Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2008, Article ID 428265, 9 pages
doi:10.1155/2008/428265

*Research Article*

# Neuromorphic Configurable Architecture for Robust Motion Estimation

**Guillermo Botella,[1] Manuel Rodríguez,[2] Antonio García,[3] and Eduardo Ros[2]**

[1] *Department of Computer Architecture and Automation, Complutense University of Madrid, 28040 Madrid, Spain*
[2] *Department of Computer Architecture and Technology, University of Granada, 18071 Granada, Spain*
[3] *Department of Electronics and Computer Technology, University of Granada, 18071 Granada, Spain*

Correspondence should be addressed to Guillermo Botella, gbotella@fdi.ucm.es

Received 1 July 2008; Accepted 8 October 2008

Recommended by Gustavo Sutter

The robustness of the human visual system recovering motion estimation in almost any visual situation is enviable, performing enormous calculation tasks continuously, robustly, efficiently, and effortlessly. There is obviously a great deal we can learn from our own visual system. Currently, there are several optical flow algorithms, although none of them deals efficiently with noise, illumination changes, second-order motion, occlusions, and so on. The main contribution of this work is the efficient implementation of a biologically inspired motion algorithm that borrows nature templates as inspiration in the design of architectures and makes use of a specific model of human visual motion perception: Multichannel Gradient Model (McGM). This novel customizable architecture of a neuromorphic robust optical flow can be constructed with FPGA or ASIC device using properties of the cortical motion pathway, constituting a useful framework for building future complex bioinspired systems running in real time with high computational complexity. This work includes the resource usage and performance data, and the comparison with actual systems. This hardware has many application fields like object recognition, navigation, or tracking in difficult environments due to its bioinspired and robustness properties.



## 1. Introduction

Bioinspired systems emulate the behavior of biological ones. Neuromorphic approximations [1] are based on the way how the nervous systems create physical architectures and computations, attending to the morphology, information coding, robustness against damage, and so on. Neuromorphic systems usually deliver good primitives for the building of more complex systems, being the output of each system simpler than its input. This data reduction helps in the task of integrating every response associated with all information channels [2].

Attending to the estimation of a pixel motion inside the image sequence, there are many models and algorithms that could be classified as belonging to the matching domain approximations [3], energy models [4], and gradient models [5]. Related to this last family, different studies [6–8] show that this represents an admissible choice for keeping a tolerable tradeoff between accuracy and computing resources. For designing systems operating efficiently, it is required to deal with many challenges, such as robustness, static patterns, illumination changes, different kinds of noise, contrast invariance, and so on. If bioinspirational behavior is required, that is, ability to detect correct motion related to optical illusions or avoiding operations like matrix inverse or iterative methods that are not biologically justified, we have to select carefully a model that carries out this kind of requirements. This is the Multichannel Gradient Model (McGM) [9–12].

Motivated by these previous results and analysis, we present the architecture and implementation of a customizable optical flow embedded processing core running in real time. This system works in the framework of a codesign scheme that is able to manage complex situations in real environments [13] better than other algorithms [14] and mimic some behavior of the mammalians [15].

This paper is organized as follows. First, the stages of McGM model are explained very briefly; after that, we tackle

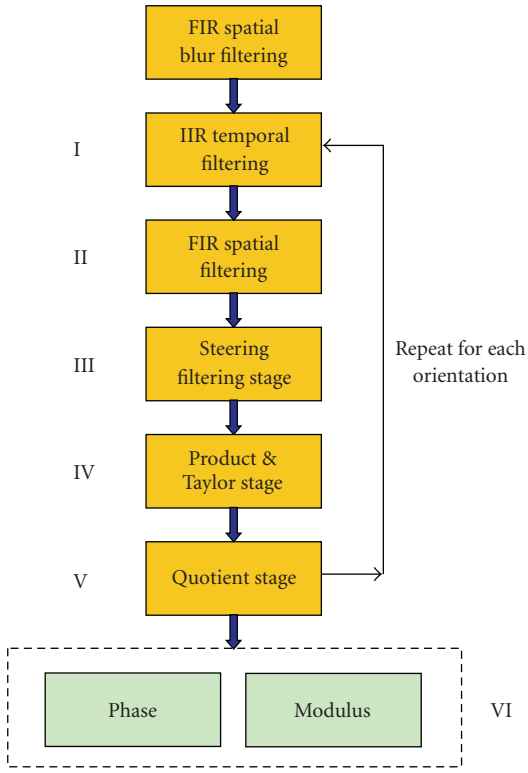


Figure 1: General scheme of the McGM algorithm.

the precision study of every conceptual stage, obtaining a set of bit width values which models the filters and the bit width stage required to obtain results that match with the statistical error metric requirements. From this previous study, we design the customizable architecture implementation attending to the original model plus several hardware modifications in order to improve the feasibility of the system. An example of this is the design of IIR filters replacing the original FIR filters due to the memory limitation of the prototyping platform, or the use of several information channels with a few bit width, replicating the nature of the brain (large number of neurons with very little precision for a few channels with huge information capacity) [14]. After that, we explain the coarse pipeline processing architecture and the platform and language used in our systems. Finally, quality results, hardware associated cost, and comparisons to other implementations are shown.

## 2. Multichannel Gradient Model (McGM)

The original algorithm was proposed by Johnston and Clifford, and we have applied Johnston's description of the McGM model [9], while adding several variations to improve the viability of the hardware implementation. Figure 1 shows a simplified scheme of the algorithm.

