Proposal of a Parallel Architecture for a Motion Detection Algorithm

Viorela Ila, Rafael Garcia
Computer Vision and Robotics Group
Institute of Informatics and Applications
C/ Lluis Santalo S/N, 17071, Girona, Spain
{viorela,rafa}@eia.udg.es

Francois Charot
Institute de Recherche en Informatique
et en Systemes Aleatoires (IRISA)
Avenue du Gral. Leclerc,35042 Rennes, France
francois.charot@irisa.fr

Abstract

This paper proposes a parallel architecture for the estimation of motion of an underwater robot. 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. In a motion estimation algorithm, correspondences between two images has to be solved at the low level. In the underwater imaging, normalised correlation can be a solution in presence of non-uniform illumination. Due to its regular processing scheme, parallel implementation of the correspondence problem can be an adequate approach to reduce the computation time. Taking into consideration the complexity of the normalised correlation criteria, a new approach using parallel organisation of every processor from the architecture is proposed.

1. Introduction

A down-looking camera mounted on an underwater vehicle provides rich information for its navigation system [5]. Correct robot navigation requires real-time performance of tasks such as motion detection. Image processing tasks associated to motion detection algorithms use mathematical techniques dominated by convolution, correlation, filtering and least squares among others. Considering the size of the image (768×576, 416×288 pixels), these tasks have a high computational cost. Therefore, real-time execution of them (25 frames per second) requires fast-processing systems. In case of general purpose computers, achieving this performance is a great challenge. The highest performance concerning time of execution can be achieved by programming the application at gate level. However algorithms in computer vision are quite complicated and require high flexibility in the implementation. Reconfigurable computing combines the advantages of both approaches: implementation at a very low level and high flexibility and rapid prototyping [4].

Our work is focused on low level image processing algorithms for motion estimation were a large amount of data has to be processed. This paper explores the possibility of hardware implementation of tasks such as interest points detection and matching procedure. Correlation algorithms have important properties like regularity and modularity. Thus, they can be divided into computational blocks which can be processed in parallel. There is an extensive literature concerning 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, various VLSI designs were proposed for decreasing BMA computation time [1, 2]. While in full search BMA the image is divided into 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 change. On the other hand, a more complex error measurement criteria like normalised correlation [13] is applied.

The remainder of this paper is structured as follows. Section 2 presents the motion estimation algorithm and defines real-time constraints. The detailed architecture will be described in section 3. Finally, section 4 outlines conclusions and future work.

2. Analysis of the motion estimation algorithm for its parallelization

The goal of this algorithm is to estimate the motion of an underwater robot. Correspondences between the current image acquired by the camera and a reference image have to be found in order to estimate the motion. This often means detecting features in one image and matching them in another. 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 [7]. However, matching those features in the second image is normally a complex task. 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*) which can introduce errors in the motion estimation process. Some authors have proposed a normalised correlation to reduce the influence of non-uniform illumination [13].

2.1. Corner detector

In our algorithm motion is estimated by computing the planar homography between the current image I_c and a previous reference image I_r. 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 version of the Tomasi-Kanade [11] algorithm which reduces the computation and avoids floating-point. In this algorithm a G matrix is considered.

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

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

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

is found. Every pixel having:

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

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 which 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.

2.2. Correspondence problem

Once interest points are detected in the current image, we search for correspondences in the reference image. Quite often local gray-level correlation is applied to detect matchings in the pair of images. A correlation algorithm provides, for each interest point $p_c = (x_c, y_c)$ of the current image, its corresponding match $p_r = (x_r, y_r)$ 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 candidate windows inside the correspondent *search window*. A normalised correlation criteria C, which assures the result is not altered in presence of nonuniform illumination is showed in equation (4). This criteria was applied to underwater images [5] where nonuniform illumination is always present.

$$C = \frac{\sum_{-\alpha}^{\alpha} \sum_{-\alpha}^{\alpha} (I_c(x_c+i, y_c+j) - \overline{I_c(x_c,y_c)})(I_r(x_r+i, y_r+j) - \overline{I_r(x_r,y_r)})}{(2\alpha+1)^2 \sqrt{\sigma^2(I_c) \cdot \sigma^2(I_r)}} \tag{4}$$

where $\alpha = (n-1)/2$; $n \times n$ 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 algorithm compares the correlation score of each pixel within the search window and selects the highest one.

