VLSI Architecture for an Underwater Robot Vision System

Viorela Ila
and Rafael Garcia
Institute of Informatics and Applications
Campus Montilivi, 17071 Girona, Spain
Email: {viorela,rafa}@eia.udg.es

Francois Charot
IRISA
Campus de Beaulieu, 35042 Rennes, France
Email: francois.charot@irisa.fr

***Abstract*—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 number of data-pixel. One of the solutions to estimate motions involves detection of the correspondences between two images. For normalised correlation criteria, previous experiments shown that the result is not altered in presence of nonuniform illumination. Usually, hardware for motion estimation has been limited to simple correlation criteria. The main goal of this paper is to propose a VLSI architecture for motion estimation using a matching criteria more complex than Sum of Absolute Differences (SAD) criteria. Today hardware devices provide many facilities for the integration of more and more complex designs as well as the possibility to easily communicate with general purpose processors.**

I. Introduction

Presently, different methods for motion estimation and localisation 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, small-size underwater vehicles such as URIS (Underwater Robotic Intelligent System) developed at the University of Girona (see figure 1). 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. Sequence of images acquired by the camera mounted on the robot can be used in constructing a map of the zone surveyed by the submersible. This map is a composite image constructed by aligning a set of smaller consecutive images. An example of a part of a underwater mosaic image is presented in figure 2.

In most of the cases the process involves recovering the motion of the vehicle by means of gray level correlation [1] or using optical flow [2]. Unfortunately underwater images are difficult to process due to the medium of transmission characteristics. Blurriness of elements of the image, cluttering and non-uniform illumination are some of the problems present in underwater imaging.

Our aim in the present work is, starting from analysis of VLSI architectures for motion estimation, to develop efficient versions of Field Programable Gate Array (FPGA) hardware implementation of motion estimation algorithm for the particular case of underwater images. In order to enable hardware implementation, a number of transformation in the algorithms may be performed. The algorithm and the required transformations are introduced in section II. Section III describes our proposed architectures for real time motion estimation for of underwater images. The implementation would allow motion to be estimated at video rate as described in section IV. The paper ends up with the conclusions and further work.

II. Real Time Motion Estimation

When an autonomous robot performs a real mission, it needs to perceive its environment in real time. In this application the time constrains are defined by the frame rate of a low-cost PAL

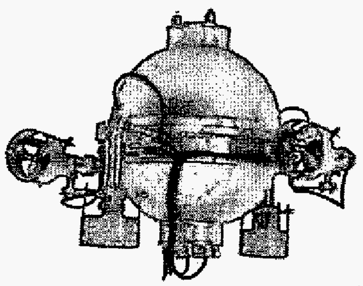

Fig. 1. URIS Underwater vehicle, developed at the University of Girona.

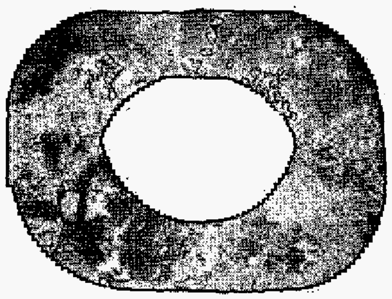

Fig. 2. Sample underwater mosaic image.

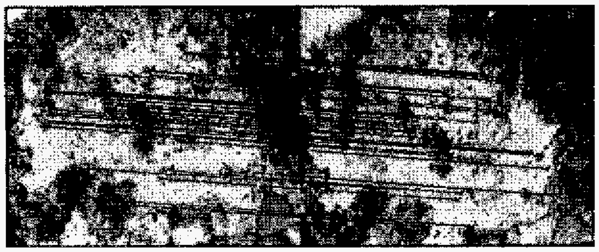

Fig. 3. Correspondences between current and reference frame detected using normalised correlation criteria.

