Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2010, Article ID 480208, 17 pages
doi:10.1155/2010/480208

Research Article

Robotic Mapping and Localization with Real-Time Dense Stereo on Reconfigurable Hardware

John Kalomiros[1] and John Lygouras[2]

[1] *Department of Informatics and Communications, Technological Educational Institute of Serres, Terma Magnisias, 62124 Serres, Greece*
[2] *Section of Electronics and Information Systems Technology, Department of Electrical Engineering & Computer Engineering, School of Engineering, Democritus University of Thrace, 67100 Xanthi, Greece*

Correspondence should be addressed to John Kalomiros, ikalom@teiser.gr

Received 1 March 2010; Revised 20 July 2010; Accepted 20 November 2010

Academic Editor: Viktor K. Prasanna

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 from stereo image pairs to use 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 stereo-processor stage, we have implemented and tested a global-matching component based on a maximum-likelihood dynamic programming technique. The system includes a Nios II processor for data control and a USB 2.0 interface for host communication. Remote control is used to guide a vehicle equipped with a stereo head in an indoor environment. The FastSLAM Bayesian algorithm is applied in order to track and update observations and the robot path in real time. The system is assessed using real scene depth detection and public reference data sets. The paper also reports resource usage and a comparison of mapping and localization results with ground truth.

1. Introduction

Vision-based robotic localization and mapping is one of the most active research areas in mobile robotics. 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 immense attention in recent years, since it holds the key to robust navigation in unknown environments. SLAM is a class of stochastic algorithms that address the problem of estimating concurrently the robot's path and a map of the surrounding environment [2]. These algorithms are based on a nonlinear model for robot dynamics and on a landmark observation model derived from the particular nature of the sensor used for data gathering.

Feature extraction is a prerequisite for robot mapping and localization, which in turn is a central problem for the navigation of autonomous vehicles [3]. Active sensors like sonars and laser range finders are commonly used for the purpose of feature extraction for localization and mapping [4]. They are effective in static, low density, and low noise environments, but are also expensive, heavy, and prone to environmental interference. Passive systems, like vision systems, are much less sensitive to environmental interference. Recently attention has been given to monocular or stereo-vision systems, and methods have been proposed to extract reliable features from image data [5]. Various vision-based features have been used in the literature. Such are Shi-Tomasi features [6], SIFT descriptors [7], edge segments [8], Harris corners [9], and so forth. 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, however, since occlusion, specularities

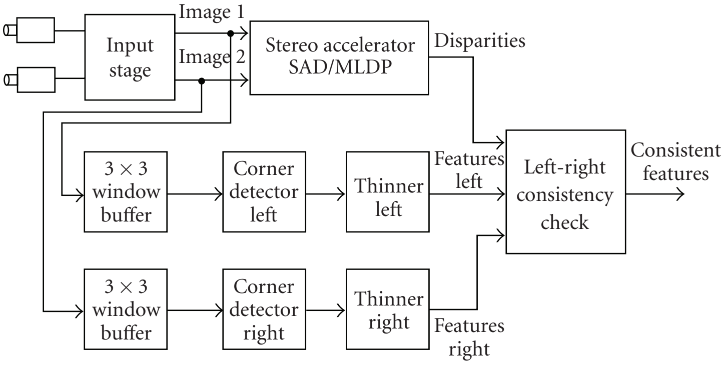


Figure 1: Block diagram of the point-feature detector.

or lack of texture can lead to wrong matches. The derivation of dense disparity maps from stereo-pairs is again a very demanding computational problem, since it requires extensive searching and optimization along scanlines [11]. A taxonomy and evaluation of different correspondence methods is given in [12]. Cost aggregation methods for real-time stereo matching are evaluated in [13].

Vision-based feature extraction for robotic mapping and localization is a very intensive computational procedure and is known as the main source of delay in visual SLAM. Moreover, stereo-assisted feature extraction has high-computational demands, especially for global stereo matching. Field programmable gate arrays represent a flexible and efficient solution for accelerating stereo matching computations and other complex image processing tasks. Their fine-grain structure of small logic elements allows parallelism combined with high processing speeds. They also present an advantage over Application Specific Integrated Circuits (ASICs) because they are reprogrammable and much cheaper for prototyping.

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 able to produce dense depth in the form of 8-bit disparities in parallel with input data. A global matching maximum-likelihood algorithm is implemented and its suitability for consistent feature extraction is tested. The algorithm provides optimized disparity computation across scanlines and addresses the problem of depth extraction at occlusion boundaries but has considerable computational cost. Other stages are left and right image corner detectors and thinning stages resulting in a pair of binary images with well-localized point features. The final stage performs a left-right consistency check, taking into account the disparity values produced by the stereo stage. Point-features surviving the check are output from the final stage. 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. An introductory description of this implementation was given in [14] and here the system is evaluated and tested extensively.

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 main contribution of this paper is twofold. It presents an original integrated sensor producing landmark measurements, based on computationally demanding dense stereo techniques. Second, the system is tested and assessed in terms of accuracy and real-time performance in a state-of-the-art Bayesian algorithm for simultaneous localization and mapping. The presented system can be used as a framework to develop more sophisticated dense depth solutions to vision-based robotic navigation.

The rest of the paper is organized as follows. In Section 2, the overall system is outlined in a block diagram. In Section 3 the stereo processing stage, the corner detection and the stage for left-right consistency check are presented. In Section 4 the system is evaluated in terms of frame rates and hardware resource usage. In Section 5, the presented hardware is used as the measurement stage for a 3D FastSLAM localization and mapping experiment. Section 6 compares the results with similar efforts presented in the literature and Section 7 concludes the paper.

2. The Overall Feature Detection System

Figure 1 shows a block diagram of the overall system. A stereo head is developed in the laboratory with two parallel cameras carefully adjusted 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

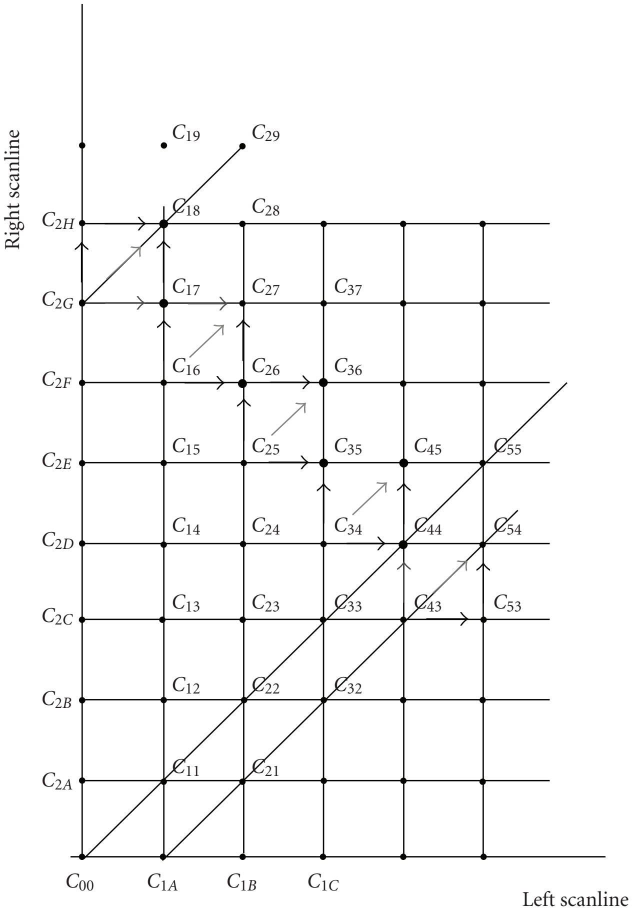


FIGURE 2: Cost-plane matrix and a diagonal slice for parallel computations.

that they produce a synchronized output in parallel with the input data streams. In our stereo-assisted feature extraction scheme the stereo correspondence accelerator implements 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 3×3 Prewitt horizontal and vertical masks. In principle, any edge detection method is applicable; however the Prewitt masks are hardware friendly since they perform convolution computations only with adders and subtractors. The same 3×3 window used for correlation in the stereo stage can be used for the purpose of convolution in the corner detection stage. A thresholding scheme is applied in order to produce binary horizontal and vertical edge images from each input image. Corners are produced by simply 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 3D map reconstruction phase.

3. Analysis and Design of the Embedded System Stages

3.1. Hardware Design of the DP Stereo Accelerator. Dense depth is extracted by matching left and right rectified image pairs captured by the stereo head. In this paper a semiglobal matching algorithm is used, based on a dynamic programming optimization method. The algorithm is computationally demanding and difficult to implement in real-time without acceleration. As pointed out by Cox et al. [15] and Brown et al. [11] the computational complexity of an algorithm matching all pixels in a stereo pair using dynamic programming is $O(N^2 \times M)$, where N, M are the horizontal and vertical image dimensions.