### 2.1. IIR Filtering

A temporal IIR filter is modeled from its original FIR description due to the limitation of available memory in our prototyping platform [15, 16]. The result is a recursive filter with only two frames of latency, being $o$ and $i$ the output and input, respectively, of the filter and $a_i$ , $b_i$ the coefficients from our previous work [14, 15]:

$$O(n) = a_1 i(n-1) - b_1 o(n-1) - b_2 o(n-2), \quad (1)$$

where $a_1 = e^{-1/\alpha}/\alpha^2$; $b_1 = 2e^{-2/\alpha}$, $b_2 = e^{-2/\alpha}$, and $\alpha$ drives the peak in the temporal impulse response function. It is calibrated with a frame peak value equals to 10 following a critical flicker fusion limit of 60 Hz, according to the human visual system evidences [11]:

$$R(t) = \frac{1}{\sqrt{\pi}\tau\alpha} e^{-(\ln(t/\alpha)/\tau)_2}. \quad (2)$$

Attending to the original algorithm, we need to perform the order zero, one and two derivatives, which represent our first triplet of information to be processed, as shown in Figure 1. The derivatives are obtained applying a gradient operator of minimal length (+1,−1) to (1):

$$\begin{aligned} T_0(n) &= o(n-1), \\ T_1(n) &= o(n) - o(n-2), \\ T_2(n) &= o(n) - 2o(n-1) + o(n-2). \end{aligned} \quad (3)$$

### 2.2. FIR Spatial Filtering

A set of spatial FIR filters is modeled by the next impulse response corresponding to bidimensional Gaussians and their separable derivatives:

$$\begin{aligned} \frac{d^n}{dx^n}(G_O) &= \frac{d^n}{dx^n}\left(\frac{e^{-(x^2+y^2)/2\sigma^2}}{\sigma\sqrt{2\pi}}\right) \\ &= Hn\left(\frac{x}{\sqrt{2}\sigma}\right) Hn\left(\frac{y}{\sqrt{2}\sigma}\right)\left(\frac{-1}{\sqrt{2}\sigma}\right)^{2n} \\ &\quad \cdot \left(\frac{e^{-(x^2+y^2)/2\sigma^2}}{\sigma\sqrt{2\pi}}\right), \end{aligned} \quad (4)$$

where $\sigma$ represents the spread of the Gaussian and $H_n$ is the Hermite polynomial of order $n$ . The convolution is done in a separable way, taking derivatives in $x$ and $y$ directions up to sixth and second order, respectively, due to bioinspired and robustness reasons [11–13]. The aim of this stage is to cover enough spread area of information channels that allow us to contribute to the calculus when any of them are null due to many reasons, such as noise. Therefore, we have three spatial structures, each one containing a pyramidal set of several filters corresponding to Gaussians and their different derivatives.

### 2.3. Steering Stage

The steering stage represents the approach to projecting the space-temporal filters calculated in previous stages, under

the different orientations. Being $n$ and $m$ the order in $x$ and $y$ directions, respectively, $\theta$ the angle projected, $D$ the derivative operator, and $G_O$ the Gaussian expression, we obtain the general expression of the filter rotated in the space as a linear combination of filters belonging to the same order basis [14]. Thus, we have to apply this transformation to each value:

$$G^{\theta}_{n,m}(x,y) = \left[\sum_{k=0}^{n}\binom{n}{k}\left(D_x\cos\theta\right)^{k}\left(D_y\sin\theta\right)^{n-k}\right] \cdot\left[\sum_{i=0}^{n}\binom{m}{i}\left(-D_x\sin\theta\right)^{i}\left(D_y\cos\theta\right)^{m-i}\right]G_0. \tag{5}$$

## 2.4. Taylor Expansion Stage

In this stage a truncated Taylor expansion is done, substituting it for the point on the space-time image in order to further enhance the algorithm. To perform this, it is necessary to use each oriented filter previously calculated. This expansion is highly versatile and represents a robust information structure of the sequence in space and time:

$$I^{\theta}(x+p,y+q,t+r) = \sum_{i=0}^{l}\sum_{j=0}^{m}\sum_{k=0}^{n}\frac{p^i q^i r^k}{i!\,j!\,k!}\frac{\partial^n}{\partial x^i\partial y^j\partial t^k} \times I^{\theta}(x,y,t). \tag{6}$$

With this, it is necessary to differentiate each Taylor expansion respect to $x$, $y$, $t$, calling these derivatives $X$, $Y$, $T$, and forming the following sextet of quotient as shown in the quotient stage:

$$\begin{bmatrix} X^{\theta} = \partial I^{\theta}/\partial x \\ Y^{\theta} = \partial I^{\theta}/\partial y \\ T^{\theta} = \partial I^{\theta}/\partial t \end{bmatrix}_{3\times 1} \longrightarrow \begin{bmatrix} X^{\theta}X^{\theta} & X^{\theta}Y^{\theta} & X^{\theta}T^{\theta} \\ Y^{\theta}Y^{\theta} & Y^{\theta}T^{\theta} & T^{\theta}T^{\theta} \end{bmatrix}_{2\times 3}. \tag{7}$$

