RECONFIGURABLE ARCHITECTURE TO ESTIMATE THE MOTION OF AN UNDERWATER VEHICLE

V. Ila, R. Garcia, J. Batlle

Institute of Informatics, Underwater Vision Lab
C/Lluis Santalo S/N, 17071, Girona, Spain
{*viorela,rafa,jbatlle*}*@eia.udg.es*

Abstract*.This work proposes a parallel architecture for a motion estimation algorithm. It is well known that image processing requires a huge amount of computation, mainly at low level processing where the algorithms are dealing with a great numbers of data-pixel. One of the solutions to estimate motions involves detection of the correspondences between two images. Due to its regular processing scheme, parallel implementation of correspondence problem can be an adequate approach to reduce the computation time. This work introduces parallel and real-time implementation of such low-level tasks to be carried out from the moment that the current image is acquired by the camera until the pairs of point-matchings are detected*

Keywords*: Image matching, Processor arrays systems, Real-time systems, Hardware*

1. INTRODUCTION

Presently, different methods for motion estimation of an underwater vehicle exist, mainly based on acoustic sensor networks. This strategy is relatively expensive since transponders have to be deployed from a ship, calibrated and recovered after the mission. Therefore, this procedure is not adequate for low cost underwater vehicles. One cost-effective alternative can be to equip the vehicle with a down-looking camera, which acquires seafloor images while the robot is performing its mission. This down-looking camera provides rich visual information which can be used for vehicle motion estimation. Our work is focusing on low-level image processing tasks associated to the motion detection algorithm where a large amount of data has to be processed. This paper explores the possibility of hardware implementation of some tasks like interest points detection and matching procedure.

Correlation algorithms have important properties like regularity and modularity. Thus, they can be decomposed in computational blocks that can be processed in parallel. An extensive literature exists about array architectures applied to image processing, especially in Block Matching Algorithms (BMA) for motion estimation [2,8,10]. Komarek et al. [8] described specific solutions for array architectures of full search BMA. They propose four different alternatives for one and two dimensional array architectures. In the early nineties, different other VLSI designs were proposed for decreasing BMA computation time [1,2]. While in full search BMA the image is divided in blocks and the algorithm looks for matches of every block in a frame, our approach is looking for correspondences of interest points. These are scene features which can be reliably found when the camera moves from one location to another and lighting conditions of the scene change. On the other hand, a more complex

error measurement criterion like normalised correlation [13] is applied.

The remainder of this paper is structured as follows. Section 2 proposes a hardware implementation for motion estimation algorithm. Section 3 defines some specification of the FPGA-based hardware for motion estimation of an underwater robot. Finally, section 4 outlines the conclusions and future work.

2. HARDWARE IMPLEMENTATION OF THE MOTION ESTIMATION ALGORITHM

The goal of the algorithm is to estimate the motion of an underwater robot. Point correspondences between the current image acquired by the camera and a previous reference image have to be found in order to compute a motion estimation matrix. Often this means detecting features in one image, and matching them in the other one. The selection of features may depend on the application, although points are commonly used because they can be easily extracted and are quite robust to noise. Underwater images are difficult to process due to the medium transmission properties and non-uniform illumination [6]. These aspects can provoke undesired bad correspondences (*outliers*) that can introduce errors in the motion estimation process. For this reason normalised correlation is very adequate to reduce the influence of non-uniform illumination [4,13].

2.1. Real time feature detection

The first step in solving the correspondence problem is the detection of a set of well-contrasted points in the current image. Corner detector algorithms consist of computing the image gradient components: I_x and I_y by convolving the current image with the Prewitt masks. Benedetti et al. [2,3] proposed a modified Tomasi-Kanade [11] algorithm which reduce the computation and avoids floating-point. In this algorithm a G matrix is considered.

$$G = \begin{pmatrix} \sum_{k=1}^{N} (I_x^k)^2 & \sum_{k=1}^{N} (I_x^k I_y^k)^2 \\ \sum_{k=1}^{N} (I_x^k I_y^k)^2 & \sum_{k=1}^{N} (I_y^k)^2 \end{pmatrix} = \begin{pmatrix} a & b \\ b & c \end{pmatrix} \quad (1)$$

The algorithm, first calculates $a(i,j)$, $b(i,j)$ and $c(i,j)$; then

$$P_{\lambda_t}(i,j) = (a-\lambda_t)(c-\lambda_t) - b^2 \quad (2)$$

is found. Every pixel having:

$$P_{\lambda_t}(i,j) > 0 \quad and \quad a(i,j) > \lambda_t \quad (3)$$

is retained, where λ_t is the imposed lower bound for the solutions of the equation (2).

