Fine grain pipeline systems for real-time motion and stereo-vision computation

Javier Díaz,* Eduardo Ros,
Alberto Prieto and Francisco J. Pelayo

Department of Arquitectura y Tecnología de
Computadores E.T.S.I. Informática,
C/ Periodista Daniel Saucedo, s/n. E-18071 Granada, Spain
Fax: +34-958-248-993
E-mail: jdiaz@atc.ugr.es E-mail: eros@atc.ugr.es
E-mail: aprieto@ugr.es E-mail: fpelayo@ugr.es
*Corresponding author

Abstract: Image processing systems require high computational load that motivates the design of specific hardware architectures in order to arrive at real-time platforms. We adopt innovative design techniques based on the intensive utilisation of the inherent parallelism available on devices based on reconfigurable hardware. We customise fine-grain pipelining and superscalar units to implement specific computing architectures for motion and stereo-vision computing circuits. This high parallelism level allows us to achieve a high data throughput (one pixel feature estimation per clock cycle). This paper extensively uses these techniques for designing high performance image processing systems which fit early cognitive vision models specifications. Furthermore, it highlights the necessity of on-chip integration mechanisms, since the data throughput (bandwidth requirements) of the full system requires a very large bandwidth.

Keywords: FPGAs; pipelined techniques; image processing algorithm parallelisation; data flow architectures; optical flow; stereo vision; high performance customised architectures.

Reference to this paper should be made as follows: Díaz, J., Ros, E., Prieto, A. and Pelayo, F. (2007) 'Fine grain pipeline systems for real-time motion and stereo-vision computation', *Int. J. High Performance Systems Architecture*, Vol. 1, No. 1, pp.60–68.

Biographical notes: Javier Díaz Alonso received an MS in Electronics Engineering in 2002 and a PhD in Electronics in 2006 from the University of Granada (Spain). Currently, he is an Assistant Lecturer in the Department of Computer Architecture and Technology at the University of Granada. His main research interests include cognitive vision systems, high performance image processing architectures and embedded systems based on reconfigurable devices. He is also interested in spiking neuron simulation tools, biomedical devices and robotics.

Eduardo Ros Vidal received his PhD in 1997 from the University of Granada. Currently, he is an Associate Professor in the Department of Computer Architecture and Technology at the same university. Currently, he is the responsible researcher at the University of Granada of two European projects related with bioinspired processing schemes. His research interests include simulation of biologically plausible processing schemes, hardware implementation of digital circuits for-real time processing in embedded systems and high performance computer vision.

Alberto Prieto received a BSc in Physics (Electronics) in 1968 from the Complutense University in Madrid. In 1976, he completed his PhD at the University of Granada. From 1971 to 1984, he was Founder and Head of the Computing Centre and he headed Computer Science and Technology Studies at the University of Granada from 1985 to 1990. Currently, he is a Fulltime Senior Lecturer and Head of the Department of Computer Architecture and Technology. and also co-Author of four text-books published by McGraw-Hill and Thomson editorials. He has co-edited five volumes of the LNCS and has been co-Author of more than 250 papers. His area of research primarily focuses on intelligence systems.

Francisco J. Pelayo Valle received a BSc in Physics in 1982, an MSc in Electronics in 1983 and a PhD in 1989, all from the University of Granada, Spain. Currently, he is a Professor in the Department of Computer Architecture and Technology of the same University. He has worked in the areas of VLSI design and test, artificial neural networks and fuzzy systems. His current research interest lies in the fields of hardware and hw-sw design of bioinspired processing and control systems, robotics and neuroengineering.

1 Introduction

Visual perception is a complex process that transforms (translates) signals (images) into cognitive information. Computer vision is a mature research field that has evolved significantly in the last decade and that studies how the translation process takes place. According to Ratha and Jain (1999), the main problem which computer vision tries to solve is the understanding of the content of different kinds of images: static (e.g. pictures) or dynamic (video sequences); two-dimensional or three-dimensional (e.g. images from a 3D scanner) and from one source (monocular) or from multiple sources (e.g. binocular stereo images).

Nevertheless, after many years of research work, we are still far away from achieving outstanding vision skills similar to biological systems. Though significant results have been reported in specific domains, where the properties of the expected content lie within a certain range (e.g. edge detection, motion processing, stereo vision, etc.), there are no general-purpose image interpretation applications yet or they have a limited performance.

Though, we have focused in this paper at the early computing stage of the vision system, in order to fully understand the vision process, we should try to schematically represent the main levels in which the vision process can be structured. The complexity of vision systems comes out from the multiple levels of abstraction that should be taken into account to achieve the *scene understanding*. Vision researchers tend to classify vision algorithms and representations into three levels: low (sensory), intermediate (symbolic) and high (knowledge based) (Díaz, 2006; Rares et al., 1999; Weems, 1991).

1 *Low level vision* deals with local operations and is composed of spatio-temporal filters. Biological systems use cells whose receptive fields project onto the retinas. From a set of basic spatio-temporal filters of different size and temporal characteristics, vision models generate information about stereopsis, motion in the scene, local contrast, etc.

2 *At intermediate level vision*, the integration and segmentation mechanisms take place. This allows efficient and constructive combination of different visual modalities (motion, stereo, orientation, etc.) or segment emerging abstracted information such as Independent Moving objects (IMOs).