Dynamic programming for stereo is mathematically and computationally more complex than local correlation methods, since stereo correspondence is derived as a globally optimum solution for the whole scan line [16]. The algorithm used in this paper is a method for maximizing likelihood, which is equivalent to minimizing a cost function [15]. The

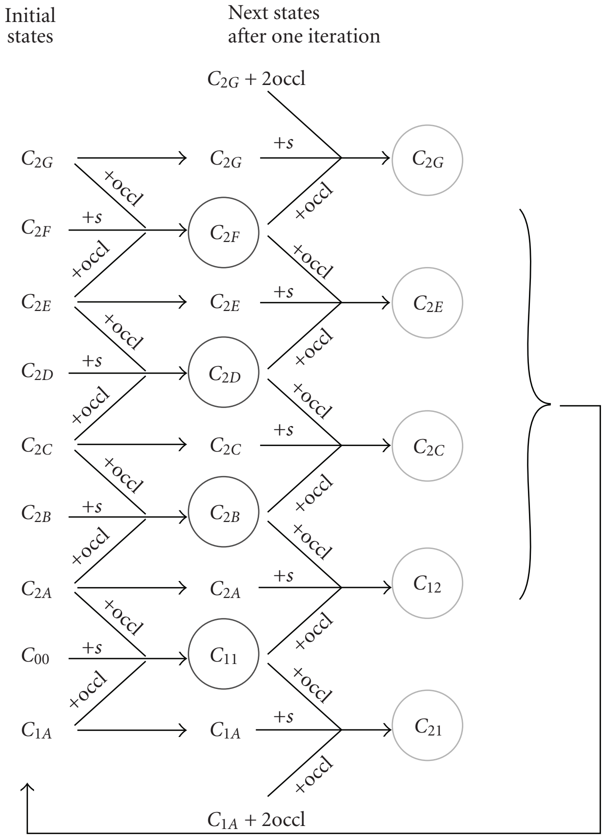


Figure 3: Successive cost-state computation with feedback, using an example nine-state machine, for the cost-plane of Figure 2.

algorithm 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. This procedure is shown in Figure 2, where 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:

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

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

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

where σ represents the standard deviation of pixel noise. Typical values are $\sigma = 0.05 - 0.12$ for image intensities in the range $[0,\ 1]$. In our implementation, we calculate s_{ij} within a 3×3 window applied in both images. The occlusion cost occl is the cost of pixel j in the right scanline being skipped in the search for a matching pixel for i and in our tests takes a value occl = 0.2.

The cost-matrix computation is a recursive procedure in the sense that for each new cost $C(i, j)$ the preceding costs on the same raw and column are needed, according to (1). In turn, previous costs need their precedent costs, rendering the parallel computation of costs intractable. In order to parallelize the cost-matrix computation in hardware, we design a variation of the DP algorithm, using an adequate slice of the cost-matrix above the diagonal of the cost plane, as shown in Figure 2. 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. D represents the maximum disparity range. Figure 2 shows a slice along the diagonal supporting, by way of example, a maximum disparity of 9 pixels. Starting from a known initial state (here c_{1A}, c_{00}, c_{2A}, c_{2B}, c_{2C}, c_{2D}, c_{2E}, c_{2F}, and c_{2G} lying on the axes), the next subset of cost states is calculated. For this purpose, the cost states are grouped as triads and occlusion (occl), and matching costs (s_{ij}) are added to the previous states, as shown in Figure 3. In this way, the processing element computes the cost of the

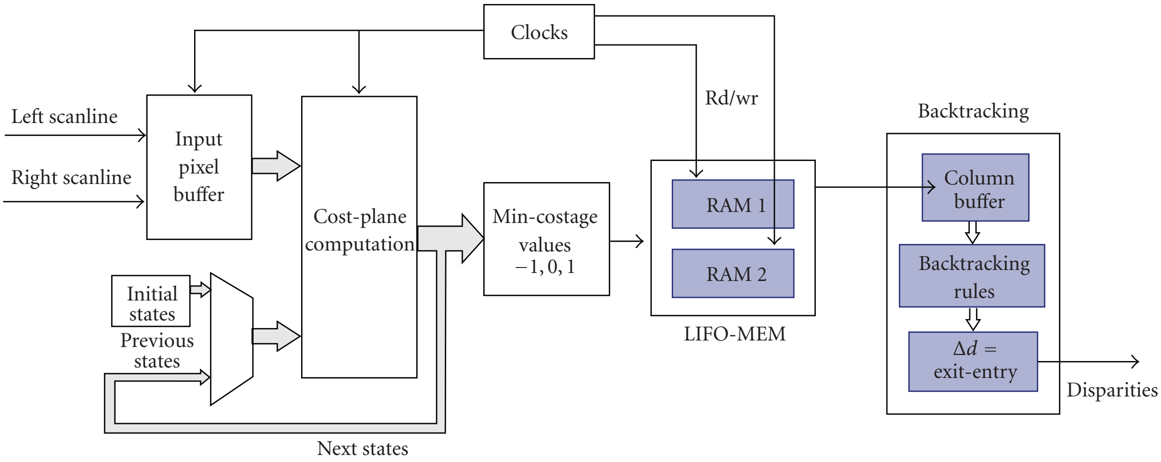


Figure 4: Hardware system based on maximum-likelihood dynamic programming algorithm.

diagonal, vertical, and horizontal path to each adjacent point. The minimum among the three cost values is produced by an appropriate 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. As shown in Figure 3, at each-one iteration the next subset of cost states are calculated in two phases. The nine states shown in circles are all the cost states needed for the next round of computations.

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. Pixel intensities are used in the matching cost computation of (2).

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 (nine in the case of the state machine working on the cost plane of Figure 2). 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 [15]. 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 Nth pixel, the system tallies the disparity values down to the first pixel, adding Δd at each step.

Figure 4 shows a block diagram of the stereo accelerator hardware system.

3.2. The Corner Detector Stage. 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 3×3 Prewitt masks. Image areas are formed with shift lines that store streaming input pixels. Using 3×3 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 shown in Figure 5. In the inset, the Prewitt mask for horizontal gradient is shown, along with the respective image window where the convolution is applied.

Vertical gradient is produced with the transposed Prewitt mask. A threshold value equal to 80 is used in order to produce horizontal and vertical binary edges. 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. In 8-bit intensity terms ones are 255.

A thinning procedure is applied in the next stage, implementing the steps of the following algorithm:

(i) Convolute a 3×3 area of the binary corner image with horizontal Prewitt mask *M1*;

(ii) Convolute the same 3×3 area with the vertical Prewitt mask *M2*;

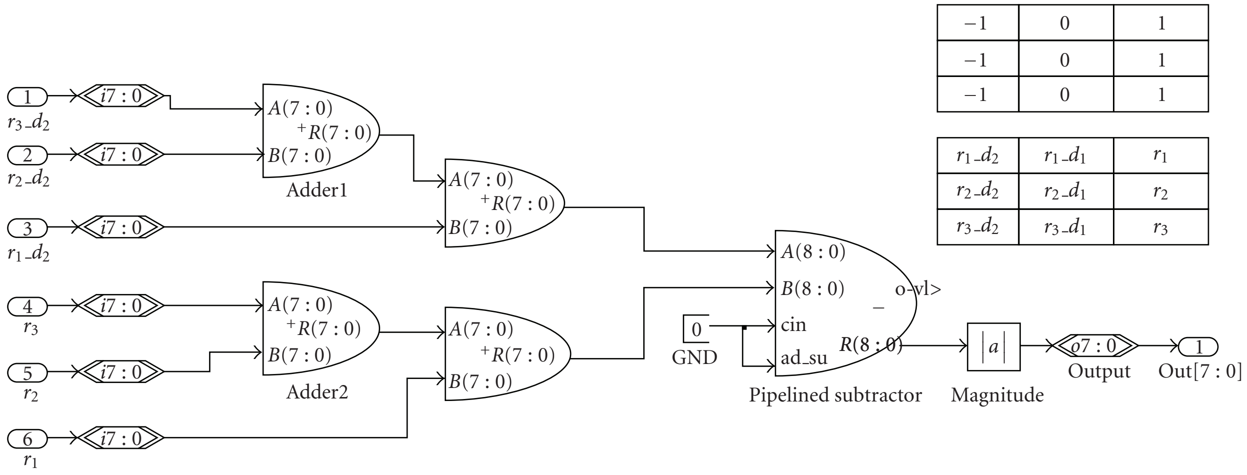
(iii) IF the central pixel in the 3×3 area is zero OR any of the convolutions in steps a and b is greater than 1

  (a) THEN central pixel remains zero;

  (b) 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. Point (iii) in the above algorithm is implemented according to Figure 6.