The last step of the algorithm discards any pixel that is not a local maximum of $P_{\lambda_t}(i,j)$. N interest points are selected, considering the highest values for $P_{\lambda_t}(i,j)$. In this approach the complexity is considerably reduced and does not require any floating point operation. As is showed above, the first step in corner detection is the computation of the image gradient components I_x and I_y by convolving the current image with a set of 3×3 Prewitt masks. Benedetti et al. [3] proposed an implementation based on two FIFOs and two buffers used to delay the incoming pixel. The Data Flow Graph (DFG) for the image convolution with the Prewitt masks and summing elements inside a 3×3 window are shown on the right side of Figure2. The delay introduced by the corner detector is important, since we are interested in the memory address of the N corners instead of their value of cornerness. Every 3×3 window generator introduces a latency of two lines and two pixels. The delay introduced by the computation of P_{λ_t} is showed in equation 4 and depends on the image size $(M_i \times N_i)$, pixel sampling time (st) and the time spent for one computational blocks from figure 1: *Prewitt* (t_P), *Sum* (t_S) and *Compute* P_{λ_t} (t_C).

$$T_{P_{\lambda_t}} = 2\cdot(2\cdot M_i + 1)\cdot st + t_P + t_S + t_C \quad (4)$$

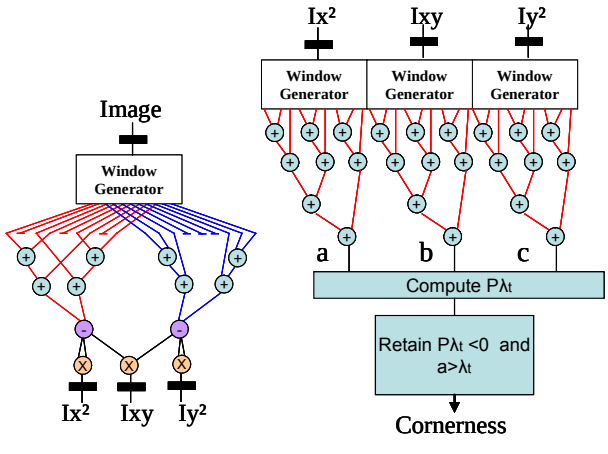

Figure 1. Corner detector DFG. a) Prewitt mask. b) Cornerness computation.

In order to retain N pixels with highest value of P_{λ_t}, a pipeline of N interconnected Sort-Processing Elements (SPE) is proposed. One SPE compares the input value with the one stored in its buffer and retains the highest cornerness value.

2.2. Parallel implementation of correspondence problem

Once interest points are detected in the current image I_c, we search for correspondences in the reference image I_r. A correlation algorithm provides, for each interest point (x_c, y_c) of the current image, its corresponding match (x_r, y_r) in the reference image. The correlation score is defined as the covariance between the grey levels of a region defined by the *correlation window* in the current image and the same region defined in the reference image. The algorithm searches for all similar patches inside the correspondent *search window*. A normalised correlation criteria C, which assures that the result is not altered in presence of nonuniform illumination [5], is showed in equation (5).

$$C = \frac{\sum_{-\alpha}^{\alpha}\sum_{-\alpha}^{\alpha}(I_c(x_c+i,y_c+j)-\overline{I_c(x_c,y_c)})(I_r(x_r+i,y_r+j)-\overline{I_r(x_r,y_r)})}{(2\alpha+1)^2\sqrt{\sigma^2(I_c)\cdot\sigma^2(I_r)}} \tag{5}$$

where $(2\alpha+1)\times(2\alpha+1)$ is the size of the correlation window. $\overline{I_c(x_c,y_c)}$ and $\overline{I_r(x_r,y_r)}$ are the average intensity and $\sigma^2(\cdot)$ defines the variance of both correlation windows. The correlation algorithm compares the correlation score of each pixel within the search window and selects the highest one.

