A Reconfigurable Architecture for Robotic Stereo Vision

John Kalomiros
Department of Informatics and Communications,
Technological Educational Institute of Serres,
Terma Magnisias, 62124 Serres, Greece

John Lygouras
Department of Electrical Eng. & Computer Eng., School of Engineering, Democritus University of Thrace,
Xanthi, Greece

Abstract— **A reconfigurable architecture for dense stereo is presented as an observation framework for a real-time implementation of the simultaneous localization and mapping problem in robotics. The reconfigurable sensor detects point features as corners from stereo image pairs, in order to use them at the measurement update stage of the procedure. The main hardware blocks are a dense depth stereo accelerator, a left and right image corner detector and a stage performing left-right consistency check for the detected features. For the stereo-processor stage we have implemented and tested both a local-matching method based on the Sum of Absolute Differences (SAD) and a global-matching component based on a maximum-likelihood dynamic programming technique (MLDP). The system includes a Nios II processor for data control and a USB 2.0 interface for host communication. The proposed hardware is applied as the sensor part in a real-time robotic localization and mapping experiment with the help of a small guided vehicle.**

Keywords-component; Reconfigurable systems; machine vision; Real time systems

I. INTRODUCTION

Next generation intelligent vehicles technology depends heavily on the successful implementation of vision-based systems for navigation, real-time obstacle detection and path planning [1]. In particular, the simultaneous localization and mapping (SLAM) problem has been given great attention in recent years, since it holds the key to robust navigation in unknown environments [2].

Feature extraction is a prerequisite for robot mapping and localization [3]. In recent years attention has been given to monocular and stereo-vision systems and methods have been proposed to extract reliable features from image data [4]. Various vision-based features have been used in the literature. Such are Harris corners and Shi-Tomasi features [5], SIFT descriptors [6], edge segments etc. Visual feature extraction and tracking in real-time is computationally demanding, particularly since the data rates coming from camera are much higher than those from other sensors.

Stereo-vision provides a solid framework for the extraction of 3D structure from image data. Finding the correspondences between left and right image frames allows depth computation for scene points. Solving the correspondence problem is not trivial and the derivation of dense disparity maps is again a very demanding computational problem, since it requires extensive searching and optimization along scanlines [7].

Field programmable gate arrays represent a flexible and efficient solution for accelerating stereo matching computations and other complex image processing tasks. In this paper a real-time point-feature extraction technique based on stereo vision is proposed and is implemented in reconfigurable hardware. Left and right image information is input to the system as a stream of 8-bit gray-scale pixels and is processed by several hardware stages integrated in a system-on-a-programmable-chip. The first stage is a stereo-processor. Next stages are left and right image corner detectors, thinning stages and a final stage performing left-right consistency check. The integrated system features also a Nios II processor for data control, an external memory interface, DMA functions and a USB 2.0 controller for communication with a host computer.

The above system is adapted to a simple mobile vehicle and is used as the sensor part in a simultaneous localization and mapping experiment. On the host part an application receives the stream of feature positions with their respective disparities. A FastSLAM algorithm is used to track and optimize the vehicle path and the map feature locations.

The rest of the paper is organized as follows: In Section II the overall system is outlined as a block diagram. In Section III the stereo processing stage, the corner detection and the stage for left-right consistency check are presented. In Section IV the Nios system on a programmable chip is presented. In Section V the proposed hardware is used as the landmark measurement part for a 3-D FastSLAM localization and mapping experiment.