## 2.5. Quotient Stage (General Primitives) and Following Stages

This is the last stage belonging to the common path, where a quotient of every sextet's component is computed from every measurement of the product of steered Taylor expansion differentiates:

$$\begin{bmatrix} Y^{\theta}Y^{\theta}/T^{\theta}T^{\theta} & X^{\theta}Y^{\theta}/X^{\theta}X^{\theta} & X^{\theta}T^{\theta}/X^{\theta}X^{\theta} \\ Y^{\theta}Y^{\theta}/X^{\theta}X^{\theta} & X^{\theta}Y^{\theta}/Y^{\theta}Y^{\theta} & X^{\theta}T^{\theta}/T^{\theta}T^{\theta} \end{bmatrix}_{2\times 3}. \tag{8}$$

The architecture of the core is branched in two separated ways, modulus and phase, with different bit operations working independently, containing products, several quotients, and even trigonometric operations as arctangent, which are performed in software. The details of the software stages can be found in previous works [14, 15] being the final aim to recover a dense representation of motion. Therefore, we have two values for each input pixel corresponding to modulus and phase of the velocity, *that is*, velocity projection in $x$ and $y$ directions, following the next expressions

$$\text{Phase} = \tan^{-1}\left(\frac{\left(\frac{X\cdot T}{T\cdot T}+\frac{X\cdot T}{X\cdot X}\left(1+\left(\frac{X\cdot Y}{X\cdot X}\right)^2\right)^{-1}\right)\sin(\theta)+\left(\frac{Y\cdot T}{T\cdot T}+\frac{Y\cdot T}{Y\cdot Y}\left(1+\left(\frac{X\cdot Y}{Y\cdot Y}\right)^2\right)^{-1}\right)\cos(\theta)}{\left(\frac{X\cdot T}{T\cdot T}+\frac{X\cdot T}{X\cdot X}\left(1+\left(\frac{X\cdot Y}{X\cdot X}\right)^2\right)^{-1}\right)\cos(\theta)-\left(\frac{Y\cdot T}{T\cdot T}+\frac{Y\cdot T}{Y\cdot Y}\left(1+\left(\frac{X\cdot Y}{Y\cdot Y}\right)^2\right)^{-1}\right)\sin(\theta)}\right), \tag{9}$$

$$\text{Modulus}^2 = \det\frac{\begin{vmatrix} \frac{X\cdot T}{X\cdot X}\left(1+\left(\frac{X\cdot Y}{X\cdot X}\right)^2\right)^{-1}\cos\theta & \frac{X\cdot T}{X\cdot X}\left(1+\left(\frac{X\cdot Y}{X\cdot X}\right)^2\right)^{-1}\sin\theta \\ \frac{Y\cdot T}{Y\cdot Y}\left(1+\left(\frac{X\cdot Y}{Y\cdot Y}\right)^2\right)^{-1}\cos\theta & \frac{Y\cdot T}{Y\cdot Y}\left(1+\left(\frac{X\cdot Y}{Y\cdot Y}\right)^2\right)^{-1}\sin\theta \end{vmatrix}}{\begin{vmatrix} \frac{X\cdot T}{X\cdot X}\left(1+\left(\frac{X\cdot Y}{X\cdot X}\right)^2\right)^{-1}\frac{X\cdot T}{T\cdot T} & \frac{X\cdot T}{X\cdot X}\left(1+\left(\frac{X\cdot Y}{X\cdot X}\right)^2\right)^{-1}\frac{Y\cdot T}{T\cdot T} \\ \frac{Y\cdot T}{Y\cdot Y}\left(1+\left(\frac{X\cdot Y}{Y\cdot Y}\right)^2\right)^{-1}\frac{Y\cdot T}{T\cdot T} & \frac{Y\cdot T}{Y\cdot Y}\left(1+\left(\frac{X\cdot Y}{Y\cdot Y}\right)^2\right)^{-1}\frac{Y\cdot T}{T\cdot T} \end{vmatrix}}. \tag{10}$$

## 3. Precission Study (Bit Width Analysis)

We have designed a specific strategy to define the bit width required in each conceptual stage following this previous algorithm. The basic idea is to transform every calculus in the model, applying a chained process of quantization. For the sake of clarity, if the parameters of the convolution are the bit width of the input $I$, the length of the filter $L$, the mask size $M$, we can compute the output bit width simply shifting the range the output bit width $O$:

$$\text{stage}_n = \frac{2^O}{2^{I+L+M}}\text{stage}_{n-1}. \tag{11}$$

Applying this method in each stage, we obtain a set of values that throw back the transformation between floating point

domain and integer domain, getting a tradeoff between bit width and affordable error.

As the metric error value, we have proposed the most common ones used in the specialized literature, such as Barron's vector [8] and Galvin's couple of metrics [6], where *Vc* and *Ve* are the values of the correct and experimental velocities, respectively, and $g^{\perp}$ is the normal component to the Galvin vector difference:

$$\vec{v} = \frac{(v_x, v_y, 1)}{\sqrt{v_x^2 + v_y^2 + 1}} \rightarrow \psi_{\text{BARRON}} = \arccos(\vec{v}_c \cdot \vec{v}_e),$$
$$\psi_{\text{GALVIN}} = ||\vec{v}_c - \vec{v}_e||,$$
$$\psi_{\text{GALVIN}\perp} = ||(\vec{v}_c - \vec{v}_e) \cdot \hat{g}^{\perp}||. \tag{12}$$

We have also taken into account the simple error measures (absolutes and relatives) relative to modulus and phase:

$$\psi_{\text{MOD}} = \left| ||\vec{v}_e|| - ||\vec{v}_c|| \right|,$$
$$\psi_{\text{FAS}} = |\arctan(\vec{v}_{cy}/\vec{v}_{cx}) - \arctan(\vec{v}_{ey}/\vec{v}_{ex})|. \tag{13}$$

Regarding the stimuli, we have used synthetic compositions of sine waves of different spatial frequencies and the famous stimulus of diverging tree and translating tree [17], commonly used to evaluate optical flow. As a result, we obtain the set of precision parameters that are applied in the model attending to the range of affordable error. Figure 2 shows the bit width of the stages performed in hardware, and Table 1 contains the final values chosen, for an FIR Blur filter length of 5 pixels, FIR spatial filter of 23 pixels, and IIR temporal filter equivalent to an FIR length of 21 frames, with a more detailed analysis available in [14].