video camera mounted on the vehicle. Frame rate of 25 frames per second is equivalent to 0.04 seconds, the time between first pixel and the last pixel of the frame are acquired. Nowadays, different solutions exist for solving real time problems. Software solutions running under Real Time Operating Systems (RTOS) provide quite good results when time constraints are critical. When time requirements overcome the capabilities of RTOS, a possible solution is to customise the implementation using Application Specific Integrated Circuits (ASIC). These devices should be considered only when the applications are specific and have a simple implementation. Indeed, alternative solutions like microprocessors have been the dominant devices in use for general-purpose computing for the last decade. In the case of image processing tasks we have to deal with complex algorithms applied to a large amount of data. Reconfigurable devices take the best of two worlds, combining the flexibility of software with the execution speed of hardware. Thus, hardware architectures based on these devices can be a good trade off for achieving real time in robotic applications. The evolution of reconfigurable devices technology in the last few years permits complex design integration, high parallelism, hardware-software co-design.

A. Matching Criteria

The motion estimation algorithm consider two images: the current image I_c acquired by the camera and a reference image I_r coming from the control system. Point correspondences between the current image and a previous reference image have to be found in order to compute a motion estimation matrix. Often this requires 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 [3]. A correlation algorithm provides, for each interest point (x_c, y_c) of the current image, its corresponding matching (x_r, y_r) in the reference image (see figure 3). 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, is showed in equation (1). In our previous work we successfully applied this criteria to underwater images, in the presence of nonuniform illumination [4]. The correlation score between a point (x_c, y_c) in the first image I_c, and point (x_r, y_r) in the second image is defined as:

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

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.

Correlation algorithms have important properties like regularity and modularity. Thus, they can be break down into computational blocks which can be processed in parallel. We propose a decomposition of criteria C for its parallelization. We can observe that there are five sums to be computed in equation (1): sum_1, sum_2, sum_3, sum_4 and sum_5.

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

Then, equation (1) becomes:

$$C = \frac{sum_3 - \frac{1}{(2\alpha+1)^2}\cdot sum_1\cdot sum_5}{\frac{1}{(2\alpha+1)^2}\cdot\sqrt{[(2\alpha+1)^2\cdot sum_2 - sum_1^2]\cdot[(2\alpha+1)^2\cdot sum_4 - sum_5^2]}} \tag{3}$$

This braking down of the equation 1 leads to an easy parallel implementation, while each Processing Element (PE) of the architecture executes in parallel the computation of these five sums. Furthermóre, the Post Processing Element (PPE) performs the remaining computation.

B. Linear Arrays for Motion Estimation

Our approach is based on some ideas applied in VLSI architectures for motion compensation algorithms used in video coding standards. Full Search Block Matching Algorithms (FSBMA) are used for motion estimation in such applications [5]–[7]. In these algorithms the current frame is divided into blocks of $n \times n$ pixels, and for every block the algorithm searches for similar blocks in the previous frame within a search area of size $(2p+n) \times (2p+n)$. In full search the algorithm has a very regular data-flow for the search area [8]. These data are repeatedly used in the computation for different candidates and can simplify the architecture.

Komarek and Pirsch [9] analysed four different systolic arrays for FSBMA mapping. Linear or quadratic array structures

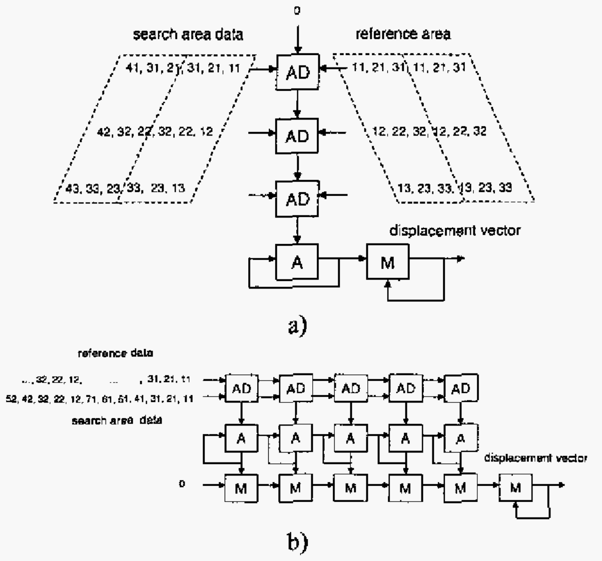
Fig. 4. Linear structures: a) AB1-type; b) AS1-type.

can be used for the implementation of the algorithm. Moreover, two possible ways of computation come out from the property of associativity of the operations in the algorithm. The four resulting array structures are denominated: $AB1$, $AS1$ for linear arrays and $AB2$, $AS2$ for quadratic arrays. These architectures exploit concurrency using different structures and numbers of PEs.

