for Real-Time motion computation with Discontinuity-Preserving variational methods," *Int. J. Comput. Vision*, vol. 70, pp. 257 – 277, 2006.
[47] H. Niitsuma and T. Maruyama, "High speed computation of the optical flow," in *Image Analysis and Processing ICIAP 2005*, vol. 3617, 2005, pp. 287 – 295.
[48] O. rt Vision project, "Open rt-vision project," http://code.google.com/p/open-rtvision/. [Online]. Available: http://code.google.com/p/open-rtvision/

Francisco Barranco received his BS in Computer Science from the University of Granada in 2007 and his MSc in Computer and Network Engineering in 2008. He works at the Department of Architecture and Computer Technology at the same university. His main research interests deal with image processing architectures and embedded systems based on reconfigurable devices, real-time machine vision, general purpose graphical programming devices, biologically processing schemes, spiking neurons. He is currently participating in an EU project related with adaptive learning mechanisms and conventional control.

Matteo Tomasi received his bachelor degree in Electronic Engineering in 2006 from the University of Cagliari, Italy. He is currently involved in a Ph.D. programme at the University of Granada since 2007 at the Department of Computer Architecture and Technology. He had involved at the European Project DRIVSCO. He received his MSc in Computer Engineering in 2007 from University of Granada. His main research interests are HDL and hardware design, Real-time systems, Reconfigurable Systems, computer vision and image processing.

Mauricio Vanegas received his bachelor degree in Electronic Engineering in 2001 from the University Pontificia Bolivariana in Medellin, Colombia. He was lecturer at the Electrical and Electronic faculty of the University Pontificia Bolivariana from 2002 until 2006. He received his Ph.D. at the University of Granada in 2010. He is currently working at the Department of Biophysical and Electronic Engineering (DIBE) at the University of Genoa, Italy. He is interested in Embedded Systems, HDL and hardware specification, Real-Time systems, and Reconfigurable Systems.

Javier Diaz received his MS in Electronics Engineering in 2002 and a Ph.D. in Electronics in 2006 both from the University of Granada. Currently, he is assistant professor at the Department of Computer Architecture and Technology at the same university. His main research interests are cognitive vision systems, high performance image processing architectures and embedded systems based on reconfigurable devices. He is also interested in spiking neurons, biomedical devices and robotics.

Eduardo Ros received the Ph.D. degree in 1997 from the University of Granada. He is currently Associate Professor at the Department of Computer Architecture and Technology at the same University. He is currently the responsible researcher at the University of Granada of two European projects related with bio-inspired processing schemes and real-time image processing. His research interests include hardware implementation of digital circuits for real time processing in embedded systems and high performance computer vision.