Table 1: Parameters of each stage (100% density).

| Stage | Bit width | Error (%) phase mod | |
|---|---|---|---|
| I | $F_1 = 6$<br>$O_{1\_r} = 9$ | 3.64 | 4.95 |
| II | $F_2 = 8$<br>$O_{2\_r\_I} = 9$<br>$O_{2\_r\_II} = 10$ | 4 | 4.95 |
| III | $W_3 = 6$<br>$O_{3\_r} = 10$ | 4.73 | 6.24 |
| IV | $W_4 = 11$<br>$O_{4\_r} = 17$ | 5.31 | 9.01 |
| V | $O_{5\_r} = 12$ | 5.52 | 12.83 |

Being $F_1$ temporal IIR, $F_2$ spatial FIR, $W_3$ steering weight, $W_4$ Taylor expansion, $O_{i\_r}$ bit width output of stage $i$,.

Table 2: Summary of the different implementations.

| Main differences | SW | SmHW | HWbas | HWext |
|---|---|---|---|---|
| Temp filter | FIR | IIR | IIR | IIR |
| Esp. filter | 6 | 6 | 5 | 4 |
| Orientations | 24 | 24 | 18 | 8 |
| Taylor weights | 100% | 65% | 65% | 65% |

## 4. Codesign Process

The system has been designed as a codesign process working with an asynchronous pipeline (micropipeline). The PC feeds the FPGA with a stream of frames through a bank of memory connected to PCI bus. The board takes a continuous stream of pixels at its input (1 byte/pixel); however, we employ 32 bits at the output, coming back to the PC, where they are reordered and written to the hard disk. We have selected Handel C to implement this core, using DK tool [18]. Relating to the prototyping board, an AlphaData RC1000 board has been used, which includes a Virtex 2000E-BG560 chip and 4 SRAM banks of 2 MB each [19]. The memory banks can be accessed both from the FPGA and the PCI bus, Figure 3 showing the communication scheme of the codesign system between the external memory banks, FPGA, and the host platform.

We have implemented a bit width precision defined version of the model, that we called "semihardware" version or SmHW; furthermore the next step is to implement different hardware cores for examining the tradeoff between accuracy and efficiency. We have developed in the FPGA two kinds of platforms that are called "basic" (HWbas) and "extended" (HWext) architectures. The SW version is implemented using the temporal FIR filtering, 24 orientations (each 15°), the SmHW version keeps the same number of orientations, although the implementation of the IIR filters and the Taylor Expansion is not completed (only are used the 65% of the weights). The basic architecture has one less order of spatial differentiation than the versions commented above, and it has only 18 orientations (each 20°), remaining the rest of the parameters constant. The extended architecture has one additional order less than the basic and also decreases in the number of orientations, taking 8 orientations (each 45°). Table 2 summarizes the main differences between these versions attending to the nature of temporal filter, the final spatial derivative order, the number of orientations, and the density of the weights used in the expansion.

## 5. Results

We have analyzed the resources required by the platform and also the number of cycles (NCs) of each stage in Table 3. Every stage belonging to both architectures has been designed as customizable, scalable, and modular.

The basic architecture computes initial blur filter in order to remove aliasing components, IIR temporal filtering that performs the temporal derivatives, FIR spatial filtering, *that is,* spatial derivatives, and steering filtering that project the results onto the whole space (the SW prefix denotes that these stages are performed in software). This architecture contains the processing scheme belonging to most of gradient-based