We propose a breaking down of criteria C for its parallelization. We can observe that there are five sums to be computed in equation (5): sum_1, sum_2, sum_3, sum_4 and sum_5.

$$\begin{aligned}
sum_1 &= \sum_{i=-\alpha}^{\alpha}\sum_{j=-\alpha}^{\alpha} I_c(x_c+i,y_c+j)\\
sum_2 &= \sum_{i=-\alpha}^{\alpha}\sum_{j=-\alpha}^{\alpha} I_c(x_c+i,y_c+j)^2\\
sum_3 &= \sum_{i=-\alpha}^{\alpha}\sum_{j=-\alpha}^{\alpha} I_c(x_c+i,y_c+j)\cdot I_r(x_r+i,y_r+j)\\
sum_4 &= \sum_{i=-\alpha}^{\alpha}\sum_{j=-\alpha}^{\alpha} I_r(x_r+i,y_r+j)^2\\
sum_5 &= \sum_{i=-\alpha}^{\alpha}\sum_{j=-\alpha}^{\alpha} I_r(x_r+i,y_r+j)
\end{aligned} \tag{6}$$

Then, equation (5) becomes:

$$C = \frac{sum_3 - \frac{1}{(2\alpha+1)^2}\cdot sum_1\cdot sum_5}{\frac{1}{(2\alpha+1)^2}\cdot\sqrt{[(2\alpha+1)^2\cdot sum_2 - sum_1^2]\cdot[(2\alpha+1)^2\cdot sum_4 - sum_5^2]}} \tag{7}$$

This breaking down leads to an easy parallel implementation, while each Processing Element (PE) of the architecture executes in parallel the computation of these five sums. Furthermore, the Post Processing Element (PPE) performs the remaining computation.

When mapping an algorithm into an array of processors, the problem is to access multiple data to feed all the PEs at the same time. Yang et al. [10,12] proposed a solution that consists of local data exchange between PEs. Adopting this

approach to our design, two parallel memory accesses (r_1, r_2) are used for the same reference image and one for the current image (c), see Figure 3. Once read from memory, the data are broadcasted to every PE. Buffers are used to delay data and multiplexers to switch between data. For higher utilisation efficiency of the architecture, the size of the search window must be defined according to the size of the correlation window by the equation $p = 2a$. The number of PEs also depends on the size of the correlation window and is equal to $(2\alpha + 1)$. One PE is in charge of the parallel computation of the five sums defined in equation (6). Two accumulations (Acc) and three multiplication-accumulations (M_Acc) are executed in parallel.

The results from the array of PEs are pipelined into the Post Processing Element (PPE) (see figure 2). The PPE computes the correlation criteria defined in the equation (7). It can be observed that, seven multiplications, three substraction, one square root and one division have to be implemented in hardware. Parallel implementation of these operations is performed [7]. The square root implementation is based on the non-restoring algorithm proposed by Li [9]. The advantage of this method is the reduced space occupied on the FPGA device and generates an exact result value. The last step of the algorithm compares all the error measurements corresponding to every candidate match. The result of the algorithm is the coordinates of the pixel with the biggest value for the correlation score. For N given interest points, the delay introduced by the parallel implementation of the correlation criteria from equation (7) is given by:

$$T_C = [2 \cdot [(2\alpha+1)^2 \cdot [(2p+1) - 2\alpha] + 2\alpha] + t_{PPE}] \cdot N \quad (8)$$

where t_{PPE} is the time delay introduces by PPE.