As the amount of interest points increases, the correlation approach becomes very time consuming. For this reason we propose a breaking down of criteria C for its parallelization. We can observe that there are five sums to be computed in equation (4): sum_1, sum_2, sum_3, sum_4 and sum_5.

$$\begin{aligned} sum_1 &= \sum_{i=-\alpha}^{\alpha} \sum_{j=-\alpha}^{\alpha} I_c(x_c+i, y_c+j) \\ sum_2 &= \sum_{i=-\alpha}^{\alpha} \sum_{j=-\alpha}^{\alpha} I_c(x_c+i, y_c+j)^2 \\ sum_3 &= \sum_{i=-\alpha}^{\alpha} \sum_{j=-\alpha}^{\alpha} I_c(x_c+i, y_c+j) \cdot I_r(x_r+i, y_r+j) \\ sum_4 &= \sum_{i=-\alpha}^{\alpha} \sum_{j=-\alpha}^{\alpha} I_r(x_r+i, y_r+j)^2 \\ sum_5 &= \sum_{i=-\alpha}^{\alpha} \sum_{j=-\alpha}^{\alpha} I_r(x_r+i, y_r+j) \end{aligned} \tag{5}$$

Then, equation (4) becomes:

$$C = \frac{sum_3 - \frac{1}{(2\alpha+1)^2} \cdot sum_c \cdot sum_5}{\frac{1}{(2\alpha+1)^2} \cdot \sqrt{[(2\alpha+1)^2 \cdot sum_r - sum_c^2] \cdot [(2\alpha+1)^2 \cdot sum_4 - sum_5^2]}} \tag{6}$$

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

Finding correspondences is the most time consuming part of our algorithm. Let us consider N interest points detected in the current image. For every interest point, we are searching for $[(2p+1) - 2\alpha]^2$ possible correspondences, where $p = (q-1)/2$ and $q \times q$ is the search window of size. Considering the braking down of the correlation cri-

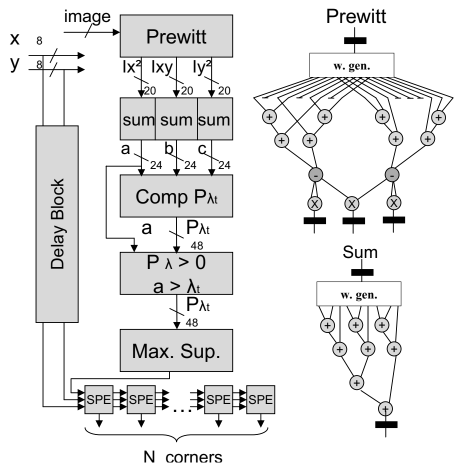

Figure 1. Corner detector Block Diagram. Prewitt mask DFG. Summing DFG.

teria in five sums as described in equation (5), two accumulations and three multiplication-accumulations have to be computed for every pixel from the correlation window. Gathering these sums by means of correlation criteria, see equation (6), twelve additional computation steps for every candidate block are necessary. The complexity of the correspondence problem becomes at frame-rate f_r:

$$O_p = [(3{*}2{+}2){*}(2\alpha{+}1)^2{+}12]{*}[(2p{+}1){-}2\alpha]^2{*}N_p{*}f_r \tag{7}$$

For $N_p = 200$, $\alpha = 7$, $p = 14$, at a frame-rate of 25 frames per second we have $O_p \simeq 2036$ GOPS (Giga Operations Per Second). Our approach tries to reduce this complexity by means of a parallelization of the correspondence problem. Real-time feature detection is also achieved.

3. Proposal of a Parallel Architecture

3.1. Corner detector hardware implementation

The current image is read from memory and the goal of the corner detector is to provide the memory address of N interest points of the image. 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 left column of Figure 1 shows the block diagram corresponding to each step in corner detection. 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 Figure 1. The computation of matrix G implies summing the correspondent values of $a = I_x^2$, $b = I_x \cdot I_y$ and $c = I_y^2$ for every pixel in an $m \times m$ window, selected experimentally to be 3×3. The next step consists of computing P_{λ_t} from equation (2) and rejects the values which do not satisfy the conditions of equation (3). Non-maximum suppression is carried out using a 3×3 window. In order to retain N pixels with the highest value of P, a pipeline of N Sort-Processing Elements (SPE) is proposed. One SPE compares the input pixel value with the one stored in its buffer and retains the bigger one. An external signal can empty the SPEs buffers at the end of each frame.