Complexity of the motion estimation algorithm from equation 3 introduces some restriction in chousing the adequate array architecture. For instance, a quadratic array is suitable only in case of simple PE architecture. When more computation must be done in parallel it can "eat" a lot of resources, which sometimes are not available. This is the reason why we restrict our analysis to linear arrays: AB1-type and AS1-type are shown in figure 4.

One solution to reduce the latency introduced by both AB1 and AS1 structures is to increase the memory access. When every PE is supplied with data coming from external memory, idle cycles can be avoided. When accessing external memory is constrained, reducing the latency can also be obtain by controlling the data-flow through the array using registers and multiplexors. This strategy was applied by Vos [7], Roma [6] and Hsieh [10] in order to make the AB2 quadratic architecture more efficient and by Yang [11] for linear arrays. When different reference and search data flow into each PE, and the intermediate results are broadcast through the array, this latency can be reduced as it is point out in table I.

Our approach estimates the motion of certain point from the image, this reducing the number of operations but increases the complexity of memory addressing. Due to the complexity of the chosen motion estimation algorithm, output bandwidth is also restricted, while a post processing calculus must be performed. The goal is to design a motion estimation architecture to reduce the input/output bandwidth maintaining the hardware efficiency and reduced execution time. Reducing the memory throughput can be solved by adding many registers for the search area and also for reference block. Vos [7] proposed one solution to reduce space occupied into the device by replacing the registers with memory blocks which reduces half of the silicon occupancy. Taking into consideration the new FPGA technology, available embedded memory blocks can save a large amount of logic elements. A part of low memory bandwidth, the solution proposed by Yang [11] also reduces the latency, being the most efficient between the proposed linear arrays. But as any other solution, it also has some small inconveniences. One can be that, for high hardware utilisation the restriction $(2p+1)=n$ must be accomplished.

III. VLSI ARCHITECTURE FOR NORMALISED CORRELATION

While in FSBMA the image is divided in blocks and the algorithm is looking for matches of every block in a search area, our approach is looking for correspondences of areas surrounding interest points. These are scene features which can be reliably found when the camera moves from one location to another, even when lighting conditions of the scene change. In our previous work we proposed a real-time implementation of interest points detection [12]. Due to its simplicity Sum of Absolute Differences (SAD) has been most extensively used matching criteria in VLSI implementation. In case of underwater imaging, our previous works [4] showed that by applying normalised correlation criteria to find matching in pairs of images, the result is invariant to nonuniform illumination. The complex error measurement computation is shared out over two computational elements: an array of Processing Elements (PE), each PE performing two accumulations and three multiply-accumulations and a Post Processing Element (PPE) containing multipliers, subtractors, square root and division to compute the error measurement.

VLSI architectures for motion estimation presented above can be easily adapted to our algorithm. The complexity of the processing element determined us to set apart the quadratic arrays, so that only linear arrays are analysed in this proposal. Figure 5 represents the AB1 and AS1 structures adapted to our design. The architectures correspond to experimental correlation window size of 3×3 and search window size 7×7.

As we can see in figure 5 b), a reduced AS1 structure is analysed, where each PE has a search window datum input from the memory and the correlation window data is broadcast through the array. Comparing with AB1, this strategy reduces the number of time cycles but increases the number of processing elements.

AB1-type architecture is suitable for applications where large motion vectors must be estimated, which imply big search windows. When the application requires faster processing speed, AS1-type architecture can achieve higher performance than AB1-type for small search window size. The approach introduced by Yang et al. [11] is an important contribution to reduce the memory throughput comparing with both AB1 and AS1. Figure 6 shows this strategy applied to

TABLE I

LINEAR ARRAYS: PERFORMANCE AND RESOURCES.