Finally, the consistency check is implemented by combining the disparity values produced in the stereo stage with the point features found above. This stage is shown in Figure 7. 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



−1	0	1
−1	0	1
−1	0	1

$r_1\_d_2$	$r_1\_d_1$	r_1
$r_2\_d_2$	$r_2\_d_1$	r_2
$r_3\_d_2$	$r_3\_d_1$	r_3

Figure 5: Application of the Prewitt mask for horizontal gradient computation.

$$M1 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad M2 = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

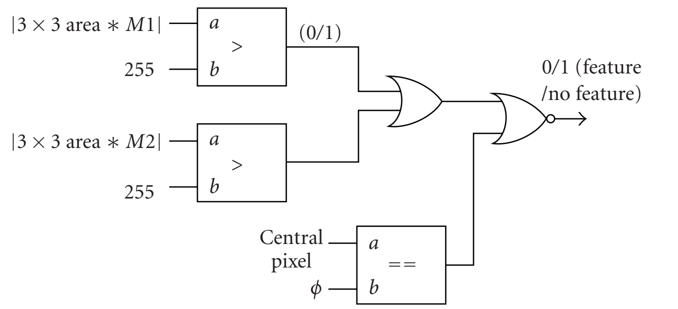


Figure 6: Implementation of the thinning step.

displacement according to the disparity value. In Figure 7 we simply use shift taps and a multiplexer in order to perform the above check.

3.3. System-on-A-Chip and Host Application. The system was modeled using CAD tools like DSPBuilder by Altera and was exported as an HDL library component, ready for integration in a System-on-a-programmable-chip. The embedded system was designed using SOPC Builder which is part of Quartus II, the Altera tool for system integration. The system is controlled by a 32-bit Nios II processor, and includes a DDR2 external memory controller and a USB 2.0 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 Figure 8 as a block diagram.

We implement the system targeting a Cyclone II DSP board featuring a 2C35 FPGA device with total resources 33,000 logic elements and 480,000 bits of on-chip memory. On the host side, the stereo-head is connected to a National Instruments frame-grabber and is controlled by a LabVIEW application. The same application controls the USB 2.0 communication with the reconfigurable hardware part achieving a practical transfer rate greater than 60 Mbps. The host application receives the output from the hardware accelerator and uses feature positions and disparities as discrete landmark measurements. These measurements are used in the update stage of a FastSLAM simultaneous localization and mapping algorithm, running in the host side. This procedure is presented in Section 5.

4. Evaluation of the Hardware System

The more sophisticated stage in the system presented in the previous section is the stereo accelerator stage. It performs a demanding computational task and has a considerable cost in hardware resources. Table 1 shows the typical resources needed for a basic and more advanced implementation of the stereo processor. Increasing the range of disparities increases proportionally the necessary resources. Our DP-based accelerator requires on-chip memory for the storage of minimum cost tag values. Increasing image resolution increases proportionally the memory needed for the storage of tag-values per scanline. The feature detector stages including the corner detectors and left-right consistency check can

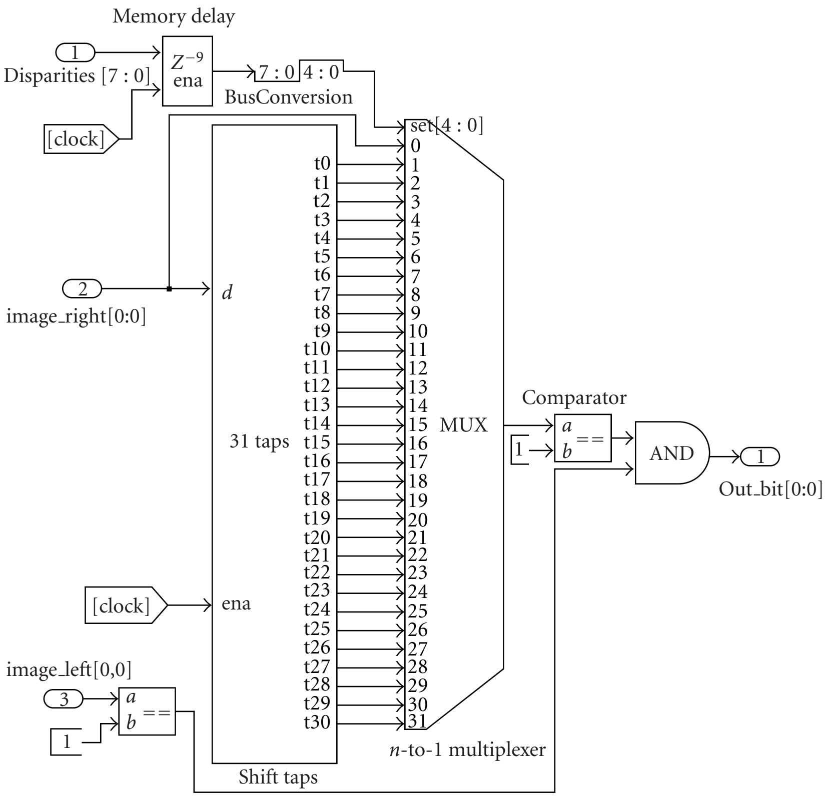


Figure 7: Left-right consistency check.

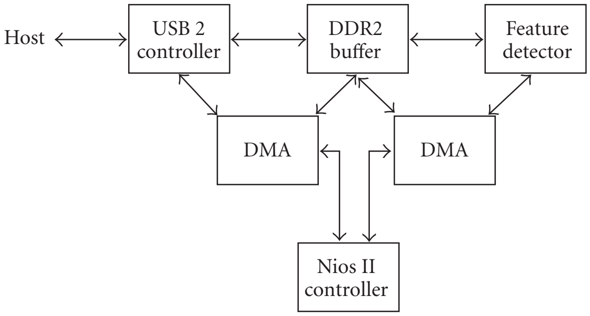


Figure 8: The main components of the system-on-a-programmable chip. The block *Feature detector* represents the system in Figure 1.

be implemented with 4000 additional logic elements. Nios II processor and peripheral controllers require an additional overhead of about 7000 LEs and 160000 bits of embedded memory. The system was implemented targeting a Cyclone II EP2C35 FPGA device.

The implemented stereo accelerator and feature detection system is able to process 49 levels of disparity and produce 640 × 480 8-bit depth maps and point features at clock rate. The higher possible frequency for our present implementation is 50 MHz. This timing restriction is caused by feedback loops, as for example in the cost-matrix computation stage and can potentially be resolved using appropriate hardware optimization. A pair of 640 × 480 images is processed at 12.28 ms, which is equivalent to 81 frames/s or 25 Mpixels/s. The reported throughput is high and suitable for demanding real-time applications, like navigation or environmental mapping. Table 2 presents timing requirements and nominal frame rates for the total system. In practice, apart from the stereo accelerator and feature detector throughput, the frame rate depends also on the camera type and other system parts. For comparison, the stereo matching procedure described in Section 3.1 has been implemented in software and was found to process a stereo pair with resolution 640 × 480 in about 20 seconds.

The stereo processor is the heart of the system and its performance was at first evaluated using reference stereo images. In addition to the Tsukuba University stereo pairs, the “cones,” “sawtooth,” “books,” “arts,” and “bowling” images from the public Middleburry data-set [10] were used for this evaluation. Figure 9 shows a series of ground

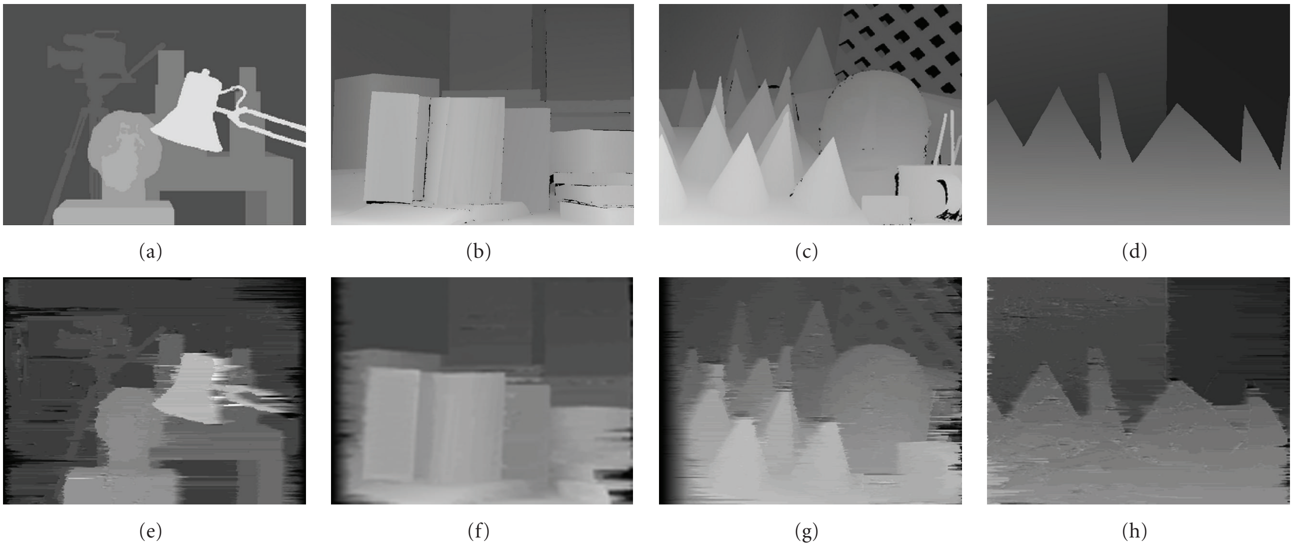


Figure 9: True disparities (upper row) and depth maps produced by the reconfigurable stereo processor (lower row), using the reference stereo set from the Middlebury data-base [10]. From left to right: "Tsukuba," "books," "cones," and "sawtooth" stereo images.