II. THE OVERALL FEATURE DETECTOR

The system is modeled and simulated using DSP Builder design software. It is synthesized with Quartus II CAD software by Altera corporation and is implemented targeting a Cyclone II device on a DSP development board. Fig. 1 shows a block diagram of the overall system. A stereo head is developed in the laboratory with two parallel cameras adjusted

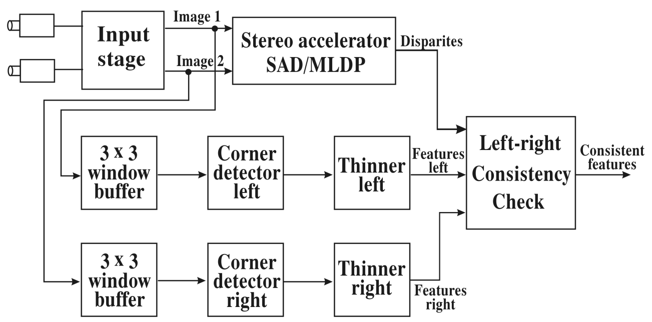

Figure 1. Block-diagram of the overall system

in order to produce rectified image pairs. First, a dense depth map is produced by the stereo accelerator stage. Left and right image pixel streams are processed in parallel by the stereo processor. The stereo processor finds the correspondences between left and right image pixels and produces the disparities $d=u_L-u_R$, where u_L and u_R are pixel coordinates on a scanline. It can produce 16, 32, 49 or even more levels of disparity, based on the same principles but making use of additional hardware resources. The stereo stage can implement any method of local correlation or global scanline optimization provided that they produce a synchronized output in parallel with the input data streams. We test and compare two different implementations of stereo stages, the first based on a local method of correlation and the second on a demanding maximum-likelihood optimization algorithm based on dynamic programming (DP).

In the feature extraction stage, corner detectors implement a simple edge detection principle based on convolution with 3x3 Prewitt horizontal and vertical masks. Corners are produced by performing a binary AND operation between horizontal and vertical edges. Thinning stages apply a thinning algorithm based again on convolution with Prewitt masks and result in a well defined point-feature at each image corner. In the final stage the consistency of point-features in the left and right image frame is tested by comparing their horizontal displacement with the disparity values produced by the stereo stage. Only features surviving the test are transmitted to the host application and are processed in the 3-D map reconstruction phase.

III. Analysis of the Embedded System Stages

A. Hardware implementation of SAD

SAD represents a wide range of techniques that find correspondences by comparing local windows along epipolar lines in left and right images of the stereo-pair [7]. It has the advantage of a particularly simple and hardware-friendly metric of similarity, namely the sum of absolute differences:

$$\sum_{u,v} \left| (I_1(u+x, v+y) - I_2(u+x+d, v+y) \right| \cdot \tag{1}$$

I_1 and I_2 refer to intensities in the left and right image, (x,y) is the centre of the window in the first image, $(x+d,y)$ is a point on the corresponding scan-line in the other image displaced by d with respect to its conjugate pair and u,v are indices inside the window. The point that minimizes the above measure is selected as the best match. This metric reduces hardware requirements only to additions and subtractions between intensities in the local windows.

While this method requires laborious search along epipolar lines in serial software implementations, it can be parallelized in hardware, allocating parallel comparisons between local windows to a number of processing elements. In our system, shown in Fig. 2, we applied window-level parallelism across 32 or 64 levels of disparity. A comparison window is created using shift registers, as shown in Fig. 2. The main processing element finds the absolute differences between respective pixels and sums the differences within each window. A number of D processing elements are needed for D-levels of disparity.

After the D SAD values are produced in parallel, their minimum value is found in one clock cycle, using an array of parallel comparators. Each pixel in the search region is attributed a tag index, from 0 to D. Among all D pixels in the search region the one pixel that corresponds to the minimum SAD value wins. The derived tag index of the winning pixel is equal to the actual disparity value. Details of this implementation can be found in [8].