Architecture	Time Cycles	Processing Elements	Memory Addressing
AB1	$n \times (2p+1) \times (2p+n)$	n	2
AB1_reduced	$n \times (2p+1)^2$	n	$n+1$
AS1	$n \times (2p+1) \times (2p+n)$	$2p+1$	2
AS_reduced	$n^2 \times (2p+1)$	$2p+1$	$(2p+1)+1$
Yang arch.	$n \times (2p+1)^2$	n	2

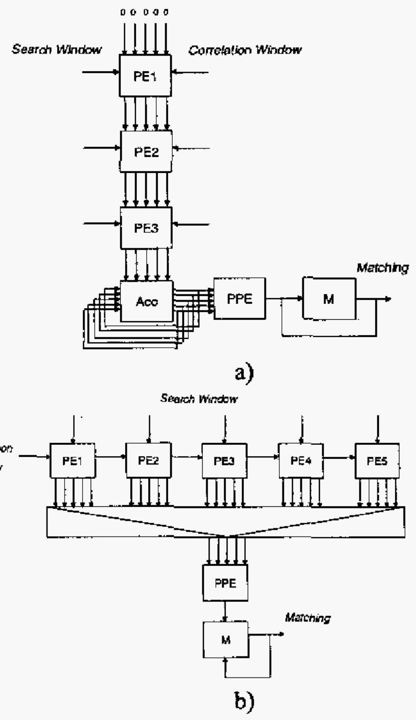

Fig. 5. Proposal of linear architectures to implement normalised correlation algorithm. a) AB1-type; b) AS1-type.

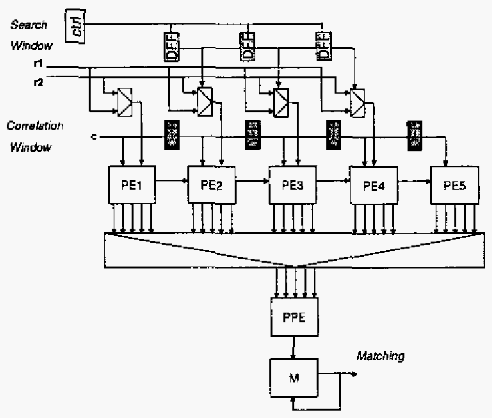

Fig. 6. Yang VLSI architecture to implement normalised correlation algorithm.

our algorithm. Another advantage of this architecture is that there is only one delay cycle between the results corresponding to each candidate in a line, while in AB1 structure there are $(n-1)$ delay cycles depending on the correlation window size. The drawback of this architecture is that it processes one column less that the others, so that "dummy" data are input to the array.

Despite of this, Yang's proposal is an efficient architecture, due to its low memory throughput, small delay introduced and the fact that the result of PEs array can be easily pipelined into PPE without extra control requirements.

1) Processing Element: A processing element may execute two accumulations and three multiply-accumulations in parallel. Figure 7 presents the internal structure of one PE in both cases: AB1-type and AS1-type (Yang's architecture). In AB1 structure the PE performs three multiplications and five additions as the accumulation is done in a separate block at the end of the PEs array. Besides, in AS1 structure the accumulations are done in the PE. It increases the size of the PE but simplify the control, while each PE has the same structure. In case of AB1, one PE applies the arithmetic operators to the outputs of the previous PE, therefore the bit-size of the inputs and the outputs of each PE vary through the array of processing element.

It is obvious that the AB1-type PE may occupy less silicon area than AS1-type PE, where multiply-accumulations must be performed in parallel. This analysis helps the designer to choose between a solution to saves hardware and a second choice to reduce the memory access and increase the execution speed. As we mentioned above, depending on the application, the designer may decide for *AB1-type*, which can deal with large search areas but introduces great latencies or *AS1-type* when performance is a critical issue. In case of underwater images, the motion of the vehicle is slow, so that the displacement between consecutive frames is quite small. Therefore the architecture proposed by Yang can accelerate the algorithm to reach real time performances with fair resources requirements.