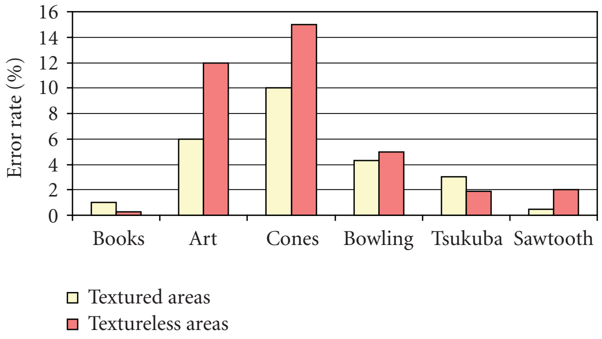


Figure 10: Error rate in terms of bad matches, produced by the reconfigurable stereo processor, using the Middlebury data set.

Table 1: Resource utilization for system implementation.

	Image resolution	Logic elements	Memory bits
Stereo processor/33 disparity levels	320 × 240	10000	74048
49 disparity levels	640 × 480	14700	173632
65 disparity levels	640 × 480	21500	270240
Feature detector	640 × 480	4,000	10,000
Nios II processor + Cashe	—	2,000	120,000
USB 2.0 controller + EP buffer	—	2520	80,000
Other controllers	—	4000	30,000
Total resources 49 disparity levels	640 × 480	27220	413632

truths in comparison with the corresponding disparity maps computed by the reconfigurable processor, based on dynamic programming. For quantitative quality assessment, the error metric proposed by Scharstein and Szeliski [12] was used. This measure is based on true disparities and computes the error rate as the percentage of pixels that differ from ground truth by more than one disparity pixel. Figure 10 shows the percentage of bad matches for the stereo-sets used in

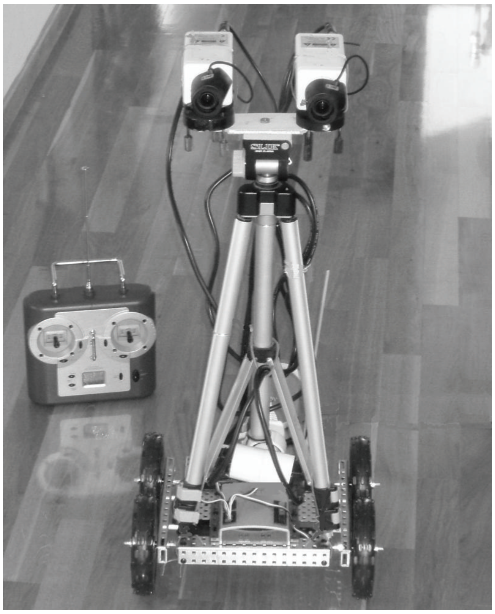


Figure 11: Mobile platform with stereo-head.

Table 2: Processing speed for the feature detection system.

Image resolution	Maximum achieved frequency (MHz)	Maximum throughput (Mpps)	Frame rate (fps)
640 × 480	50	25	81

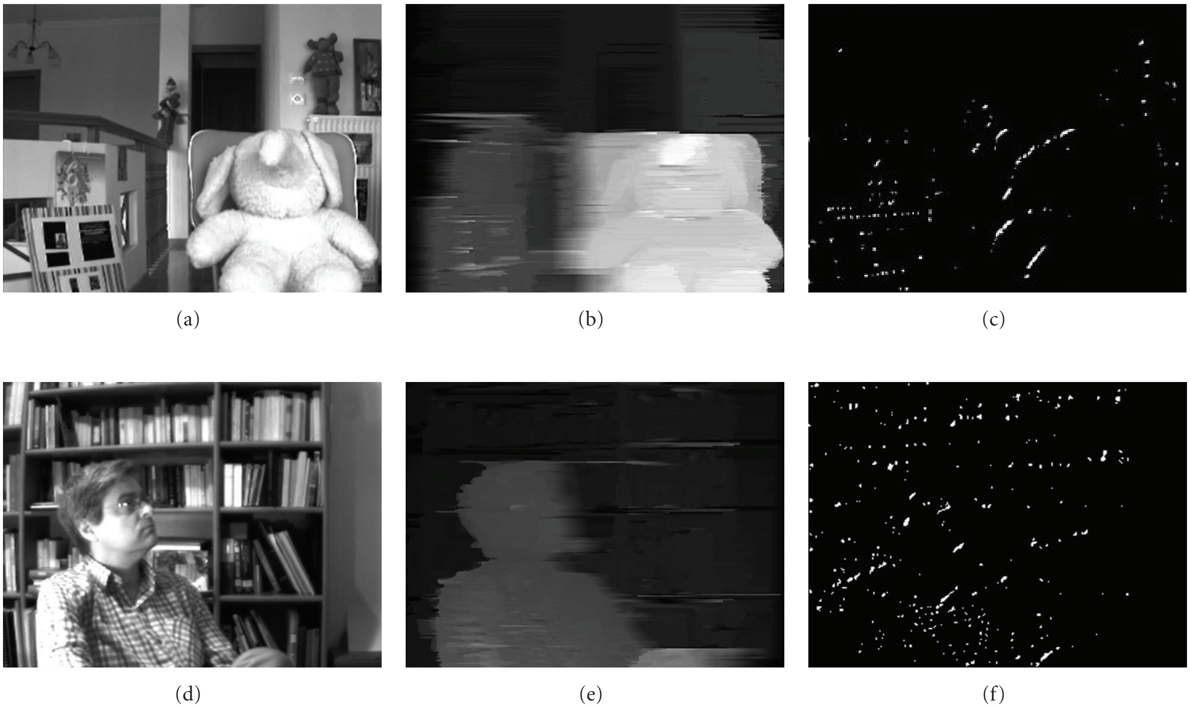


Figure 12: (a,d) Left image of a stereo pair, (b,e) the disparity map produced by the reconfigurable stereo detector, and (c,f) corners prior to thinning.

this evaluation. The error rate in textured and textureless regions is examined separately. Textureless regions are image areas where the average horizontal gradient is below a given threshold. Occlusion boundaries are excluded from the above evaluation. As shown in Figure 10, bad matches are in average lower than 10%. This result confirms quantitatively that the proposed stereo processor produces reliable depth maps independently of image content. It also confirms that the proposed global method implemented in hardware is generally better than common local correlation techniques, like normalized cross correlation or SAD, although the later are easier to fit in specific hardware architectures. An assessment of different stereo-matching techniques can be found in [12].

The stereo processor was also evaluated in terms of depth maps acquired from familiar real scenes. Although this is a subjective measure, it can be a complementary indication of the quality of the proposed system. The stereo head shown in Figure 11 was used to capture and process real scenes. Typical results are shown in Figure 12. Left captured images are followed by depth maps produced by the proposed stereo processor. As shown in Figure 12, depth maps produced by the hardware stereo system are smooth and accurate in the most part of the image. Although some streaking effect is blurring object boundaries, the overall subjective result is satisfactory. Figure 12 also presents the corner points produced by the feature detector described above, prior to thinning. Results of the last two steps (thinning and consistency checking) are pixel-size and cannot be easily displayed.

5. 3D Mapping Experiments

5.1. The Localization and Mapping Procedure. A calibrated stereo head is adapted to a simple mobile vehicle, shown in Figure 11, which can be guided indoors using remote control. In this experimental setup, the mobile platform is tethered and transmits images to a desktop computer. The computer interfaces with the FPGA board through USB 2.0 high speed port. A pair of gray-scale 8-bit video sequences is captured at each step and dense depth disparities are produced along with consistent features in real-time using the system-on-a-chip presented above. The result is transmitted back in the form of pixel stream to the host computer via the USB 2.0 interface. The output pixel array contains zeros, where no feature is found, and the corresponding disparity value in every place where a point feature has been established. In the host part a measurement vector is attributed to every feature, giving the disparity and the feature position in left image coordinates at each time step t

$$z_t = [d, u_{\mathrm{R}},\ v_{\mathrm{R}}\ ]^T, \tag{3}$$

where u_{R}, v_{R} are the horizontal and vertical coordinates of a right image feature and $d = u_{\mathrm{L}} - u_{\mathrm{R}}$.

This measurement set is used as the observation part in a real-time simultaneous localization and mapping experiment. The vehicle is moving on a plane and the vision system reconstructs a complex map based on point features corresponding to 3D landmarks in the world frame.