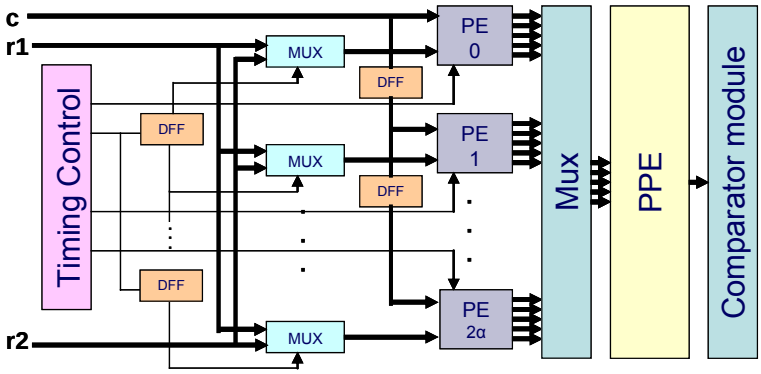

Figure 2. Array of Processing Elements and Post Processing Element.

3. VISION BASED SYSTEM

Based on the hardware implementation of the motion estimation algorithm presented above we propose a vision system based on an FPGA device for the PC/104+ standard. This system is in charge of the acquisition and processing of the image, and communication with the control system running on a PC/104+ computer. Nevertheless, FPGA devices offer the possibility of parallel implementation of image processing tasks such as: corner detection, solving the correspondence problem, etc. Thus, frame-rate performance can be achieved. Our previous experiments showed that the execution of a matching algorithm from sub-section 1 can run 50 times faster in an FPGA-based architecture than in a Pentium based PC/104+ computer.

The heart of the proposed system is an Altera Stratix EP1S25F672 FPGA device. A balance between price and performance was considered when choosing the device. Parallel processing of computer vision tasks like the one mentioned above requires multiple memory accesses. Thus, the smart frame grabber is provided with four external memories like in the figure 3. One solution to obtain real time is to store the incoming image in a memory while the previous one is being read and processed from another memory. The FPGA internal memory is reserved for intermediary storage (FIFOs and buffers). Computation associated with computer vision tasks requires a large area on the FPGA device. A commercial PCI controller chip is used for FPGA interfacing with the control system, so that the FPGA designs need only to

interface to a simple synchronous local bus. Figure 3 shows a block schema for the FPGA hardware architecture.

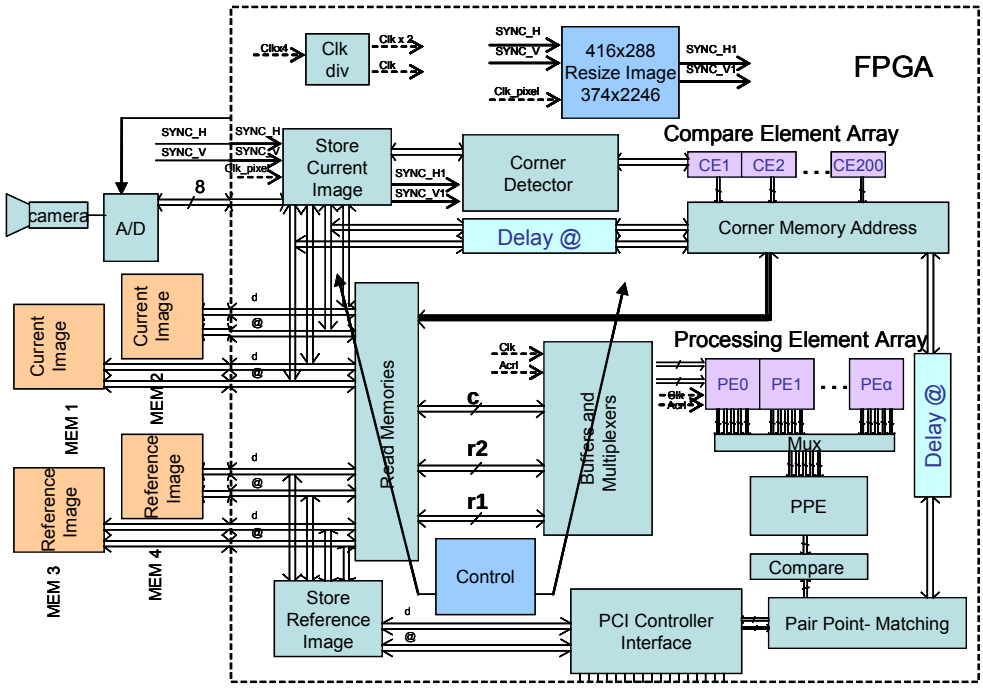

Figure 3. FPGA hardware architecture.