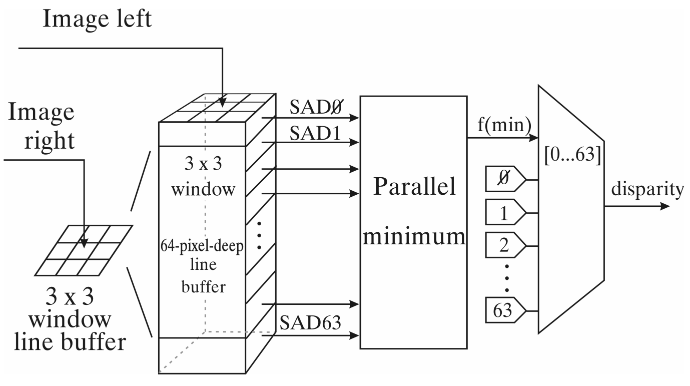

Figure 2. Block-diagram of the SAD stereo-processor

B. Hardware design of the DP stereo accelerator

The second stereo algorithm implemented and tested performs semi-global matching based on a dynamic programming optimization method. The algorithm is a method for maximizing likelihood [9], which is computationally demanding and difficult to implement in real-time without acceleration. We refer to the algorithm as ML-DP. It is developed in two phases, namely the cost-plane building phase and the backtracking phase. The cost-plane is computed as a two-dimensional matrix of minimum costs, one cost-value for each possible correspondence $I_i \leftrightarrow I_j$ between left and right image intensity values, along a scan line. One always proceeds from left to right, as a result of the ordering constraint. Each point of the two-dimensional cost function is derived as the minimum transition cost from the three neighboring cost values. Transition costs result from previous costs, adding a matching or occlusion cost, s_{ij} or *occl*, according to the following recursive procedure:

$$C(i,j) = \min\{C(i-1,j-1)+s_{ij}, C(i-1,j)+occl, C(i,j-1)+occl\} \quad (2)$$

In the above equations, the matching cost s_{ij} is minimized when there is a match between left and right intensities. The following dissimilarity measure was used:

$$s_{ij} = \frac{(I_l(i) - I_r(j))^2}{\sigma^2}, \quad (3)$$

where σ represents the standard deviation of pixel noise.

In order to parallelize the cost-matrix computation in hardware, we design a variation of the maximum likelihood algorithm, using an adequate slice of the cost-matrix above the diagonal of the cost plane. We take into account only the part of the cost plane, where the minimum cost path is contained. Working within this slice, along the diagonal, allows a subset of D cost states perpendicular to the diagonal to result in parallel from the preceding subset of cost states, in step with the input stream of left and right image scanlines.

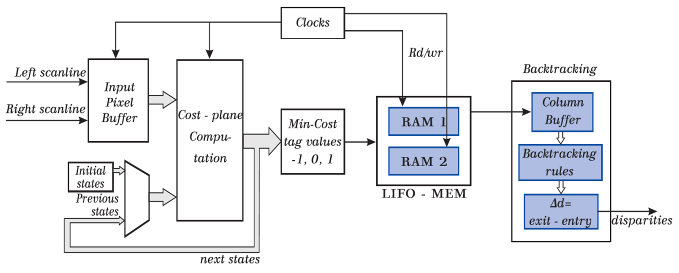

Figure 3. Block diagram of the stereo accelerator based on ML-DP

Fig. 3 shows a block diagram of the stereo accelerator hardware system. The processing element computes the cost of the diagonal, vertical and horizontal transition path to each adjacent point of the cost matrix plane. The minimum among the three cost values is produced by a min-computation parallel stage. Tag values (-1, 0, 1) are attributed to all three possible paths and the wining tag, corresponding to the minimum cost, is stored in RAM memory, at each point of the cost plane. In this way, it is possible to calculate all states in the slice, up to the end of the cost-plane, using the same hardware stage. This stage receives as input the previous costs, along with the current left and right pixel intensities and produces the next subset of costs, like a state machine. This idea is contained in the feedback loop of the cost-plane computation stage in Fig. 3.

RAM memory is implemented as M4K blocks, N positions deep, where N is the number of pixels per scanline. A number of D RAM blocks are needed for a maximum disparity of D pixels. Each position is 2-bit wide, since it only stores a tag value -1, 0 or 1.

During backtracking, we compute N disparities for each one of the N scanline pixels, in N clock cycles. Winning tags are read from RAM memory in reverse order and are ordered according to the columns of the cost plane, using shift registers. At each step we move from one column to the next, following the optimum path, according to the retrieved tag values and using well-defined rules [9]. At each step the optimum path traverses vertically a tag column by a number of pixels equal to the change Δd of disparity at this particular step. Starting with d=0 at the N_{th} pixel, the system tallies the disparity values down to the first pixel, adding Δd at each step. Details of this system can be found in [10].