In order to estimate vehicle motion, our algorithm uses odometry data for translational speed while it derives

Table 3: Performance comparison between hardware stereo matching systems.

Reference	Method of correspondence	Area utilization/memory bits	Image resolution/max disparity	Performance	Quality assessment (average bad matches)	Technology
Present work	Dynamic programming/maximum likelihood	27220 logic elements/413632	640 × 480/49 pixels	25 Mpps/81 fps	<10%	Altera Cyclone II FPGA board + Nios II controller
Diaz et al. (2007) [17]	Phase-based	13048 slices/1308672	1280 × 960/29 pixels	65 Mpps/52 fps	—	Custom FPGA, Xilinx Virtex-II
Ambrosch et al. (2009) [18]	Correlation-SAD	106658 logic elements/425984	330 × 375/120 pixels	136 fps	~38%	FPGA, Altera Stratix EP2S130
Darabiha et al. (2003) [19]	Local weighted phase correlation	~67000 4-input LUT/800000	360 × 276/20 pixels	2.8 Mpps/30 fps	<10%	Custom FPGA board Xilinx Virtex 2000
Liang et al. (2009) [20]	Tile-based belief propagation	2.5 Mgates total	640 × 480/64 pixels	8.2 Mpps/27 fps	—	ASIC
Niitsuma and Maruyama (2004) [21]	Correlation-SAD	31000 slices/405504	640 × 480/27 pixels	9.2 Mpps/30 fps	—	Custom FPGA Xilinx Virtex-II
Kalomiros and Lygouras (2008) [22]	Correlation-SAD	15000 logic elements/196000	320 × 240/32 pixels	25 Mpps/325 fps	~26%	FPGA, Altera Cyclone II
Wang et al. (2006) [23]	Dynamic programming	—	640 × 480/48 pixels	1 Mpps/3 fps	<10%	Graphics processing unit

rotational speed from visual data. This combination is proved to be robust. We find that measuring rotation by tracking image intensity features between frames can be very exact, while estimating translation using image data can be prone to error. This is true especially in the regime of high rotational speed, where image content tends to change rapidly.

Assuming a stereo system with parallel optical axes and a pinhole camera model, our nonlinear measurement model $\hat{z}_t = h(s_t, \hat{\theta})$ can be written in terms of an observed landmark's world coordinates $\hat{\theta} = (x_i,\ y_i,\ z_i)$ and camera position and rotation $s_t = (x_C,\ y_C,\ \psi)$

$$z_t = \begin{bmatrix} u_{\mathrm{L}} - u_{\mathrm{R}} \\ u_{\mathrm{R}} \\ v_{\mathrm{R}} \end{bmatrix} = \begin{bmatrix} \dfrac{fb}{(x_i - x_C)\cos\psi + (y_i - y_C)\sin\psi} \\[2ex] u_0 + \dfrac{f[(x_i - x_C)\sin\psi - (y_i - y_C)\cos\psi - b/2]}{(x_i - x_C)\cos\psi + (y_i - y_C)\sin\psi} \\[2ex] v_0 + \dfrac{f(z_i - z_C)}{(x_i - x_C)\cos\psi + (y_i - y_C)\sin\psi} \end{bmatrix}. \tag{4}$$

In the above equations, f is the camera focal length, b is the baseline of the stereo head, and $(u_0,\ v_0)$ is the central image pixel. The reference frame of the stereo head is shown in Figure 13.

FastSLAM algorithm proposed by Montemerlo and colleagues [24] is adapted to the case of our stereo-assisted point feature observations captured by the hardware system. This method is a Bayesian algorithm for the estimation of the robot's path and environmental map based on landmark observations. It approaches the localization and mapping problem by maintaining low-dimensional Bayesian filters instead of the multidimensional state vector and covariance matrix of the majority of SLAM approaches. Low-dimensionality reduces computational complexity and it is a prerequisite for real-time implementation of localization and mapping in the case of large 3D maps that contain thousands or millions of landmarks. FastSLAM factorizes a posterior over maps and robot paths

$$p(s^t, \Theta \mid z^t, u^t, n^t) = p(s^t \mid z^t, u^t, n^t) \prod_{n=1}^{N} p(\theta_n \mid s^t, z^t, u^t, n^t), \tag{5}$$

where s^t is the robot's path until time t, which is the set of all positions $\{s_t\}$. Θ is the map which consists of the mean position and the covariance of all landmarks. z^t is the set of observed features $\{z_t\}$ from time 0 until time t, where the measurement z_t at each time step is given by (3). n^t is the set of correspondences between observed features z_t and stored landmarks θ_t in the map. The history of control parameters is represented by u^t. Superscripts denote the whole set of data until time t, while subscripts denote data values at a given time step.

The above factorization, first developed by Murphy [25], is derived from the fact that given the vehicle's path, the position of every landmark can be estimated as an independent quantity. In FastSLAM, the distribution for the robot posterior $p(s^t \mid n^t, z^t, u^t)$ in (5) is estimated using a particle filter, and the remaining N conditional landmark posteriors $p(\theta_n \mid s^t, z^t, u^t, n^t)$ are estimated using Extended Kalman Filters (EKF). Each particle m in the particle filter contains a robot path s^t and N independent EKFs, each one tracking a single landmark position [26].

In order to estimate the map feature positions and the vehicle's path according to the posterior (5) we implement the steps summarized below.

(1) Obtain the controls u_t for the last step and sample a new vehicle pose for each particle. This sampling step is performed using the vehicle's motion model and represents the particle filter's proposal distribution. In our implementation, vehicle controls u_t represent translational speed $\Delta s/\Delta t$ which is derived from odometry and rotational speed $\Delta\psi/\Delta t$ which is derived by minimizing absolute intensity differences between successive video frames.

(2) A new set of measurements is captured according to (3) and the resulting point features are associated to previous measurements in the map. Data association n_t is chosen by calculating the minimum Mahalanobis distance between the measured and stored landmarks

$$\hat{n}_t = \arg\min_{n_t}\left\{(z_t - \hat{z}_{n,t})^T Z_{n,t}^{-1}(z_t - \hat{z}_{n,t})\right\}, \tag{6}$$

where z_t is the current measurement and $\hat{z}_{n,t}$ is the measurement prediction of the nth landmark stored in the map Θ. The prediction of measurements for stored landmarks is performed by the measurement model (4). $Z_{n,t}$ is the measurement innovation covariance for the nth landmark at time step t, given in the context of the EKF by

$$Z_{n,t} = H_{\theta_{n_t}} P_{n_t,t-1}^{[m]} H_{\theta_{n_t}}^T + R_t, \tag{7}$$

where R_t is the measurement error covariance matrix and $P_{n_t,t-1}^{[m]}$ is the error covariance for the nth landmark in the mth particle. $H_{\theta_{n_t}}$is the observation matrix, calculated as the Jacobian of the measurement model, with respect to landmark coordinates

$$\hat{z}_t = h\left(s_t^{[m]}, \hat{\theta}_{n_t,t-1}\right), \tag{8}$$

$$H_{\theta_{n_t}} = \nabla_{\theta_{n_t}} h(s_t, \theta_{n_t}). \tag{9}$$

Equation (8) gives the measurement prediction according to the robot path and map in the mth particle and is given analytically by (4). $H_{\theta_{n_t}}$ is calculated from (4) differentiating with respect to x_i, y_i, z_i:

$$\begin{aligned} H &= \begin{bmatrix} \dfrac{\partial(u_L - u_R)}{\partial x_i} & \dfrac{\partial(u_L - u_R)}{\partial y_i} & \dfrac{\partial(u_L - u_R)}{\partial z_i} \\ \dfrac{\partial u_R}{\partial x_i} & \dfrac{\partial u_R}{\partial y_i} & \dfrac{\partial u_R}{\partial z_i} \\ \dfrac{\partial v_R}{\partial x_i} & \dfrac{\partial v_R}{\partial y_i} & \dfrac{\partial v_R}{\partial z_i} \end{bmatrix} \\ &= \begin{bmatrix} \dfrac{-fb\cos\psi}{X^2} & \dfrac{-fb\sin\psi}{X^2} & 0 \\ \dfrac{f\left(y_i - y_C + (b/2)\cos\psi\right)}{X^2} & \dfrac{-f\left(x_i - x_C - (b/2)\sin\psi\right)}{X^2} & 0 \\ \dfrac{-f(z_i - z_C)\cos\psi}{X^2} & \dfrac{-f(z_i - z_C)\sin\psi}{X^2} & \dfrac{f}{X} \end{bmatrix}, \end{aligned} \tag{10}$$

where $X = (x_i - x_C)\cos\psi + (y_i - y_C)\sin\psi$.

(3) There follows the update stage of the landmarks' EKFs, according to the update equations of the EKF theory [27]. Updating the filter produces the posterior at time t from the one at time $t-1$. Features with null correspondences are initialized in terms of their mean and covariance and are inserted in each particle as new landmarks.