3 *High level vision*: this is a very high level processing stage where scene interpretation is performed through more specific subtasks, such as object recognition, effects prediction, comparison with already perceived scenarios, etc.

For a computer architect, a vision system presents therefore three distinct sets of requirements. At each level, their requirements can be broken into three categories: computation, communication and control which correspond to the different levels of abstraction of the vision systems. It is important to note that early cognitive vision is inherently dense, that is, that it requires processing each pixel in the scene, while higher level tasks deal with discrete entities with well defined semantic meaning. Furthermore, medium and high levels vision algorithms are still at development stages and therefore, the best architectural processing strategy should be able to maximise generality and flexibility at this processing level.

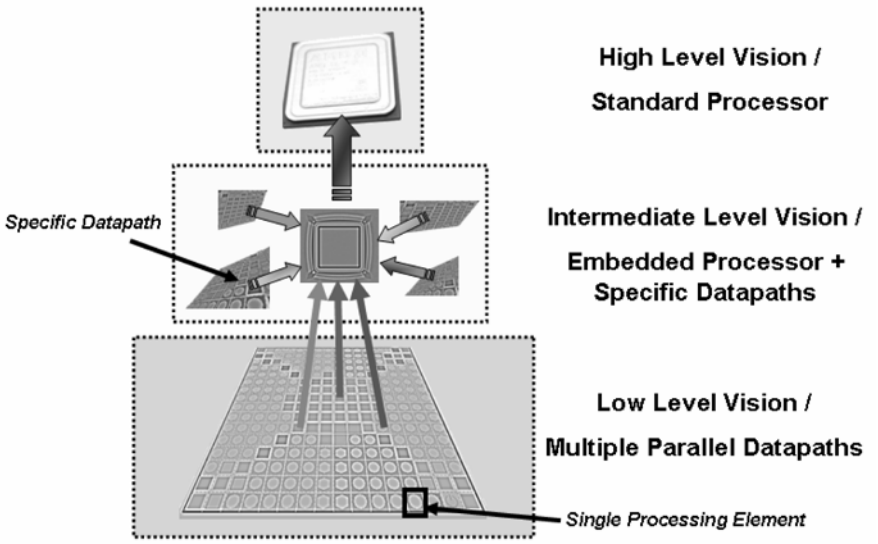
Clearly, no homogeneous processor can simultaneously support all levels with optimal efficiency and this is one of the reasons that explains why computer vision has not advanced as further as expected. Although current standard processors have significantly improved their processing power in these last years, they are not very efficient for real-time complex vision models. They are well suited for high level vision tasks, because this level works with discrete data trying to maximise the generality and flexibility of the models. On the other hand, low level vision tasks require massive parallelism for regular pixel-wise computation. Sequential processors (even using current multicore architectures) are too constrained to achieve real-time.

Although there is no general agreement about how to structure such a complex system, we propose a computing architecture as presented in Figure 1.

In this contribution, we will focus on the way of using current digital technology for developing architectural solutions for two well known low level computer vision tasks: motion and binocular depth estimation. Typically, real-time systems work with limited spatial or temporal resolutions (Bruhn et al., 2005; Darabiha et al., 2006; Focus Robotics, 2006; Martín et al., 2005) which significantly restricts system capabilities, specially if compared to biological systems which have very large visual resolution and motion perception (Curcio et al., 1990; García-Pérez and Peli, 2001; Goodchild et al., 1996; Watson et al., 1986). We consider that a stereo image resolution of 1024×1024 at 30 fps and motion estimation of 512×512 at 100 fps are the primary specifications in order to develop powerful vision systems based on these features. In previous works, we presented motion and disparity estimation datapaths separately (Díaz et al., 2007, 2006b) and showed that this performance is feasible. This contribution goes one step further, showing how very complex processing cores can be integrated without degrading the system performance thanks to an efficient management of system resources. To the best of our knowledge, this is the first time that a real-time embedded device for computing stereo and motion information concurrently is presented.

This paper is structured as follows. In Section 2, we will describe how to deal with bioinspired massive parallel computations thanks to finely defined datapath structures. Specific solutions are required in order to maximise results for low vision tasks. This design strategy will be illustrated in Section 3 with applications to motion and stereo disparity computation. Section 4 focuses on the properties and requirements of the whole system and finally, Section 5 presents the main conclusions achieved in this contribution.

Figure 1 Schematic of the vision system architecture. Each of the three different computer vision levels suits better different computing devices. High level vision fits well on standard processors. Intermediate level vision shares requirements from the upper and lower levels. Therefore, a hybrid software–hardware platform is required. This scheme can be efficiently developed using current FPGAs devices which make feasible to design customised embedded processors and specific datapaths. Finally, low level vision fits quite well on highly parallel fine grain parallel architectures which are able to compute in a very parallel way multiple vision features, each of them computed in a single processing element



2 Long pipelines as an architectural solution for bioinspired computing structures

In the late 1980s, Carver Mead coined the term *neuromorphic* to describe Very Large Scale Integration (VLSI) systems containing electronic analog circuits that mimic neurobiological architectures present in the nervous system (Mead, 1989). Recently the term *neuromorphic* has been used to describe analog, digital or mixed-mode analog/digital VLSI systems that implement models of bioinspired neural systems (for different tasks such as vision, motor control or sensory processing) as well as software algorithms.