C. *Corner detector, thinner and left-right consistency check*

As mentioned above corners in the left and right images are formed at the cross-section of vertical and horizontal edges. In order to produce image gradients we convolute the images with 3x3 Prewitt masks. Image areas are formed with shift lines that store streaming input pixels (Fig. 1). Using 3x3 windows results in a sensitive edge detector and at the same time keeps the required hardware resources low. Since the Prewitt masks contain only zeros and ones the convolution is implemented with adders and subtractors, as it is shown in [11]. A binary AND operation produces thick white spots at the cross section of horizontal and vertical edges. White spots are ones, black areas are zeros. A thinning procedure is applied in the next stage, implementing the steps of the following algorithm:

```
a. Convolute a 3x3 area of the binary corner image with horizontal Prewitt mask.
b. Convolute the same 3x3 area with the vertical Prewitt mask.
c. IF the central pixel in the 3x3 area is zero OR any of the convolutions in steps a and b is greater than 1
    THEN central pixel remains zero
    ELSE central pixel is 1
```

The idea behind the above hardware-friendly steps is to crop clusters of ones until only a central point remains.

Finally, the consistency check is implemented by combining the disparity values produced in the stereo stage with the point features found above. Stereo and corner stages work in parallel and the disparities stream can be easily aligned with the corner image stream by means of a delay line. Each corner feature in the left stereo frame is compared to the corresponding pixel in the right image frame. The corresponding feature is found by displacement according to the disparity value. In hardware we simply use shift taps and a multiplexer in order to perform the above check.

IV. SYSTEM-ON-CHIP AND HOST APPLICATION

The system was created as a HDL library component, ready for integration in a System-on-a-programmable-chip. It is controlled by a 32-bit Nios II processor, and includes a DDR2 external memory controller and a USB2.0 high speed controller for communication with a host computer. The feature detection hardware is integrated in the form of HDL library component. A set of DMA functions transfers data between units. The SOPC system is shown in Fig. 4 as a block diagram. We implement the system targeting a Cyclone II DSP board featuring a 2C35 FPGA device using approximately 27500 logic elements and 413632 memory bits out of 33,000/480000 total resources. A frame-grabber is also developed in the Cyclone device, while the system transfers data to a host application via a USB2.0 controller achieving a practical transfer rate greater than 60 Mbps. The host application receives the output from the hardware accelerator and uses feature positions (u_L, v_L) and disparities d as discrete landmark measurements. These measurements are used in the update stage of a FastSLAM simultaneous localization and mapping algorithm, running on the host side. This procedure is presented in Section V.

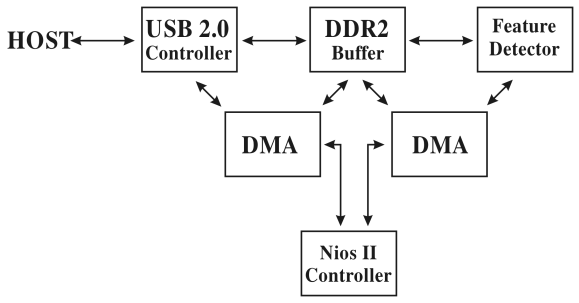

Figure 4. System-on-a-chip based on Nios-II processor

V. Experimental Results and Conclusions

Localization and mapping experiments were carried out using an observation model based on point-feature measurements. A vehicle is guided indoors by a human operator, recording features in a 5.5x4.2 m room, following an almost circular path and completing one or two rounds. The speed on the direction of motion is kept constant and equal to 0.15 m/s. The main body of the localization and mapping algorithm is executed on a host computer which receives the feature stream produced by the proposed hardware via the USB2.0 connection. The host application builds an observation vector out of the feature measurements and attributes a 3-D landmark position to each measurement, taking into account the robot pose. Robot pose and the map are updated at each step following Montemerlo's FastSLAM algorithm [12]. Fig. 5 presents a result of mapping the room and simultaneously localizing the vehicle, employing ten particles in the particle filter of our FastSLAM implementation. In this figure, map features are shown with dots at the room's perimeter, while the circular line records the vehicle path, as estimated by the algorithm. As is shown in this figure, acceptable maps and robot paths can be estimated even with few particles. Both SAD and ML-DP stereo processors described in the previous