Steps (2) and (3) are repeated for all features captured in a frame at each time step t.

(4) After processing all measurements in a given stereo pair, each particle m in the current generation of particles is weighted according to the probability of the current observation, conditioned on the robot path. In our implementation, this is equal to the probability M of all measurements at time t, and can be defined in terms of log-likelihoods as

$$\begin{aligned} \log M &= \sum_i \log p\left(z_{t,i} \mid s^{t,[m]}, z^{t-1}, u^t, n^t\right) \\ &= \sum_i -\frac{1}{2}\min\left\{(z_{t,i} - \hat{z}_{n_t,t})^T Z_{n,t}^{-1}(z_{t,i} - \hat{z}_{n,t})\right\}. \end{aligned} \tag{11}$$

The sum is taken over all observations in a given stereo pair. Each measurement probability is equal to the probability of the innovation $z_{t,i} - \hat{z}_{n_t,t}$, given by the Mahalanobis distance for the correct correspondence n_t. The weight update for particle m is calculated by

$$\hat{w}_{m,t} = \frac{\exp(\log M)}{\sum_{i=1}^{N}\exp(\log M)} w_{m,t-1}, \tag{12}$$

where N is the number of observations in the stereo pair. Particle weights are finally normalized

$$w_{m,t} = \frac{\hat{w}_{m,t}}{\sum_{m=1}^{M}\hat{w}_{m,t}}. \tag{13}$$

(5) Particles are redistributed in order to correct the difference between the proposal distribution and the desired posterior (5). The resampling procedure in our implementation is performed according to the technique "Select with replacement" [28].

The above basic operations are repeated at each time step. At the final time step, the vehicle path and the map are

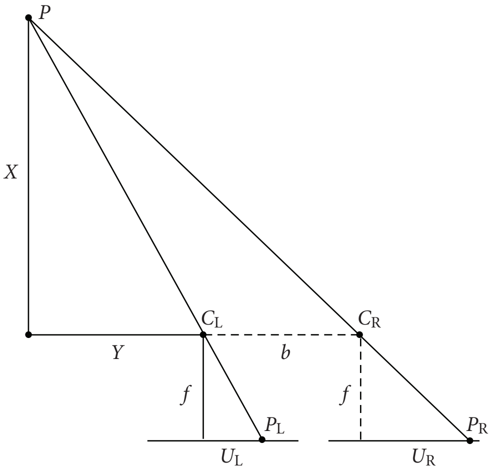


Figure 13: Top-down view of two identical parallel cameras with focal length f at distance b to each other.

estimated according to the particle possessing the maximum weight. The main steps in the above algorithm are derived from Montemerlo's original implementation of FastSLAM 1.0 [24, 26] while (10) represents the observation model developed in the present paper. Equations (11), (12), and (13) represent our preferred method for weight update in terms of the "select with replacement" resampling algorithm.

5.2. Experimental Results. Localization and mapping experiments were carried out using the observation model described in the previous sections. The vehicle is guided indoors by a human operator, recording features in a 5.5 × 4.2 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. A nonzero rotational speed is maintained so that the ability of the system to tolerate errors in odometry is tested. Multiple rounds can test the behavior of the system at loop closure.

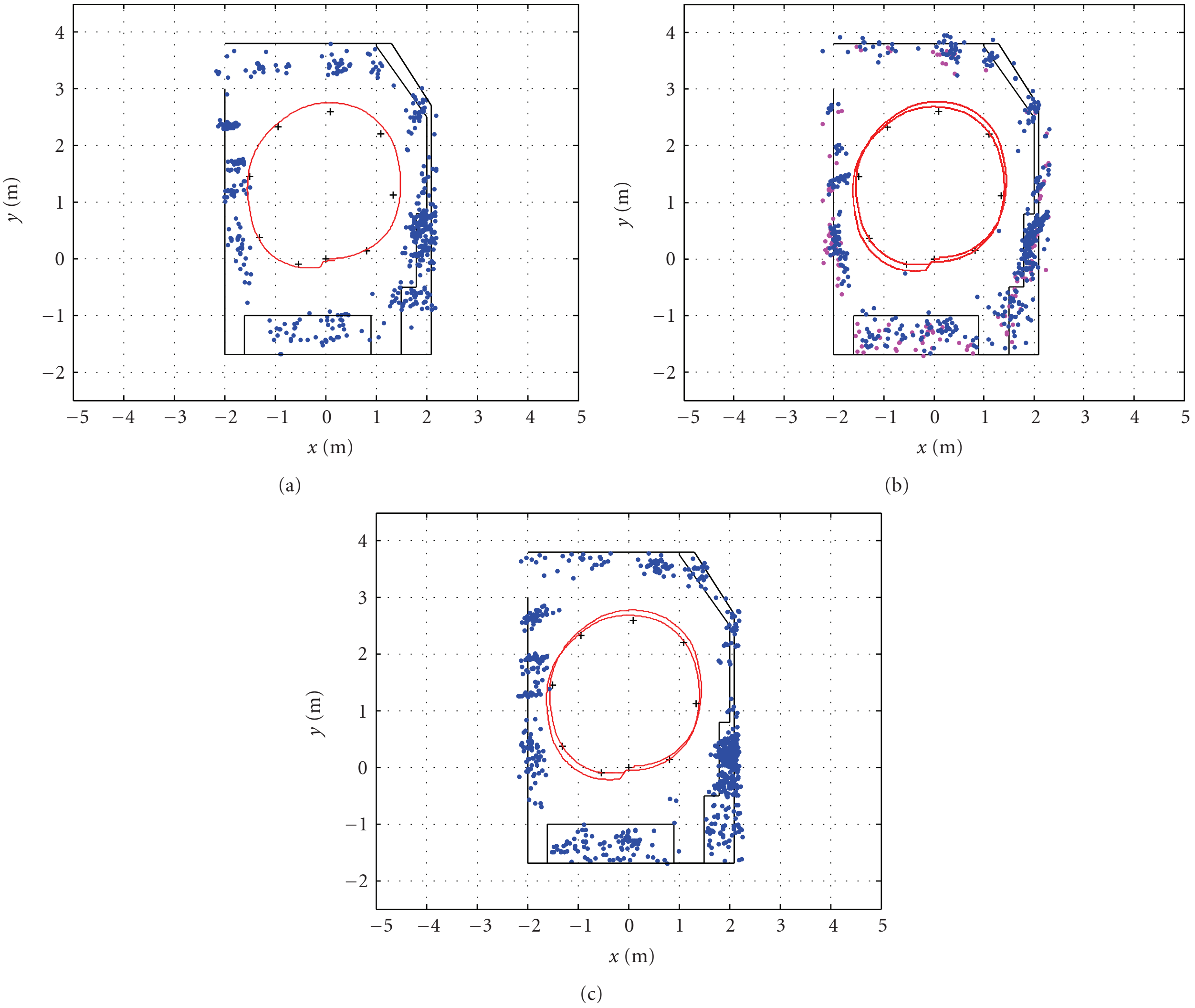
Figure 14(a) presents a result of mapping the room and simultaneously localizing the vehicle, employing just one particle in our FastSLAM implementation. In this figure, map features are shown with blue dots at the room's perimeter, while the red line records the vehicle path as estimated by the algorithm. For comparison the true path is presented with crosses, as it was recorded on the floor by a marker adapted on the vehicle. Perimetric lines show the true position of the walls and furniture. The metric map presented in Figure 14(a) is referenced to the initial position of the vehicle at point (0,0). Figure 14(b) shows the result after two rounds of the vehicle and utilizing five particles in the estimation procedure. New landmarks observed during the second round are shown in this figure with purple dots. Figure 14(c) shows the result of the same experiment utilizing ten particles. Figure 15 shows a 3D map of the room obtained using two independent particles.

As is shown in these figures, acceptable maps and robot paths can be estimated even with just one particle, while ten particles make a slight improvement over five particles.

In spite of the low dimensionality of the measurement vector, features produced by the stereo sensor are tracked efficiently from frame to frame and are updated appropriately, keeping the total number of features in the map relatively low. At loop closure, revisited features are recognized and updated. As shown in Figure 14(b), relatively few point features are acknowledged as new features in the second round, since most of them have already been observed during the first round of the vehicle.

The accuracy of the above procedure mainly depends on the accuracy of odometry measurements and the accuracy of disparity measurements. The Bayesian algorithm can partially correct odometry errors by tracking feature points and updating the filter. However, systematic errors in pixel disparity, especially at occlusion boundaries and at regions without texture, can introduce erroneous observations into the map. Although FastSLAM tends to remove wrong associations from the map, by propagating high probability particles in the resampling step and attenuating improbable ones, this does not work in case the error distribution is not Gaussian. The proposed semiglobal stereo-matching algorithm implemented in hardware is proved to produce quite accurate measurements, as compared to less sophisticated techniques tested for the same purpose. For example, an implementation of the Sum of Absolute Differences (SAD) in reconfigurable hardware [22] was also tested, in the place of the stereo processor stage of our sensor. It is found that the map produced by the SAD stereo processor includes a considerable number of outliers; it presents fewer consistent features and a more dispersed cloud of points. The result of this experiment is shown for comparison in Figure 16.