These approaches assume the hypothesis that in biological systems, the computing structure is much related to the computation that is implemented. Therefore, *neuromorphic* engineers try to mimic the neural topologies in which the computations take place to investigate at which level they are related. In a similar way as *neuromorphic* Engineers, we consider that engineering processing architectures can benefit considerably by mimicking nature. However, we consider that neurons based topologies are not completely well suited for our aim datapaths, since the physical principles and constraints upon which biological tissues are based are very different from those characteristically used in electronic technology.

Our approach tries to abstract the key-functional principles that contribute to the outstanding performance of biological systems, to formulate them in a more mathematical way and to translate them into silicon using well known designing techniques such as time-slicing, pipelining and superscalar units. For instance, bioinspired approaches for motion computation are usually based on energy models (Mota et al., 2005; Simoncelli, 1993). These models are based on a neural population of cells, each of them tuned to a different velocity. The velocity estimation process relies in the estimation of the winner cell. Nevertheless, as demonstrated by Simoncelli (1993), this process is fully equivalent to a least squares fitting process over a restricted cell population. This is used for instance in gradient methods as described by Barron et al. (1994) and based on that, motion can be computed from a more mathematical point of view which can be easier to implement on standard digital hardware. Then, taking full advantage of the fast clock rates and design techniques available on current technology, this mathematical scheme allows overcoming the restricted parallelism affordable in silicon devices.

In particular, one innovative aspect is the utilisation of very deep pipelines. As showed in Hartstein and Puzak (2002), different applications fit in different parallelism levels and optimal pipeline depth. A proper selection of the low level vision algorithms and tasks translates into hazards-free architectures which allow very deep pipelines with outstanding data throughput. Most of the low level vision tasks mimic in more or less detail biological nets where inhibition/excitation mechanisms take place instead of branch conditions. The main limitation consists on how to efficiently plan the pipeline data feed to avoid stopping the processing modules, taking into account that most of the algorithms use local data and can therefore take full advantage of small and local memories.

This design methodology requires that data arithmetic representation and bit-width should be carefully chosen. For instance, by using fixed point arithmetic, data bit-width linearly increases with additions and subtractions, doubles with multiplications and depending on target precision, significantly changes with divisions and trigonometric operations. In Díaz (2006), a new Matlab based tool called *MCode for DSP Analyser* is presented. This tool has been satisfactorily utilised to help the designer to properly balance the resources consumption versus performance trade-off.

We illustrate here two examples of this design methodology. We have developed a system for real-time processing of optical flow and depth estimation that is able to extract these modalities at high frame rates with large image resolution. Pipelines with both more than 70 stages and a high parallelism level have been accomplished in current reconfigurable devices to achieve an outstanding computing performance, as shown in Sections 3 and 4. The effect of this design methodology is quite relevant even when using low clock rates. In the optimal case, a system running, for instance, at 50 MHz, should be able to estimate one feature per clock cycle. This translates into a computing power of 50 million estimations per second. We can adapt the clock rate according to the target application by using the rate which fits our target application or reduce the parallelism level by sharing processing units, thus reducing the resources consumption.

Though reconfigurable devices are able to achieve a very high parallelism level, their physical implantation is often constrained due to the limited transmission bandwidth available from external memory devices. This is usually one of the important bottlenecks for FPGA processing capability. An efficient management of external (large) and internal (small) memory resources is critical to achieve the maximum system performance.

3 Local methods for solving the image correspondence problem

The problem of image correspondence is a complex task which deals with matching image objects in different images taken under different conditions (for instance, taken from different points of view or in a different instant). Hereon, we will only consider pixel-wise features, which are the primitives presented at early vision stages. In a very coarse way, these methods can be classified into local or global methods (see for instance Brown et al. (2003) for the case of disparity estimation).

On the one hand, global correspondence methods usually require storing the whole image into memory resources, which is not affordable into embedded memory of current silicon devices and therefore requires the utilisation of external memories. Furthermore, they are generally based on iterative operations which require a large number of accesses. This degrades significantly the performance of these approaches, which usually work at low clock rates (compared to standard PCs) and penalties its skills for real-time operations.

On the other hand, local methods only use information available on small patches of the image which make feasible to use caching methods in optimising the system throughput. This scheme fits quite well on high parallel devices such as FPGAs and therefore, we will focus herewith on two well-known and efficient methods for image motion and disparity.

3.1 Motion computation based on least squares fitting of image derivatives

The method for computing motion from a sequence of images was proposed by Lucas and Kanade (1984) (*L&K*) and has been highlighted by several authors as a good alternative in terms of accuracy and efficiency trade-off (Barron et al., 1994; Liu et al., 1998).

The key-idea is based on the assumption of luminance constancy over time. Luminance is approximated by its Taylor series expansion and truncated in the first order. This becomes an ill-posed equation usually known as first order constraint, which is expressed in Equation (1).

$$\left[\nabla_{xy} I(x,y,t)\left(v_x, v_y\right) + I_t(x,y,t)\right] = 0 \tag{1}$$

In Equation (1), V_x and V_y stand for the two components of each pixel velocity; I_t represents the temporal derivative of the sequence and ∇_{xy} stands for the spatial $(x - y)$ gradient operator which computes the image derivatives I_x and I_y. Lucas and Kanade (19984) solved this problem by applying least squares fitting on a local neighbourhood Ω of each pixel, typically of 5×5 pixels.