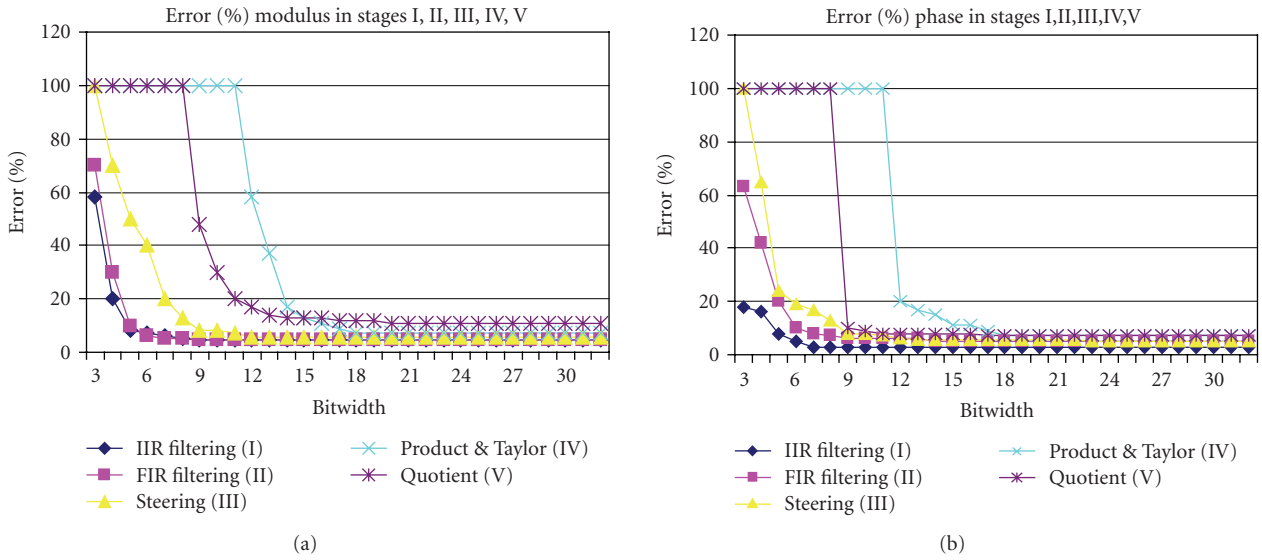


Figure 2: Evaluation of the bit width needed in the modulus (a) and phase (b) converting the data to fixed point.

Table 3: Slices and memory requirements and number of cycles for basic and extended architectures.

| Pipeline stage | Basic architectures | | | Extended architectures | | |
|---|---|---|---|---|---|---|
| | Slices (%) | Block RAM (%) | MC | Slices (%) | Block RAM (%) | NC |
| Blur filter | 289 (2%) | 1% | 4 | 289 (2%) | 1% | 4 |
| IIR temporal filtering | 190 (1%) | 1% | 9 | 190 (1%) | 1% | 9 |
| FIR spatial filtering | 1307 (7%) | 36% | 17 | 1307 (7%) | 36% | 17 |
| Steering | 5961 (31%) | 2% | 15 | 2012 (10%) | 2% | 29 |
| Product and Taylor | SW | SW | SW | 5952 (31%) | 13% | 24 |
| Quotient | SW | SW | SW | 8831 (46%) | 19% | 21 |

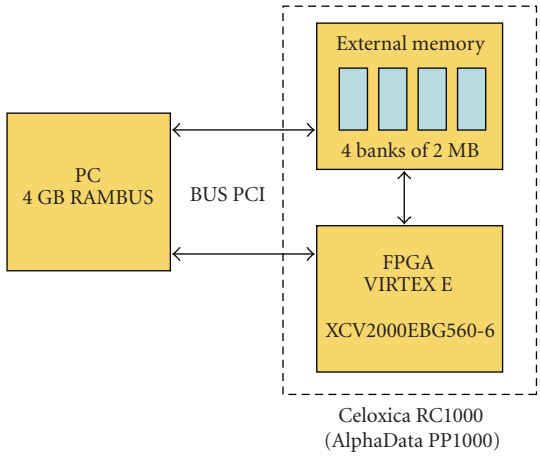


Figure 3: Scheme of the communication process.

optical flow models, thus it could be considered as a motion preprocessor [15, 16]. The extended architecture is able to cover more stages and is focused in the specific McGM algorithm, implementing all the stages commented previously, plus a Taylor expansion, Taylor product (their derivative products), and the quotient stage as shown in Figure 4.

## 5.1. Hardware Cost

The basic architecture consumes 41% of the board slices, with every stage being performed with parameter values very close to the original model (derivatives in $x$ up to order 5, 18 orientations in the steering stage), implementing 4 stages. Nevertheless, the extended architecture requires 97% of the development board.

## 5.2. Performance

Related to the number of cycles, we have noted the Xilinx timing analyzer tool [20] to be very conservative; thus we can increase the throughput around 25%–35% if we clock the system manually from the values obtained. The slower stage in the basic architecture is the FIR filtering, while the last stages designed need the maximum number of Block RAMs and slices due to the computation being performed replicating the spatial convolution (FIR filter) concurrently

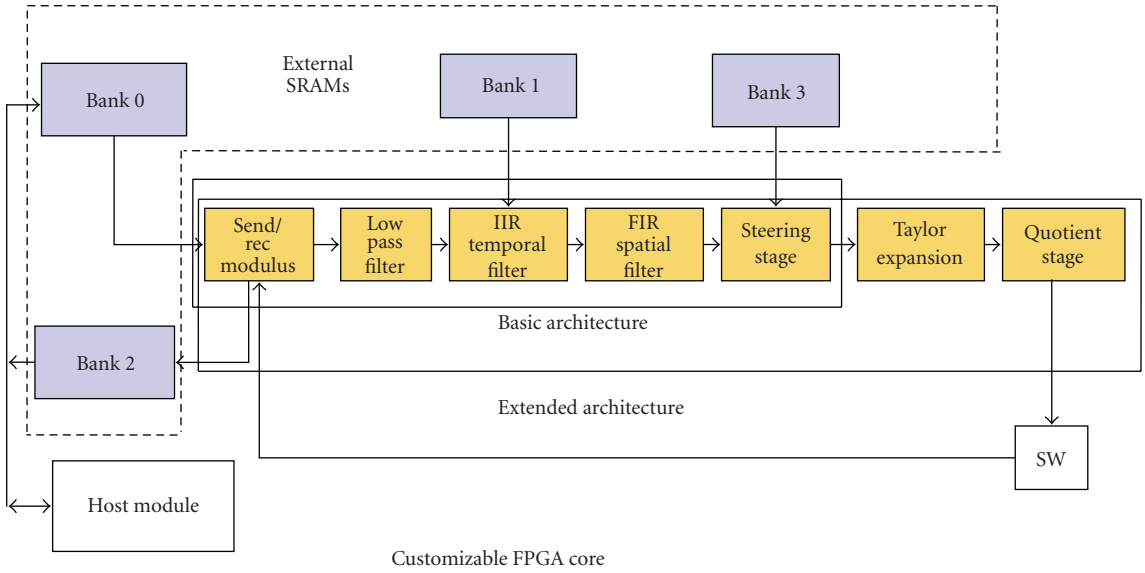


Figure 4: Scheme of the two architectures working with an asynchronous pipeline.