Next, the real-time processing ability of the proposed system is explored. Figure 17 shows processing rate (steps or iterations per second) as a function of the number of particles in the FastSLAM filter, for the first one hundred steps. Adequate maps were produced even with a single particle. Using more particles has as a result the most probable map and robot path, since each particle contains its own vehicle path and a map conditioned on this path. In each processing step the vehicle is promoted forward, a new stereo pair is captured, the dense depth map is produced, and a set of point features is extracted using the proposed



(a)

(b)

(c)

Figure 14: Projection of the estimated map (blue dots) on the *xy* plane. Red line is the estimated vehicle path. Black lines represent the ground truth of the objects in the map and crosses are the true vehicle positions during the first round. (a): result after one round, with one particle in the filter. (b): five particles, two rounds. Purple dots represent new landmarks observed during the second round. (c): ten particles.

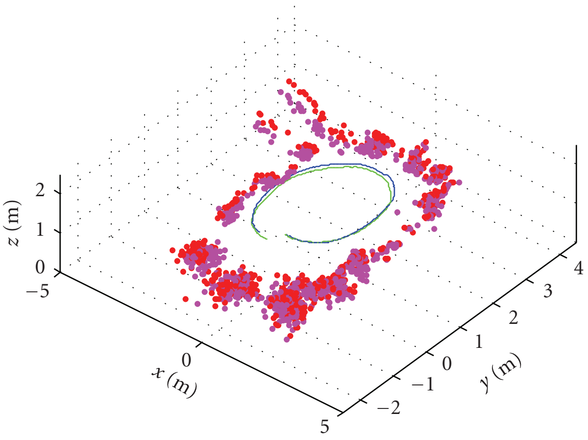


Figure 15: 3D graph of the estimated map. Two independent particles are projected in the same map.

reconfigurable system. The measurement set is input to the FastSLAM filter, which uses the measurements in order to update the particle weights and the positions of the observed landmarks. As it is shown in Figure 17, the proposed system can process up to 12 full iterations per second with just one particle and slows down to approximately two iterations per second for one hundred particles. Using more than ten particles, however, deteriorates the real-time efficiency of the algorithm, especially when thousands of features are gathered in the map, after several minutes of exploration. Certainly, the notion of real-time processing here depends on the required speed of the vehicle. Our experiments were conducted with the controller preserving constant speed 0.15 m/s on the direction of motion. Using this standard, the system can process at least four iterations per second without loosing track of the observed landmarks in subsequent camera frames. The ability to track and correlate features

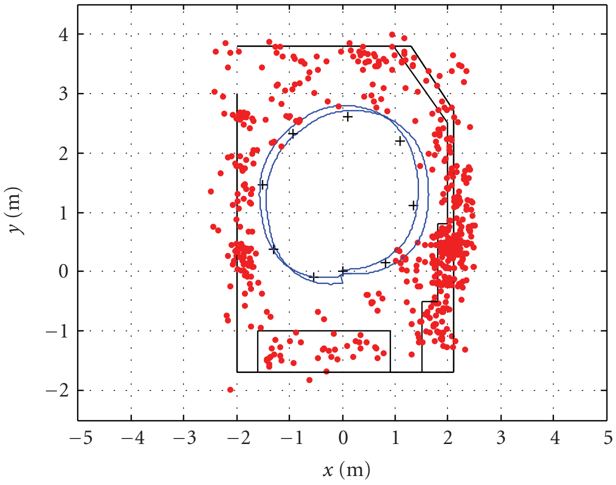


Figure 16: Result of a localization and mapping experiment in the same premises as in Figure 14 using a SAD reconfigurable processor in the place of the proposed dynamic programming stereo system.

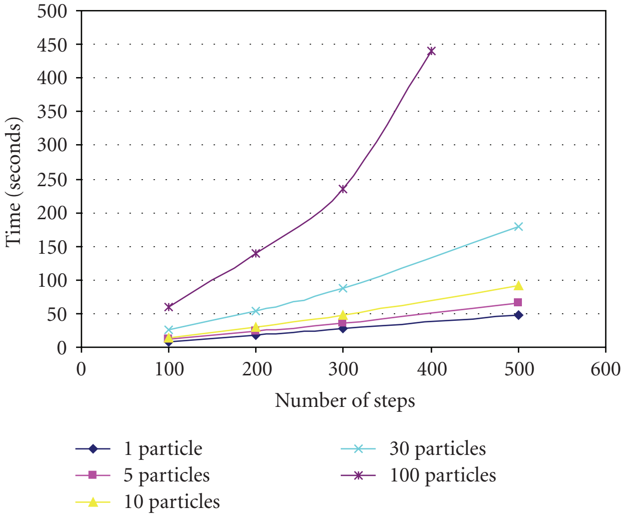


Figure 18: Processing time with increasing number of steps, for different number of particles in the filter.

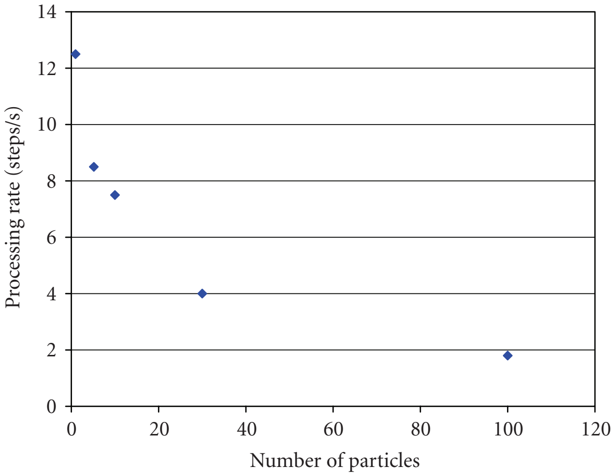


Figure 17: Processing rate in the first one hundred steps of the vehicle, as a function of the number of particles in the FastSLAM filter.

between frames is imperative for the correct function of the proposed system.

On the other hand, the processing rate is a function of the number of landmarks in the map. The map is augmented as the vehicle moves forward gathering new measurements. Figure 18 presents the evolution of processing time as a function of the number of processing steps, for different number of particles in the filter. For more than ten particles, the processing time increases superlinearly with the number of steps, hence the processing rate decreases. Choosing between five and ten particles, we find that the proposed system is well within limits of real-time execution of the FastSLAM algorithm and also keeps its processing rate almost constant. We also note that utilizing more than ten particles in the filter does not result in perceptible improvements in the estimation of the robot path and environmental map.

Processing rate limitations in the proposed system are due to the computational load in the software part of the system, which implements the FastSLAM algorithm on a desktop computer with a quad core CPU running at 2.4 GHz. The configurable part of the system can respond well within the rate allowed by the software part and does not add to the computational load. This is characteristic of the significance of specific hardware in a robotic localization and mapping procedure. As it is noted by other researchers, the typical time required for each iteration can measure up to several seconds and most of this time is usually spent on the feature extraction stage [29, 30]. Attributing feature extraction to dedicated hardware can ease the computational load and allow real-time response. This is particularly true for stereo-assisted feature extraction.

The present experiments were conducted indoors with a stereo baseline adjusted for optimum operation at several meters. A proper autonomous vehicle and stereo head are under development in order to conduct tests in larger or outdoors environments.

6. Comparison with Other Systems

A meaningful comparison can be made with other stereo sensors designed in reconfigurable hardware. A number of real-time stereo systems were presented in the literature in recent years. Several systems were built using FPGA devices, like the Xilinx Virtex series or the Altera Cyclone and Stratix families. Most such systems are based on area correlation methods, using techniques like SAD [31, 32] and rank transforms [33]. Ambrosch and co-authors have recently implemented SAD-based stereo matching, rank and cencus transform using FPGAs [18, 34]. They use 106658 logic elements out of a Stratix EP2S130 device, and their quality assessment yields 61.12% and 79.85% correct matches for their SAD and census transform, respectively, while frame rates range to several hundreds.

Niitsuma and Maruyama [21] described a real-time system for the detection of moving objects, based on SAD stereo matching. Their implementation uses a Virtex-II FPGA and processes 30 frames per second with resolution 640 × 480 pixels.

Darabiha et al. [19] and Masrani and MacLean [35] implemented phase-based stereo in FPGAs using the equivalent of 70000 logic elements and about 800 Kbits of on-chip memory. Their system is built with Stratix S80 devices and supports a maximum disparity range of 128 pixels. Diaz et al. implemented a fine-grain pipeline structure for phase-based disparity estimations. Their implementation uses a Xilinx Virtex-II FPGA device and is able to produce one pixel of disparity per system clock cycle [17]. A similar phase-based implementation is also reported in [36].