The final velocity estimation is computed from Equation (2), where W stands for the weights used to properly integrate estimations over the neighbourhood Ω (typically a five taps Gaussian kernel).

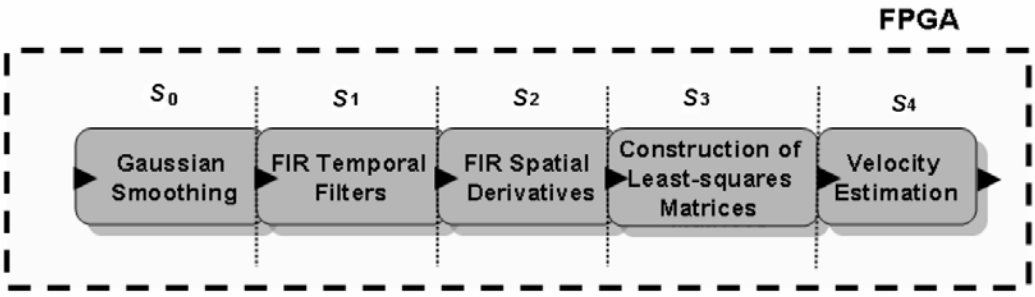
$$\begin{aligned} V_x &= A_{xy}A_{yt} - A_{yy}A_{xt} \\ V_y &= A_{xy}A_{xt} - A_{xx}A_{yt} \\ A_{kj} &= \sum_{\Omega} W^2 I_k I_j \end{aligned} \tag{2}$$

According to Equation (2), the main steps required for this model can be summarised as follows:

- S_0: spatio-temporal image smoothing based on FIR Gaussian filters.
- S_1: temporal smoothing and derivative operations.
- S_2: spatio-temporal image derivatives computation from temporal data based on first order Gaussian kernels.
- S_3: image derivatives combination on a local neighbourhood to construct the basic elements for least squares fitting (A_{kj} elements on Equation (2)).
- S_4: estimation of local velocity values.

The corresponding data-flow is presented in Figure 2. The whole architecture is described in more detail in Díaz et al. (2007, 2006b). Each image operation is decomposed into simple microoperations (such as addition, subtraction, multiplexers, etc.), performed by different Simple Processing Stages (SPS) and structured in a long pipeline datapath. This is similar to systolic circuits, but in our system, each SPS is different and therefore, regular structures such as systolic systems are not feasible.

Figure 2 Simplified schematic of the motion extraction system. The coarse stages of a Gradient based optical flow computing circuit are represented. Each stage, S_i, represents a coarse description of the pipelined and superscalar architecture developed for motion computation



3.2 Phase difference computation of binocular images for depth estimation

Most of the real-time systems for image disparity estimation are based on block-matching correspondence techniques (Brown et al., 2003) with some exceptions, such us Darabiha et al. (2006). We have focused on the phased-based computing model proposed by Solari et al. (2001). In a first approach, the positions of the corresponding points are related by a 1D horizontal shift, the disparity, along the direction of the epipolar lines. Disparity can be estimated in terms of phase differences in the spectral components of the binocular pair of images as described in Solari et al. (2001). Spatially localised phase measurements can be obtained by filtering operations with a pair of quadrature bandpass Gabor filters.

The advantage of phase-based approaches, according to different authors, consists of their robustness to luminance variations and cameras imbalance problems. Furthermore, they lead to a better behaviour against affine transformations (for instance, due to different camera perspectives) (Cozzi et al., 1997; Fleet and Jepson, 1993). The phase difference is computed from Equation (3):

$$\delta(x) = \frac{\phi^{\mathrm{L}}(x) - \phi^{\mathrm{R}}(x)}{k(x)} \approx \frac{1}{k_0} \arctan 2\left(\frac{C^{\mathrm{R}}S^{\mathrm{L}} - C^{\mathrm{L}}S^{\mathrm{R}}}{C^{\mathrm{L}}C^{\mathrm{R}} + S^{\mathrm{L}}S^{\mathrm{R}}}\right) \qquad (3)$$

where we note with ϕ^{L} and ϕ^{R} the left and right local image phases. C^{R} and C^{L} correspond to the values of left and right image pixels after convolving with the even part of the quadrature filter and S^{R} and S^{L} to the odd quadrature filter outputs. The arctan 2 stands for the principal part of the argument (i.e. the argument belonging to $[-\pi, \pi]$). Finally, $k(x)$ is the average instantaneous frequency of the bandpass signal, which can be approximated by k_0, the quadrature filter peak frequency.

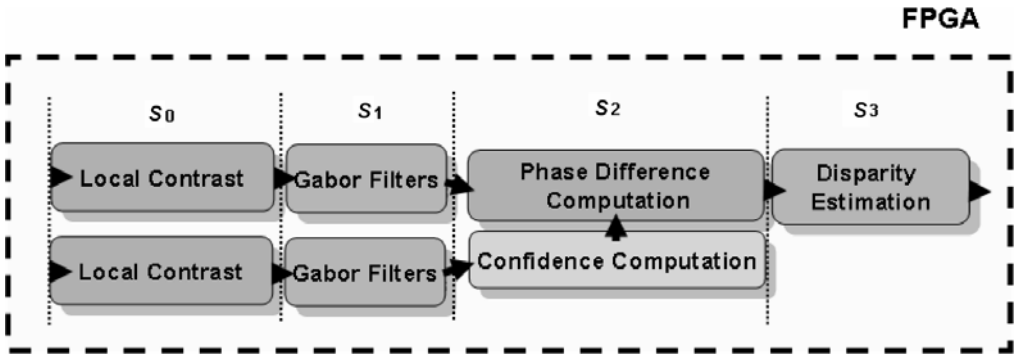
It should be noted that Equation (3) does not require the explicit calculation of the left and right phases and thus, we can compute the phase difference directly by avoiding the problem of phase wrapping.