for $n$ orientations until order $m$ in $x$. Nevertheless, in the extended architecture we must keep resources for the next stages, removing some contributions and parallelizing the processing scheme in discrete groups, which replace the whole group entirely concurrently. For instance, the steering stage is performed with fewer terms and with reduced parallelization level, requiring almost the double of cycles. Applying this strategy of keeping enough resources in the prototyping board, we can extend the model to additional stages. We can see in Figure 4 the global codesign scheme and the two architectures involved, representing the transactions between external RAM (grey blocks) and the stages. The stage corresponding to IIR filter has to keep 3 frames using the bank number one, the steering stage reads the orientation weights from bank number three, and the send/receive modulus connects the input/output data between the FPGA and the host system via the PCI bus using DMA transfer. Figure 5 shows the performance for the whole systems using chained stages, attending to the pixel/seconds processed, concluding that it is possible to compute 177 frames/second with a resolution of $128 \times 96$ pixels in the basic architecture, and 37.9 frames/second for the extended architecture.

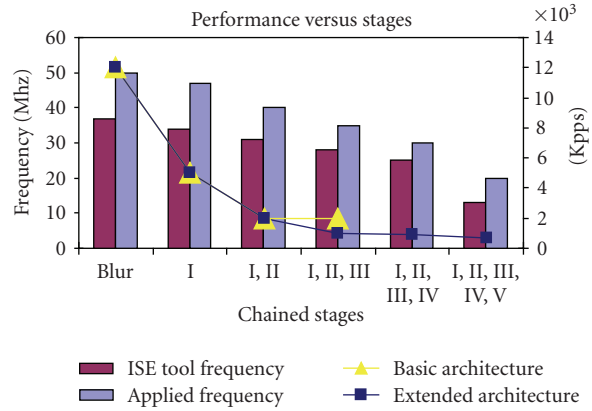


Figure 5: Throughput of the pipeline (Kpps) and frequency corresponding to basic and extended architectures.

## 5.3. Quality of the Results

An accuracy analysis has been carried out, being possible to examine the quality of the results under different transformations and metrics, as we can see in Figure 6. The phase and modulus metrics (difference between values) show a good behavior regarding the implementation changes, while Barron's metric seems to go well keeping the proportion accuracy under changes, but Galvin and Galvin perpendicular metrics suffer with the implementation change from SW to HW. It is due to the nature of the metric, which gives an idea about how the algorithm copes with the Aperture Problem [8], this topic being discussed in previous work [14]. Despite restricting every version in terms of precision parameters one step further until finally taking the extended architecture, in general the error values are delimited reasonably.

## 5.4. Some Visual Results

Figure 7 shows some visual results corresponding to different versions of our system, concretely SW versus HWbas. It can be noted that while the SW version keeps a calculus density close to 100% (middle row in Figure 7), HWbas loses some points due to precision bit width (bottom row in Figure 7), *that is*, the bit number of the parameters in each stage. The input sequence, called diverging tree (upper row in Figure 7) has a divergent structure where the modulus is supposed to vary poorly and the phase is changing regularly over 360°. Since we are working with synthetic sequences, we can estimate the error without any ambiguity. Also we have used

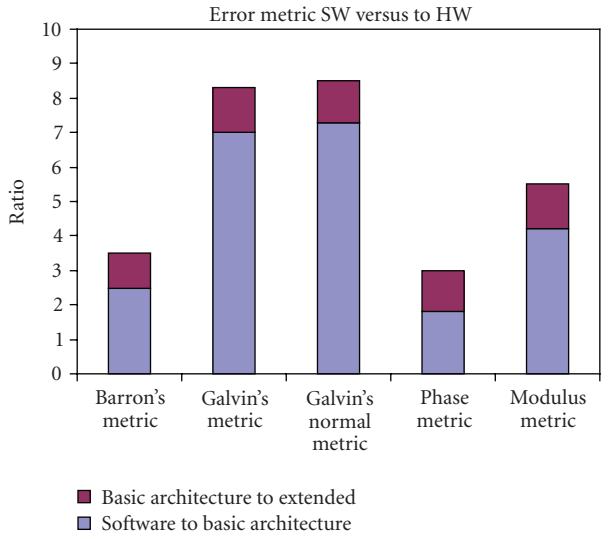


Figure 6: Quality of different implementations.