2) Post Processing Element: The Post Processing Element (PPE) is one of the critical part of our design. The results from the array of PEs are pipelined into the PPE. The PPE computes the correlation criteria defined in the equation (3). Seven multiplications, three substraction, one 64-bit square root and one 32-bit division have to be implemented in hardware. Parallel implementation of this operations is performed. The square root implementation is based on the non-restoring algorithm proposed by Li [13]. The advantage of this method is the

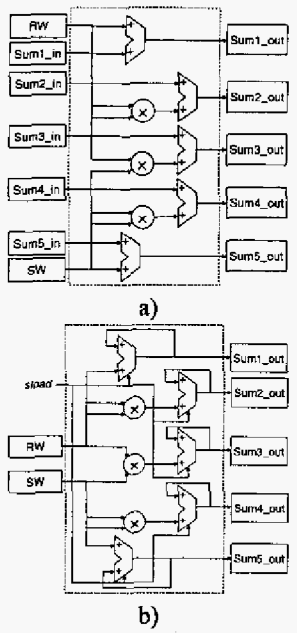

Fig. 7. Processing element internal structure. a) AB1-type; b) AS1-type (Yang).

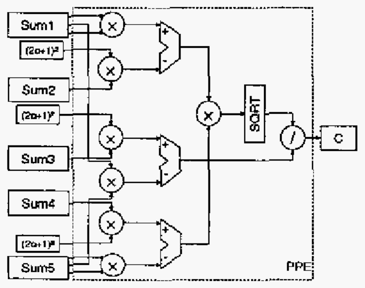

Fig. 8. Post Processing Element.

reduced space occupied on the FPGA device and generates an exact result value. Figure 8 shows the computations performed by PPE.

The last step of the algorithm compares all the 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.

IV. Implementation and Analysis

The purpose of this work is the implementation of the proposed motion estimation algorithm on a target FPGA hardware. This was accomplished by describing the algorithm in VHDL language and then synthesising it for the FPGA device. Prior to any hardware design we chose to implement a MATLAB software version corresponding to every step of the algorithm. MATLAB is a tool which facilitates procedural routines to operate on images represented as a matrix. The software implementation of the algorithms has two important roles: to chose the most adequate algorithm for our application and to provide benchmark results for hardware implementation.

The application was targeted for FPGA devices for many reasons. FPGAs are new trends for signal processing applications, where more original work can be done in terms of performance optimisation. On the other hand, today FPGA devices offer very attractive hardware facilities: great I/O pin-count, embedded memory blocks, large logic area, high possible clock speed and soft and hard embedded processors. Advanced software CAD tools are available for assisting every design stage.

VHDL language was chosen for hardware design, foremost because of familiarity and also its wide supporting range. Parameterisable VHDL blocks were implemented making possible the reuse of the design for different FPGA platforms. ModelSim simulation tool was used for design verification together with Altera's Quartus II design software. Altera's Statix family was chosen as hardware target, mostly because its accessibility and low cost. Important characteristics such as embedded multipliers and memory blocks have also been an influential factor. We benefit from the Nios Development Kit Stratix 10k edition and MJL Nios Development Kit Stratix 25k hardware platforms for algorithm's synthesis and test.

The implementation must be flexible, which means that by changing some of the parameters of the algorithm, the new generated hardware must be valid. The architecture was design in such a way to permit changing of these parameters. Computation complexity affects the level of parallelism, while many multiplications and accumulations must be performed at the same time. When talking about *flexibility* we can refer either to the architecture or to the implementation. The architecture must be able to support variation of the algorithm's parameters such as number of corners, correlation and search window size. Indeed, it imply parametrisation of the implementation. As FPGA implementation allows optimisation at bit level this parameters determine the bit size of the accumulator results, which furthermore affects the computational requirements in PPE. Table II shows the hardware requirements and the performance in case of Yang's architecture applied to different correlation and search window sizes. The required resources are quantified using Logic Elements(LE) and DSP blocks. LE are basic logic blocks of the selected FPGA device architecture and DSP blocks are embedded multiplies from the FPGA device. The performance is defined in ms and represents the latency introduced by the normalised correlation algorithm for to the detection of 100 matchings using a reasonable clock-frequency of 10MH.