To address the hardware implementation of this approach, the basic steps can be summarised as follows:

- S_0: DC component image removal using the local image contrast $I - I_{\mathrm{mean}}$ operator for the even Gabor filter.
- S_1: even (C) and odd (S) 1D Gabor filtering of left and right images.
- S_2: direct phase-difference calculation from (3).
- S_3: disparity computation, assuming $k(x) \approx k_0$ (Figure 3).

The corresponding data-flow is presented in Figure 3. The whole architecture is described in more detail in Díaz et al. (2007). In a similar way to the previous motion estimation system, each image operation is decomposed into simple microoperations, computed by SPS implemented in dedicated micropipeline stages. Note that there are some

Figure 3 Simplified schematic of the disparity computation system. The basic stages of the circuit S_i, represent a coarse description of the pipelined and superscalar architecture (some of them duplicated because the system is fed from 2 cameras). Note that there is a new module in stage S_2 which decides if each disparity estimation passes or is rejected because it corresponds to a low contrast image area



common operations for left and right images (stages S_0 and S_1) which can broadly speaking be described as a two-ways scalar unit. This circuit parallel expansion is motivated by the hard constraints required to keep the throughput of one pixel estimation for each clock cycle.

4 High performance motion and disparity estimation architecture

We have implemented a high performance embedded system for motion and disparity estimation using reconfigurable hardware (FPGAs). This technology fits very well the requirements of low level vision systems as described in the introduction section and illustrated in Figure 1. Furthermore, current FPGAs are heterogeneous devices which do not only contain logic gates, but also distributed embedded memories and MAC units. Massive parallel simple units can be easily developed in these devices using the available logic gates. Caching techniques are implemented taking full advantage of the distributed memories. Besides, high efficient DSP operations can be defined thanks to the embedded multipliers and MAC units.

Our system scheme is outlined in Figure 4. We use two input cameras whose images are digitised and stored into external memories by using a dual camera frame-grabber circuit. This frame-grabber is used to feed the motion and disparity datapaths. System output can be visualised by using a VGA output. The user interface units are used to transfer the user commands. For instance, this interface allows to modify confidence circuit thresholds or to switch the visualisation output (stereo or motion).

Next sections deal with the system resources, the results and a qualitative evaluation of the whole architecture complexity.

4.1 Feasibility of massive parallel circuits utilisation

In the previous works, we presented the different processing units separately (Díaz et al., 2007, 2006b). However the integration of these complex processing cores without degrading the performance is not straightforward. This contribution shows that this integration is feasible and that fine pipeline operations can be scheduled in a non-interfering way to keep our architectural specifications, one estimation per clock pixel. In order to achieve this parallelism level, we should be able to provide arbitration circuits to the different SPS to properly feed each processing unit. In this line, a fundamental point is to control the memory resources efficiently, which is performed using a high performance MMU described in Díaz et al. (2006a). This allows to implement the whole architecture by using only three external memory banks.

The whole system is a very complex and parallel architecture. To provide an idea of the complexity, Table 1 presents the number of SPS which are required for the motion and stereo processing cores. For each coarse stage S_i described in Sections 3.1 and 3.2,

Figure 4 Binocular stereo and motion estimation system architecture. External memories (3 banks), cameras and VGA output are schematically represented. Dark boxes represent the image processing cores developed for stereo and motion estimation