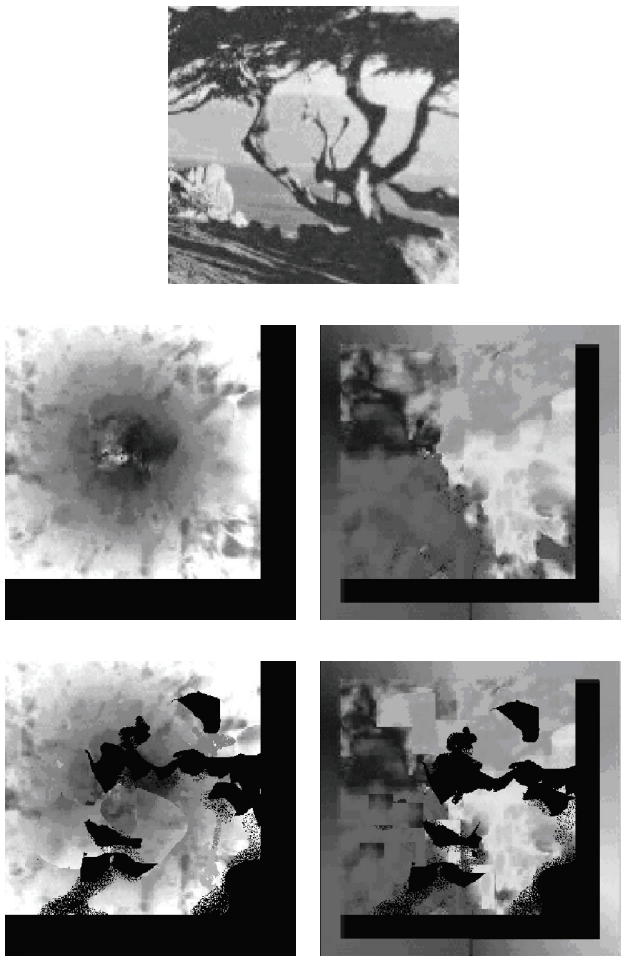

Figure 7: Some visual results corresponding to the software version versus the basic architecture (diverging tree sequence). Left hand indicates velocity modulus and right hand velocity phase.

Table 4: Summary of the different implementations for the Yosemite sequence. NP means not provided.

| Models | Average error | Standard deviation | Density |
|---|---|---|---|
| Described here (HWbas) | 5.5° | 12.3 | 100% |
| Described here (HWext) | 7.2° | 11.1 | 100% |
| Described here (HWext) | 6.1° | 6.2 | 60% |
| Described here (HWext) | 4.3° | 3.1 | 20% |
| Díaz et al. [21] | 18.30° | 15.8° | 100% |
| Díaz et al. [22] | 7.6° | NP | <55% |
| Martín et al. [25] | NP | NP | <50% |

the translating tree sequence, where modulus is changing from left to right and the phase is practically almost the same.

## 6. Comparison with other Approaches

There are other gradient optical flow models implemented in hardware [21, 22], belonging to the Lucas and Kanade algorithms [23] and to Horn and Schunk approximations [24, 25], while in Table 4 we can see the average error for different metrics, although only we compare the Barron's metric since the cited authors do not provide other measurements.

Attending to the errors, our implementation provides better results than the other approaches, even with calculation density 100%. Nevertheless, the final results are improved if the points where the scene structure changes, *that is,* points smaller than a determinate temporal derivative, are filtered. This is caused by a least squares process being performed at the end of the algorithm for calculating the modulus and the phase final values. The points filtered would force the slope of the linear regression to be very small, with the value of velocity is almost null.

Regarding throughput, we are able to calculate more than 2000 Kpixel/s in the basic Architecture and about 1000 Kpixel/s in the extended. It would locate our implementation between those in [23, 26], enough for real-time purposes, although it could be improved using a board with more resources that is used here and increasing the parallelism level.

The error using the diverging and translating tree sequences [17] is shown in Figure 7, and it is obtained with different metrics regarding the expressions (12)-(13).

## 7. Conclusion

We have developed an FPGA-based implementation of a bioinspired robust motion estimation system with an associated complexity higher than those found in other gradient-based models commonly used in the literature. The study of precision calibrates the model and adjusts the bit width needed for keeping a tradeoff solution between accuracy and efficiency, acting as a bridge between software and hardware and estimating the cost to convert every stage from floating to fixed point. Taking the results from this precision study, different hardware moduli have been designed, organizing

this in two high parallelized architectures. The first one, referred to as basic architecture and common to optical flow gradient models, is a superconvolutive processor orientated along multiple angles. It could be used as a starting point for many computer vision algorithms, not necessarily restricted to the motion estimation field, like change detection, stereo, or even biometry techniques such as real-time signature recognition. The second architecture, called extended, is focused in the Mutichannel Gradient Model, and includes the truncated Taylor expansion representation of space temporal information of the scene, its three differentiates respect space and time, and the quotients of the products of these last functions. The rest of the stages, called velocity primitives, corresponding to the expressions (8)-(9) are performed in software in the framework of a codesign process, where the final modulus value is a quotient of determinants and the final phase is an arctangent. This extension can be implemented using a board with more resources than the VIRTEX 2000 E and, depending on the accuracy required, using a structure based on LUTs or implementing a CORDIC core. Both architectures are scalable and modular, and also extensible to one device with more resources that our prototyping platform.

Additionally, the resources consumed have been evaluated as well as the throughput and the accuracy of the designed coprocessors. All models come forward with asynchronous segmented architectures (micropipelines). Regarding quality, the average error has been compared using Barron's metric, since other authors do not provide results with other metrics; also the throughput of the design has been compared with other implementations. This work generates dense optical flow maps up to 80 frames/second and 185 frames/second for a resolution of $128 \times 96$ in the extended and basic architectures, respectively. The present contribution opens the door to embed complex bioinspired systems that require a huge quantity of computation. We are currently improving the system to extend the model to a fully stand alone platform also to deal with stereo vision. Several application fields are though to use it, such as motion illusion detection or video compression.