Chousing an adequate description language allows migration of the design to different hardware platforms. Moreover, *complexity* reduction means avoiding floating-point units which are very area expensive. Both, corner detection and matching algorithm make use of division operation. Transformation of these algorithms must be performed such that the computation to be based only on fixed-point arithmetic. Silicon must be optimally used to implement the computation so that the data storage and the control parts must be minimised [14].

TABLE II

HARDWARE REQUIREMENTS AND PERFORMANCE CORRESPONDING TO VLSI IMPLEMENTATION OF NORMALISED CORRELATION.

Correlation window	Search window	Logic Elements	DSP blocks	Performance (ms)
7×7	13×13	2946	35	0.3534
9×9	17×17	3259	41	0.7186
11×11	21×21	3470	47	1.3458
13×13	25×25	3688	50	2.2213
15×15	29×29	3933	56	3.8438

Concerning the data storage, embedded block memories can be used for data storage for structures such as FIFOs or RAM memory blocks, so that the on-chip storage must be restricted to the size of this blocks. Frequency used for interfacing the external memory where the whole image must be stored, have impact on the total execution time of the algorithm. Therefore, architecture must be carefully chosen to reduce the memory throughput. Saving silicon can also be achieved by using external controllers for the communication with the host computer.

V. CONCLUSIONS AND FURTHER WORK

Based on an extensive analysis of the hardware design and implementation of motion estimation algorithm, a vision system targeting an FPGA device is proposed. The vision system will be in charge of the acquisition and processing of the image acquired by the camera and communication with the control system of the underwater robot running on a Pentium processor from an embedded PC/104+ computer. The constraints of our design are clearly defined by the frame-rate performance, memory access and device capacity. The linear array proposed by Yang was chosen to implement the matching problem. Testing and synthesis using CAD tools provides us information about timing and FPGA hardware utilisation of the algorithms. Our experiments show that the execution of the matching algorithm can run 50 times faster in an FPGA-based architecture than in a Pentium based PC/104+ computer. Indeed, commercial alternatives to this system exist on the market: frame-grabbers, FPGA boards for PC/104+, FPGA based dedicated image processing boards, etc. Proposed system overcome these existing solutions by providing real-time image processing facilities at low cost and small size.

REFERENCES

[1] N. Gracias and J. Santos-Victor, "Underwater video mosaics as visual navigation maps,," *Computer Vision and Image Understanding*, vol. 79 (1), pp. 66–91, 2000.

[2] X. Xu and S. Negahdaripour, "Vision-based motion sensing from underwater navigation and mosaicing of ocean floor images," in *Proceedings of the MTS/IEEE OCEANS*, 1997, pp. vol.2, 1412–1417.

[3] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of the Fourth Alvey Vision Conference*, Manchester, 1988, pp. 147–151.

[4] R. Garcia, X. Cufí, 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*, 2003, pp. 255–262.

[5] 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.*, Aug. 1998, pp. 123 –130 vol.1.

[6] N. Roma and L. Sousa, "Efficient and configurable full search block matching processors," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 12, pp. 1160–1167, December 2002.

[7] L. Vos and M. Stegherr, "Parameterizable VLSI architectures for the full-search block-matching algorithm," in *IEEE Transactions on Circuits and Systems*, October 1999, pp. 1309–1316.

[8] P. Baglietto, M. Maresca, A. Migliaro, and M. Migliardi, "Parallel implementation of the full search block matching algorithm for motion estimation," in *International Conference on Application Specific Array Processors*, July 1995, pp. 182–192.

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

[10] C. H. Hsieh and T. P. Lin, "VLSI architecture for block -matching motion estimation algorithm," *IEEE Trans. on Circuits and Systems for Video technology*, vol. 2, no. 2, pp. 169 – 175, June 1992.

[11] 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*, pp. 1317 –1325, Oct. 1989.

[12] 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.

[13] 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*, 7-9 Oct. 1996, pp. 538 – 544.

[14] F. Charot, C. Labit, and P. Lemonnier, "Architectural study of a block-recursive motion estimation algorithm," *Real-Time Imaging*, vol. 3, no. 2, pp. 111–128, 1997.