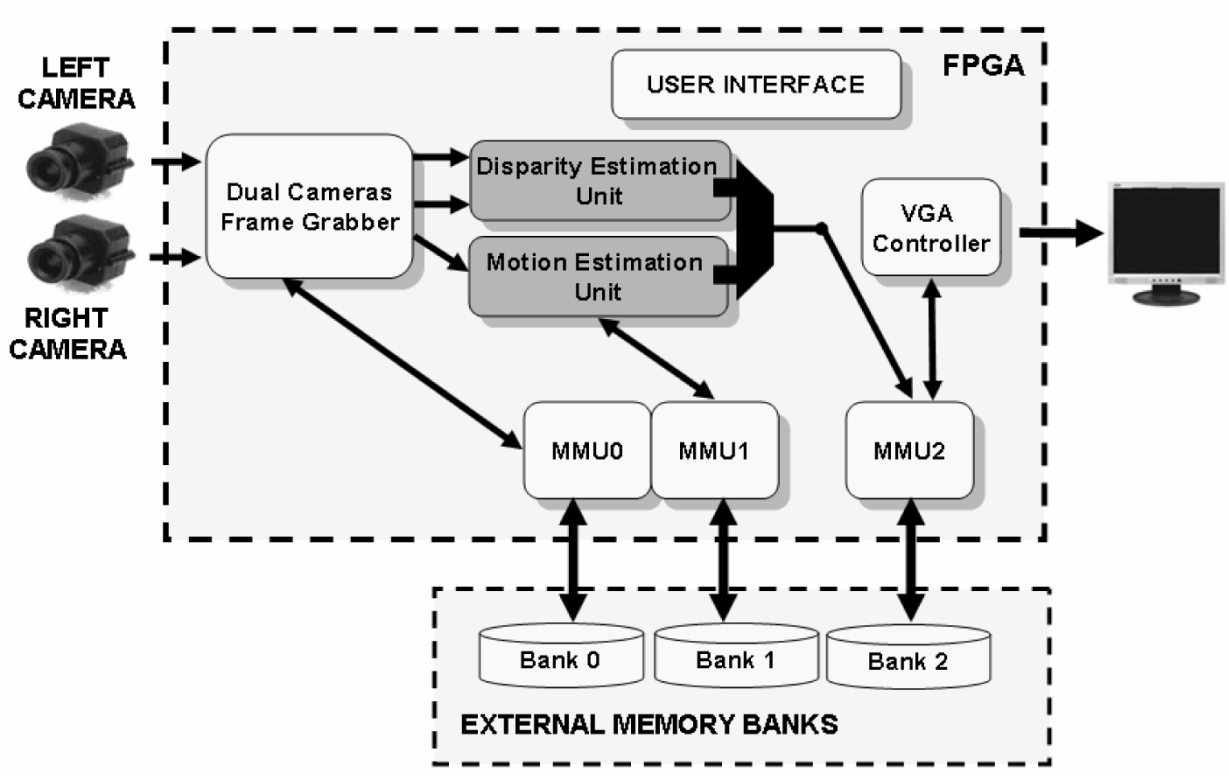


Table 1 indicates the number of fine pipeline stages as well as the number of scalar units. In total, 446 SPS are required for the image processing cores. In this table, we exclude the SPS used for interfacing and memory management. The whole architecture uses more than 500 SPS.

Table 1 Number of SPS used for the image processing cores. We indicate the number of fine pipeline stages as well as the number of scalar units used in each coarse stage S_i. Note that stage S_2 of the stereo units uses two independent datapaths, one for computing the disparity (D) and the other one for estimating the confidence threshold (C_T). This is noted as D/C_T in the scalar unit column

Processing unit (coarse pipeline stages)	*Pipeline stages*	*Scalar units*	*Simple circuits*
Motion processing unit			
S_0: Image smoothing	12	1	12
S_1: Temporal operations	9	2	18
S_2: Spatio-temporal image derivatives	12	3	36
S_3: Least squares fitting elements	12	5	60
S_4: Local velocities computation	25	2	50
Binocular stereo processing unit			
S_0: DC component removal	50	2	100
S_1: Gabor filtering	28	4	112
S_2: Direct phase computation	45/9	1/1	54
S_3: Disparity computation	4	1	4
Total number of SPS: Motion (176) + Stereo (270) = 446 SPS			

4.2 System resources and qualitative results

The whole system has been successfully synthesised into a Virtex II XC2V6000-4 Xilinx FPGA. It required less than 35% of the device, including the user interface unit, memory controller, frame-grabber of binocular cameras and VGA visualisation module. Note that the whole system only computes motion information for one of the cameras, as this is sufficient for most vision applications. The complete system resources consumption is presented in Table 2.

We need to point out that complete system of resources for stereo-motion is less than the addition of the two systems. This is clearly explained if we take into account that stereo and motion units have several common interfacing units (e.g. frame-grabbers, VGA controllers, etc.) which can be shared in the whole architecture.

It also important to point out that each image feature (stereo or motion information) has different requirements about image resolution and frame rate. Table 2 represents an example of image resolution for each case, providing larger resolution for the stereo unit to increase depth estimation accuracy and higher frame-rate to the motion module to reduce the temporal aliasing. When both modules are integrated into the same device, though image streaming can be managed completely independently, it is quite difficult (and expensive) to find cameras which allow this high frame rate and resolution. A compromise solution given in Table 1 consists in the utilisation of a medium frame-rate and image resolution camera. Nevertheless, it shall be rather specified depending on the target application.