## Acknowledgments

This work was partially supported by Projects TEC2007-68074-C02-01/MIC, TIN2005-05619-2004-07032 (Spain), EU Project DINAM-VISION (DPI2007-61683), and an EU "Marie Curie" Fellowship (QLK5-CT-1999-50523). The authors would like to thank the anonymous reviewers for their insightful suggestions and Professor Johnston and Dr. Dale, from the Vision Group at University College London, for their help and support during this research.

## References

[1] C. Mead, "Neuromorphic electronic systems," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1629–1636, 1990.

[2] C. Mead, *Analog VLSI and Neural Systems*, Addison-Wesley, Reading, Mass, USA, 1989.

[3] H.-S. Oh and H.-K. Lee, "Block-matching algorithm based on an adaptive reduction of the search area for motion estimation," *Real-Time Imaging*, vol. 6, no. 5, pp. 407–414, 2000.

[4] C.-L. Huang and Y.-T. Chen, "Motion estimation method using a 3D steerable filter," *Image and Vision Computing*, vol. 13, no. 1, pp. 21–32, 1995.

[5] S. Baker and I. Matthews, "Lucas-Kanade 20 years on: a unifying framework," *International Journal of Computer Vision*, vol. 56, no. 3, pp. 221–255, 2004.

[6] B. McCane, K. Novins, D. Crannitch, and B. Galvin, "On benchmarking optical flow," *Computer Vision and Image Understanding*, vol. 84, no. 1, pp. 126–143, 2001.

[7] H. Liu, T.-H. Hong, M. Herman, T. Camus, and R. Chellappa, "Accuracy vs efficiency trade-offs in optical flow algorithms," *Computer Vision and Image Understanding*, vol. 72, no. 3, pp. 271–286, 1998.

[8] J. L. Barron, D. J. Fleet, and S. S. Beauchemin, "Performance of optical flow techniques," *International Journal of Computer Vision*, vol. 12, no. 1, pp. 43–77, 1994.

[9] A. Johnston and C. W. G. Clifford, "A unified account of three apparent motion illusions," *Vision Research*, vol. 35, no. 8, pp. 1109–1123, 1995.

[10] A. Johnston and C. W. G. Clifford, "Perceived motion of contrast-modulated gratings: predictions of the multi-channel gradient model and the role of full-wave rectification," *Vision Research*, vol. 35, no. 12, pp. 1771–1783, 1995.

[11] A. Johnston, P. W. McOwan, and C. P. Benton, "Robust velocity computation from a biologically motivated model of motion perception," *Proceedings of the Royal Society B*, vol. 266, no. 1418, pp. 509–518, 1999.

[12] P. W. McOwan, C. Benton, J. Dale, and A. Johnston, "A multi-differential neuromorphic approach to motion detection," *International Journal of Neural Systems*, vol. 9, no. 5, pp. 429–434, 1999.

[13] A. Johnston, P. W. McOwan, and C. P. Benton, "Biological computation of image motion from flows over boundaries," *Journal of Physiology-Paris*, vol. 97, no. 2-3, pp. 325–334, 2003.

[14] G. Botella, *Robust optical flow implementation in reconfigurable hardware*, Ph.D. thesis, University of Granada, Granada, Spain, 2007, ISBN 978-84-338-4381-4.

[15] G. Botella, E. Ros, M. Rodríguez, A. García, and S. Romero, "Pre-processor for bioinspired optical flow models: a customizable hardware implementation," in *Proceedings of the 13th IEEE Mediterranean Electrotechnical Conference (MELECON '06)*, pp. 93–96, Málaga, Spain, May 2006.

[16] G. Botella, E. Ros, M. Rodríguez, and A. García, "Bioinspired robust optical flow in a FPGA system," in *Proceedings of the 32nd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO-SEAA '06)*, Dubrovnik, Croatia, August-September 2006.

[17] The input sequence was created by David Fleet at Toronto University and can be obtained from: ftp://ftp.csd.uwo.ca/pub/vision/TESTDATA.

[18] Handel-C Languaje reference manual and DK tool. Celoxica company, 2007.

[19] AlphaData RC1000 product, 2006, http://www.alpha-data.com/adc-rc1000.html.

[20] "Timing Analysis and Optmization of Handel-C designs for Xilinx chips," Celoxica application note AN 68 v1.1, 2005.

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

[22] J. Díaz, E. Ros, R. Rodriguez-Gomez, and B. del Pino, "Real-time architecture for robust motion estimation under varying illumination conditions," *Journal of Universal Computer Science*, vol. 13, no. 3, pp. 363–376, 2007.

[23] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of DARPA Image Understanding Workshop*, pp. 121–130, Washington, DC, USA, April 1981.

[24] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, no. 1–3, pp. 185–203, 1981.

[25] J. L. Martín, A. Zuloaga, C. Cuadrado, J. Lázaro, and U. Bidarte, "Hardware implementation of optical flow constraint equation using FPGAs," *Computer Vision and Image Understanding*, vol. 98, no. 3, pp. 462–490, 2005.

[26] Z. Wei, D.-J. Lee, and B. E. Nelson, "FPGA-based real-time optical flow algorithm design and implementation," *Journal of Multimedia*, vol. 2, no. 5, pp. 38–45, 2007.