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 shown in equation 8 and depends on the image size $(M_i \times N_i)$, pixel sampling time (t_s) and the number of time-cycles for the computational blocks from Figure 1: *Prewitt* (t_P), *Sum* (t_S) and *Compute* P_{λ_t} (t_C).

$$T_{P_\lambda} = [3 \cdot (2 \cdot M_i + 3) + P_c + S_c + C_c] \cdot t_s \tag{8}$$

3.2. Parallel implementation of the correspondence problem

For every interest point we are looking for correspondences in the reference image. When mapping an algorithm into an array of processors, the problem is to access multiple data to feed all the processing elements (PE) at the same time. Yang et al. [10, 12] proposed a solution which consists of a local data exchange between PEs. This approach uses a two memory access for reference image (r_1, r_2) and one for current image (c), see Figure 2. Once read from memory, the data are broadcasted to every PE. Buffers are used to delay data and multiplexers to switch between data. For high utilization efficiency of the architecture, the size of the search window must depend on the size of the correlation window, according to equation $p = 2\alpha$. The number of PEs is also determined by the size of the correlation window and is equal to $(2\alpha + 1)$. A schematic representation of the specific hardware architecture is shown in Figure 2. One PE is in charge of the parallel computation of the five sums defined in equation (5). Two accumulations and three multiplication-accumulations are executed in parallel (Figure 3(a)). For a given interest point, the necessary time to search for the computation of the five sums of equation (5) is defined by:

$$T_p = [(2\alpha + 1)^2 + 2\alpha] * (2\alpha + 1) * \Delta t \tag{9}$$

where Δt is the time required for one computational level. After T_p seconds, the post processing element can compute the correlation criteria from equation (6). The DFG for this computation is shown in Figure 3(b). Hardware implementation of square roots and division operations is crucial for

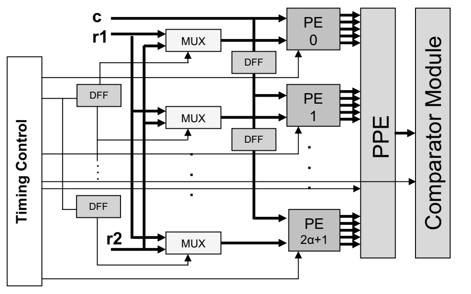

Figure 2. Schematic representation of PE based architecture data-flow.

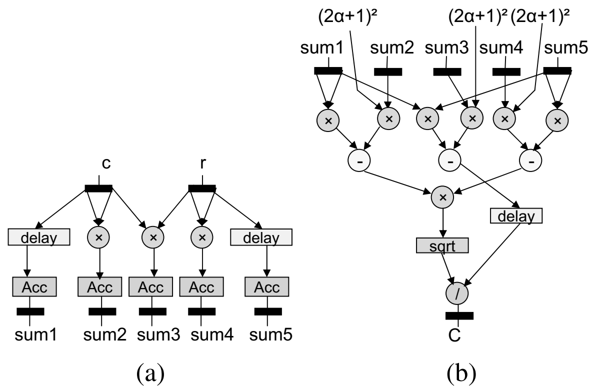

Figure 3. (a) PE Data-Flow Graph. (b) Post PE Data Flow Graph.

the delay introduced by PPE element. The square root algorithm provides an integer part of the square root end a reminder. 32 clock cycles are necessaries for this computation. Considering the high number of bits used for the radicand representation, a non-restoring square proposed by Li [9] root method is an optimal selection for reducing the space on the FPGA. 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 highest value for the correlation score.

4. Conclusion and future work

This paper describes a real-time and parallel implementation of two high-cost computational tasks from a motion estimation algorithm. We propose a real-time feature detector which provides N interest points from the current image. An efficient array of processing elements is in charge of the computation of the correlation algorithm. Future work consists of optimal implementation of the post-processing element. Taking into consideration the evolution of reconfigurable device's technology in the last few years this allows to increase the complexity of our design. New powerful devices with characteristics such as a million of logic gates, memory resources for on-chip storage, fast multipliers, large number of input/output pins, etc., are available on the market at low prices. The goal of this work is to advance in developing a system for motion estimation of an underwater robot.

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] A. DeHon and J. Wawrzynek. Reconfigurable computing: what, why, and implications for design automation. In *Design Automation Conference*, pages 610–615, 1999.

[5] 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*, 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] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the Fourth Alvey Vision Conference*, pages 147–151, Manchester, 1988.

[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.