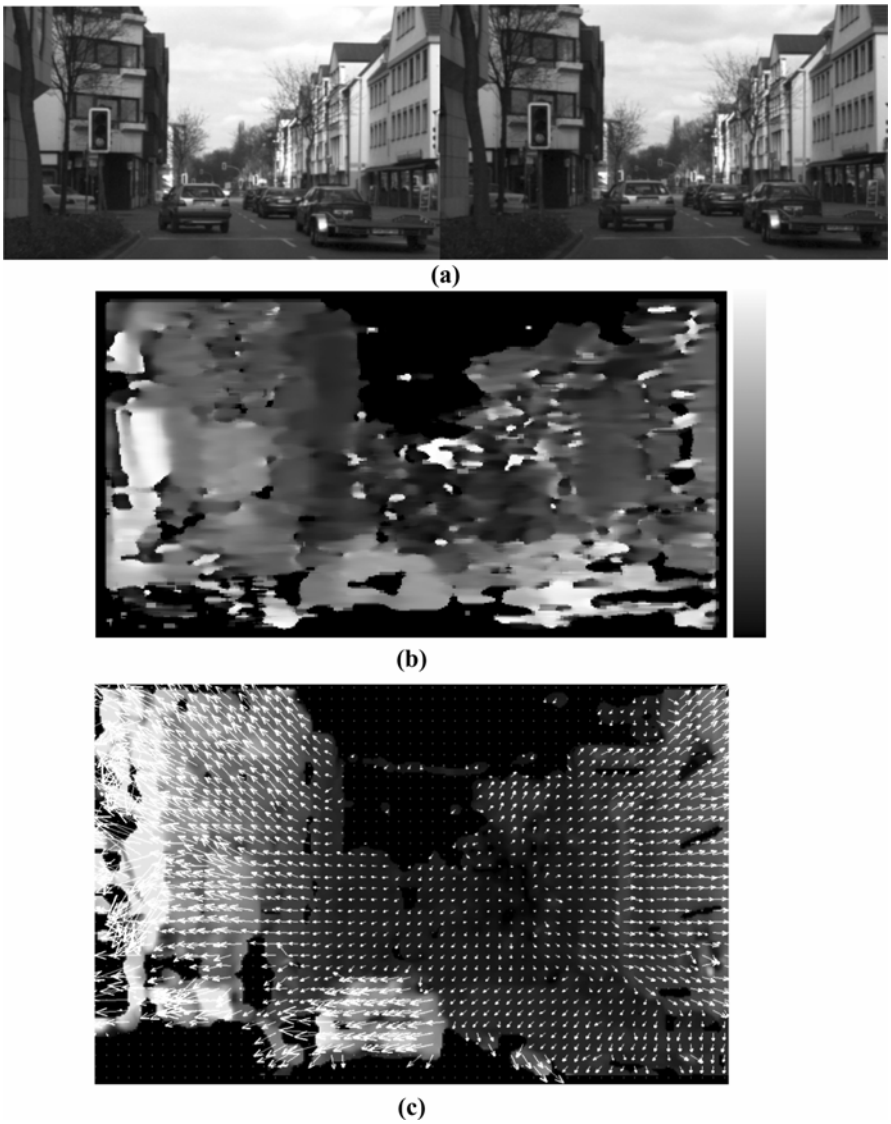
A last point to consider is related to the bandwidth requirements. For each pixel image, the disparity value and x–y velocity component uses 12 bits. This means that we produce 36 bits per pixel. If we try to send this information at the maximum speed, 46.7 MHz, this means that we require up to 1.7 Gbits/s. This is difficult though feasible with current communication buses such as PCI-express. Nevertheless, it is a huge quantity of data, which is difficult to manage for a standard processor. This motivates the necessity for integration/fusion mechanisms as commented in Section 1.

Figure 5 shows an example of the final system outputs. A binocular driving scene is acquired and processed. Stereo information can be clearly used to determine the object distances (using a proper calibration). Motion shows the expansion focus of the image which can be used to estimate heading information.

Table 2 System resources required on a Virtex II XC2V6000-4. The first row includes the resources for a monocular motion system. The second row corresponds to the disparity estimation module, and finally, the last row indicates the whole stereo-motion processing system. Each row presents the resources required to implement the image processing core, including the memory management unit, camera Frame-grabber, VGA signal output generation and user configuration interface. (*Mpps*: *mega-pixels per second* at the maximum system processing clock frequency, *EMBS*: embedded memory blocks)

Processing stage	*Logic cells/(%)*	*EMBs/(%)*	*Embedded multipliers/(%)*	*Mpps*	*Image resolution*	*Fps*
Motion	18096 (26%)	28 (19%)	12 (8%)	50.7	640 × 480	165
Stereo	13852 (20%)	15 (10%)	8 (5%)	48.35	1280 × 960	39
Stereo-motion	27454 (40%)	43 (29%)	20 (13%)	46.7	800 × 600	97

Figure 5 Optical flow and disparity computation results. (a) Input binocular images of a forward view driving scene. (b) Estimated Disparity values. Closer objects represented in light grey levels and far away objects in dark grey levels. Black codifies unconfident values which are rejected after the computation. (c) Module of the computed velocity with arrows superimposed to indicate the motion direction. Due to the image perspective, closer objects seem to move faster and are represented using light grey levels. Black areas represent unconfident values rejected after the computation.



5 Conclusion

This paper has illustrated how two different low level vision visual modalities, stereo and motion, can now be efficiently computed on a single chip. This is of high interest for a large number of applications which can benefit from this information in real-time. This system opens the way to new models and applications for integrating this information in a robust way by using the mechanisms described in the introduction for intermediate and high level vision tasks.

We have validated our architectural hypothesis of high parallelism computation as a high performance and feasible solution. The whole system requires more than 500 SPS. It runs at a 46.7 MHz rate and is able to provide one estimation (disparity and velocity) per clock cycle. The image resolution can be determined by the user and according to that, the maximum frame-rate can be estimated. This device achieves an outstanding performance compared to any previous literature contribution, which validates our design strategy.

Nevertheless, is important to point out that the output data stream requires a large bandwidth, which makes difficult to transmit from the FPGA to a standard processor. Therefore, future work will address the integration mechanisms required to properly compress the data stream into only highly reliable information. We will address this by building multimodal entities which encode and compress the relevant image stereo and motion information.

Acknowledgements

This work has been supported by the Spanish Grant (DPI2004-07032) and the EU Grant DRIVSCO (IST-016276-2).