4.CONCLUSIONS

This paper has described a parallel implementation of the correspondence problem for a motion estimation algorithm in order to obtain frame-rate performance. This implementation defines the constraints for the development of a powerful vision system based on reconfigurable devices with the goal to be integrated in the architecture of an underwater robot. The only size and communication restrictions of this system are set up by the PC/104+ standard. In this way, this board can be integrated in any similar system. Due to its reconfigurable characteristics, the proposed system can be used to solve correspondence problem in situations other than motion estimation. For instance, when an autonomous robot carries a stereo vision system, two boards connected together may detect correspondences between left and right images in real time. Indeed, commercial alternatives to this system exist in the market: frame-grabbers, FPGA boards for PC/104+, etc. But, input/output throughput, memory and computation requirements of the computer vision algorithm applied to robotics overcome the possibilities of these commercial systems. Our system based on reconfigurable devices technology, permits real time performance, complex design integration, high parallelism and flexibility.

5. REFERENCES

[1] P. Baglietto, M. Maresca, A. Migliaro, and M. Migliardi. Parallel implementation of the full search block matching algorithm for motion estimation. In *Proceedings of the International Conference on Application Specific Array Processors*, pages 182 –192, 24-26 July 1995.

[2] A. Benedetti and P. Perona. Real-time 2-D feature detection on a reconfigurable computer. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 586 –593, 23-25 June 1998.

[3] A. Benedetti, A. Prati, and N. Scarabottolo. Image convolution on FPGAs: the implementation of a multi-FPGA FIFO

structure. In *Proceedings on Euromicro Conference 1998.*, pages 123 –130 vol.1, Aug. 1998.

[4] X. Cufí, R. Garcia, and R. Ridao. An approach to vision-based station keeping for an unmanned underwater vehicle. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 799–804, Lausanne, 2002.

[5] R. Garcia, X. Cuf´ı, and V. Ila. Recovering camera motion in a sequence of underwater images through mosaicking. In *First Iberian Conference on Pattern Recognition and Image Analysis, Lecture Notes in Computer Science , no. 2652*, pages 255–262, 2003.

[6] R. Garcia, T. Nicosevici, and X. Cufí. On the way to solve lighting problems in underwater imaging. In *IEEE OCEANS Conference (OCEANS)*, pages 1018–1024, Mississipi, 2002.

[7] V. Ila, R. Garcia, and F. Charot. Proposal of a parallel architecture for a motion detection algorithm. In *International Conference on Pattern Recognition*, Cambridge, Aug. 2004.

[8] T. Komarek and P. Pirsch. Array architectures for block matching algorithms. *IEEE Transactions on Circuits and Systems*, 36:1301 –1308, 10 , Oct 1989.

[9] W. Li and W. Chu. A new non-restoring square root algorithm and its VLSI implementations. In *1996 IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pages 538 – 544, 7-9 Oct. 1996.

[10] M.-T. Sun and K.-M. Yang. A flexible VLSI architecture for full-search block-matching motion-vector estimation. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, pages 179 –182 vol.1, 8-11 May 1989.

[11] C. Tomasi and T. Kanade. Detection and tracking of point features. Cmu-cs-91-123, Carnegie Mellon University, Apr. 1991.

[12] K.-M. Yang, M.-T. Sun, and L. Wu. A family of VLSI designs for the motion compensation block-matching algorithm. *IEEE Transactions on Circuits and Systems*, pages 1317 –1325, Oct. 1989.

[13] Z. Zhang, R. Deriche, O. D. Faugeras, and Q.-T. Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. *Artificial Intelligence*, 78(1-2):87–119, 1995.