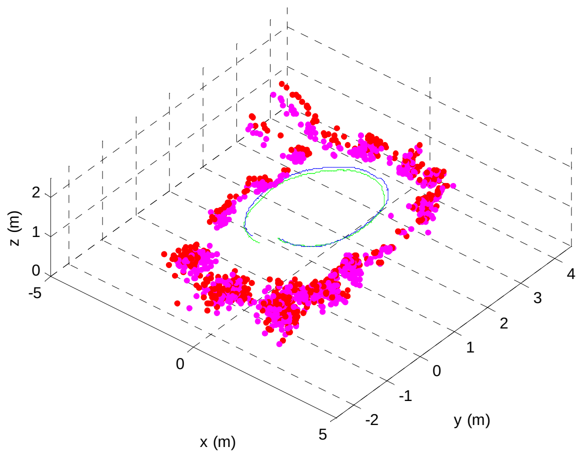

Figure 5. Map and robot path produced by indoors experiment

sections can be used, while the ML-DP stage produces better results with equivalent hardware resources.

References

[1] W. van der Mark and D. Gavrila, "Real-time dense stereo for intelligent Vehicles", IEEE Transactions on Intelligent Transportation Systems, vol. 7, no. 1, pp. 38-50, 2006.

[2] S. Thrun, "Robotic Mapping: A Survey. Exploring Artificial Intelligence in the New Millenium", Morgan Kaufmann, 2002.

[3] S. Se, D. Lowe, J. Little, "Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks", Int. Journ. Robotics Research, vol. 21, no. 8, pp. 735-758, 2002.

[4] A. Davison, I. Reid, N. Motton, and Olivier Stasse, "MonoSLAM: Real-Time Single Camera SLAM", IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 29, no. 6, pp. 1-16, 2007.

[5] J. Shi, and C. Tomasi, "Good features to track", IEEE Computer Society Conference on Computer Vision and Pattern recognition (CVPR '94), pp. 593-600, June 1994.

[6] D. G. Lowe, "Distinctive Image Features from Scale Invariant Keypoints", International Journal of Computer Vision, vol. 60 (2) pp.91-110, 2004.

[7] M. Brown, D. Burschka, and G. Hager, "Advances in Computational Stereo", IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 25, no. 8, pp. 993-1008, Aug. 2003.

[8] J. Kalomiros and J. Lygouras, "Hardware implementation of a stereo co-processor in a medium-scale FPGA", IET Computers and Digital Techniques, vol. 2, No 5, pp. 336-346 (2008).

[9] I. J. Cox, S. Hingorani, S. B. Rao, and B. M. Maggs, "A Maximum Likelihood Stereo Algorithm", Computer Vision and Image Understanding, vol. 63, no. 3, pp. 542-567, 1996.

[10] J. Kalomiros and J. Lygouras, "Design and hardware implementation of a stereo-matching system based on dynamic programming", Microprocessors and Microsystems, vol. 35, pp. 496-509 (2011).

[11] J. Kalomiros and J. Lygouras, "Design and Evaluation of a hardware/software FPGA-based system for fast image processing", Microprocessors and Microsystems, vol. 32 (2) (2008) 95-106.

[12] M. Montemerlo, S. Thrun, "FastSLAM, a scalable method for the Simultaneous Localization and Mapping problem in Robotics", Springer Tracts in Advanced Robotics, vol 27, B. Sicilinao, O. Khatib, F. Groen, Editors, Springer-Verlag, Berlin Heidelberg, 2006.