References

Barron, J.L., Fleet, D.J. and Beauchemin, S. (1994) 'Performance of optical flow techniques', *International Journal of Computer Vision*, Vol. 12, No. 1, pp.43–77.

Brown, M.Z., Burschka, D. and Hager, G.D. (2003) 'Advances in computational stereo', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 25, No. 8, pp.993–1008.

Bruhn, A., Weickert, J., Feddern, C., Kohlberger, T. and Schnorr, C. (2005) 'Variational optical flow computation in real time', *IEEE Transactions on Image Processing*, Vol. 14, No. 5, pp.608–615.

Cozzi, A., Crespi, B., Valentinotti, F. and Wörgötter, F. (1997) 'Performance of phase-based algorithms for disparity estimation', *Machine Vision and Applications*, Vol. 9, Nos. 5–6, pp.334–340.

Curcio, C.A., Sloan, K.R., Kalina, R.E. and Hendrickson, A.E. (1990) 'Human photoreceptor topography', *Journal of Computer Neural*, Vol. 292, No. 4, pp.497–523.

Darabiha, A., MacLean, W.J. and Rose, J. (2006) 'Reconfigurable hardware implementation of a phase-correlation stereo algorithm', *Machine Vision and Applications Journal*, Vol. 17, No. 2, pp.116–132.

Díaz, J. (2006) 'Multimodal bio-inspired vision system. High performance motion and stereo processing architecture', PhD Dissertation, University of Granada (electronic copy available under request to the author).

Díaz, J., Ros, E., Carrillo, R. and Prieto, A. (2007) 'Real-time system for high-image-resolution disparity', *IEEE Transacitons on Image Processing*. Vol.16, No. 1, pp.280–285.

Díaz, J., Ros, E., Mota, S. and Rodríguez-Gomez, R. (2006a) 'Highly paralellized architecture for image motion estimation', *Lecture Notes in Computer Science*, Vol. 3985, Springer-Verlag, pp.75–86.

Díaz, J., Ros, E., Pelayo, F., Ortigosa, E.M. and Mota, S. (2006b) 'FPGA based real-time optical-flow system', *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 16, No. 2, pp.274–279.

Fleet, D.J. and Jepson, A.D. (1993) 'Stability of phase information', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 15, pp.1253–1268.

Focus Robotics, Stereo products (2006) Available at: http://www.focusrobotics.com/products/systems.html.

García-Pérez, M.A. and Peli, E. (2001) 'Intrasaccadic perception', *The Journal of Neuroscience*, Vol. 21, No. 18, pp.7313–7322.

Goodchild, A.K., Ghosh, K.K. and Martin, P.R. (1996) 'Comparison of photoreceptor spatial density and ganglion cell morphology in the retina of human, macaque monkey, cat, and the marmoset Callithrix jacchus', *Journal of Comparative Neurology*, Vol. 366, pp.55–75.

Hartstein, A. and Puzak, T.R. (2002) 'The optimum pipeline depth for a microprocessor', *Proceedings of the 29th Annual International Symposium on Computer Architecture*, pp.7–13.

Liu, H.C., Hong, T.S., Herman, M., Camus, T. and Chellappa, R. (1998) 'Accuracy vs efficiency trade-offs in optical flow algorithms', *Computer Vision and Image Understanding*, Vol. 72, No. 3, pp.271–286.

Lucas, B.D. and Kanade, T. (1984) 'An iterative image registration technique with an application to stereo vision', *Proceedings of the DARPA Image Understanding Workshop*, pp.121–130.

Martín, J.L., Zuloaga, A., Cuadrado, C., Lázaro, J. and Bidarte, U. (2005) 'Hardware implementation of optical flow constraint equation using FPGAs', *Computer Vision and Image Understanding*, Vol. 98, No. 3, pp.462–490.

Mead, C. (1989) *Analog VLSI and Neural Systems*, Boston, MA: Addison-Wesley Longman Publishing Co., Inc.

Mota, S., Ros, E., Díaz, J., Ortigosa, E.M. and Prieto, A. (2005) 'Motion-driven segmentation by competitive neural processing', *Neural Processing Letters*, Vol. 22, No. 2, pp.125–147.

Rares, A., Reinders, M.J.T. and Hendriks, E.A. (1999) 'Image interpretation systems', *MCCWS Project Technical Report (MCCWS 2.1.1.3.C)*.

Ratha, N.K. and Jain, A.K. (1999) 'Computer vision algorithms on reconfigurable logic arrays', *IEEE Transactions on Parallel and Distributed Systems*, Vol. 10, No. 1, pp.29–43.

Simoncelli, E.P. (1993) 'Distributed analysis and representation of visual motion', PhD Thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, Cambridge, MA.

Solari, F., Sabatini, S.P. and Bisio, G.M. (2001) 'Fast technique for phase-based disparity estimation with no explicit calculation of phase', *Electronics Letters*, Vol. 37, No. 23, pp.1382–1383.

Watson, A., Ahumada, A. and Farrell, J. (1986) 'Window of visibility: a psychophysical theory of fidelity in time-sampled visual motion displays', *Journal of the Optical Society of America A: Optics, Image Science, and Vision*, Vol. 3, No. 3, pp.300–307.

Weems, C.C. (1991) 'Architectural requirements of image understanding with respect to parallel processing', *Proceedings of the IEEE*, Vol. 79, No. 4, pp.537–547.