Global optimization VLSI architectures are just beginning to emerge. Recently, hardware-efficient algorithms for realizing belief propagation were proposed. Belief propagation (BP) provides global optimality and good matching performance when applied to disparity estimation. However, it has great computational complexity and requires huge memory and bandwidth. Cheng et al. [37] propose a tile-based BP algorithm in order to overcome the memory and bandwidth bottlenecks. In a related paper Liang et al. [20] implemented the tile-based approach as well as a method for parallel message construction on a dedicated VLSI chip. The chip consists of 2.5 M gates and processes images in VGA resolution with 64 pixels maximum disparities at 27 frames per second.

The problem of mapping in hardware belief propagation for global stereo estimation is also addressed by Park and Jeong [38]. They implement a BP algorithm using a Xilinx XC2vp100 FPGA chip. They process 16 disparity levels and image resolution 256 × 240 at 25 frames per second, using 9564 slices (approximately 20000 LE) and more than 10 Mbits of memory partitioned in on-chip RAM blocks and external SRAM.

Many researchers investigate the performance of commodity graphics cards in accelerating stereo algorithms. Most implementations investigate local correlation techniques and achieve real-time performance [13, 39]. Global algorithms produce good results but are computationally intensive and not very appropriate for real-time speed. Some attempts to parallelize dynamic programming on Graphics Processing Units (GPUs) achieve good balance between quality and speed and can be a good choice for PC-oriented applications [23, 40]. However, using dedicated hardware, like in our proposed system, is faster and more suitable for autonomous systems equipped with vision sensors.

Table 3 presents a performance comparison between recently published stereo matching hardware systems. Most systems in this table are implemented using FPGAs, while one system is prototyped as Application Specific Integrated Circuit (ASIC) and one is a GPU implementation, shown here for comparison. Area utilization is given in various units, as published. We note that one logic element is approximately equivalent to one 4-input Lookup Table (LUT) and one slice is approximately equivalent to two 4-input LUTs.

In Tables 1 and 2 of the present paper resource usage and processing speed are presented for our reconfigurable stereo system designed to assist detection of point features for robot mapping. These data compare favorably with implementations reported in the aforementioned literature. The strong point of the architecture proposed in this paper is its full parallelism, resulting in one output disparity pixel at every clock cycle. Parallelism of the stereo algorithm is achieved by means of a state machine allowing cost computation of D states along the diagonal of the cost plane in one clock cycle. The system processes 25 Mpps or 81 frames per second in full VGA resolution (640 × 480). Resolving timing restrictions can result in even higher performance.

As shown in the previous section, the features computed by the presented hardware can be used successfully in the measurement part of a real-time simultaneous localization and mapping experiment. Landmark-based SLAM with stereo vision has also been explored by a number of other researchers. Most of them exploit the properties of discrete multidimensional image features, like SIFT features [41] and Shi-Tomasi features [42]. Others track 3D line segments [8]. They all use sparse disparity values computed at the location of interest by some optimized software technique. In our work, we exploit stereo acceleration provided by reconfigurable hardware and produce high accuracy dense depth maps using a global method of correspondence based on dynamic programming. Dense and accurate depth information makes it possible to extract low-dimensional point features, using an integrated multistage system on a chip. Processing simple features makes it possible to manage large maps and robot paths in real-time by applying a suitable FastSLAM algorithm, as presented in the previous sections.

7. Conclusions

In this paper, an integrated system-on-a-chip is presented for the detection of stereo-assisted point features, suitable for robotic mapping and localization. The system is based on a stereo accelerator implementing a parallelized semiglobal dynamic programming technique. Corners are extracted from the stereo pair using horizontal and vertical edge detectors and a hardware-friendly thinning technique. A final hardware stage implements left-right consistency check based on the disparity values extracted from the dense depth map at corner pixels. The system is implemented as a system-on-a-chip with a Nios II processor for data control and a USB 2.0 high speed interface for communication with a host computer. A stereo head is adapted on a mobile platform and the reconfigurable feature detector is used to implement the measurement part in a simultaneous localization and mapping experiment. A state-of-the-art FastSLAM algorithm on the host part is used to test mapping accuracy and real-time performance of the system. The experimental results illustrate that the system is able to perform in real-time and produce robot paths and maps of indoors environments. Future work will further explore the use of dense depth to real-time vision-based robotic navigation.

References

[1] W. van der Mark and D. M. 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, Boston, Mass, USA, 2002.

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

[4] M. W. M. Gamini Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 229–241, 2001.

[5] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: real-time single camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[6] J. Shi and C. Tomasi, "Good features to track," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '94)*, pp. 593–600, June 1994.

[7] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the 7th IEEE International Conference on Computer Vision (ICCV '99)*, vol. 2, pp. 1150–1157, Kerkyra, Greece, September 1999.

[8] M. N. Dailey and M. Parnichkun, "Landmark-based simultaneous localization and mapping with stereo vision," in *Proceedings of the Asian Conference on Industrial Automation and Robotics*, 2005.

[9] C. J. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of the 4th Alvey Vision Conference*, pp. 147–151, Manchester, UK, 1988.

[10] May 2010, http://cat.middlebury.edu/stereo/data.html.

[11] M. Z. Brown, D. Burschka, and G. D. Hager, "Advances in computational stereo," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 8, pp. 993–1008, 2003.

[12] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International Journal of Computer Vision*, vol. 47, no. 1–3, pp. 7–42, 2002.

[13] M. Gong, R. Yang, L. Wang, and M. Gong, "A performance study on different cost aggregation approaches used in real-time stereo matching," *International Journal of Computer Vision*, vol. 75, no. 2, pp. 283–296, 2007.

[14] J. Kalomiros and J. Lygouras, "A reconfigurable architecture for stereo-assisted detection of point-features for robot mapping," in *Proceedings of International Conference on ReConFigurable Computing and FPGAs (ReConFig '09)*, pp. 404–409, Cancun, Mexico, December 2009.

[15] I. J. Cox, S. L. 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.

[16] Y. Ohta and T. Kanade, "Stereo by intra- and inter-scanline search using dynamic programming," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 7, no. 2, pp. 139–154, 1985.

[17] J. Díaz, E. Ros, R. Carrillo, and A. Prieto, "Real-time system for high-image resolution disparity estimation," *IEEE Transactions on Image Processing*, vol. 16, no. 1, pp. 280–285, 2007.

[18] K. Ambrosch, W. Kubinger, M. Humenberger, and A. Steininger, "Flexible hardware-based stereo matching," *EURASIP Journal on Embedded Systems*, vol. 2008, Article ID 386059, 12 pages, 2008.

[19] A. Darabiha, J. Rose, and W. J. MacLean, "Video-rate stereo depth measurement on programmable hardware," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '03)*, vol. 1, pp. 203–210, Madison, Wis, USA, June 2003.

[20] C. K. Liang, C. C. Cheng, Y. C. Lai, L. G. Chen, and H. H. Chen, "Hardware-efficient belief propagation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '09)*, pp. 80–87, Miami, Fla, USA, June 2009.

[21] H. Niitsuma and T. Maruyama, *Real-Time Detection of Moving Objects*, vol. 3203 of *Lecture Notes in Computer Science*, Springer, New York, NY, USA, 2004.

[22] J. A. Kalomiros and J. Lygouras, "Hardware implementation of a stereo co-processor in a medium-scale field programmable gate array," *IET Computers and Digital Techniques*, vol. 2, no. 5, pp. 336–346, 2008.

[23] L. Wang, M. Liao, M. Gong, R. Yang, and D. Nister, "High-quality real-time stereo using adaptive cost aggregation and dynamic programming," in *Proceedings of the 3rd International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT '06)*, pp. 798–805, Washington DC, USA, June 2006.

[24] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: a factored solution to the simultaneous localization and mapping problem," in *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI '02), the 14th Innovative Applications of Artificial Intelligence Conference (IAAI '02)*, pp. 593–598, Edmonton, Canada, July-August 2002.

[25] K. Murphy, "Bayesian map learning in dynamic environments," in *Advances in Neural Information Processing Systems (NIPS)*, MIT Press, Cambridge, Mass, USA, 1999.

[26] M. Montemerlo and S. Thrun, "FastSLAM, a scalable method for the simultaneous localization and mapping problem in robotics," in *Springer Tracts in Advanced Robotics*, B. Sicilinao, O. Khatib, and F. Groen, Eds., vol. 27, Springer, Berlin, Germany, 2006.

[27] M. S. Grewal and A. P. Angus, *Kalman Filtering, Theory and Practice*, Prentice-Hall, Upper Saddle River, NJ, USA, 1993.

[28] J. Carpenter, P. Clifford, and P. Fearnhead, "Improved particle filter for nonlinear problems," *IEE Proceedings: Radar, Sonar and Navigation*, vol. 146, no. 1, pp. 2–7, 1999.

[29] R. Sim, P. Elinas, M. Griffin, and J. Little, "Vision-based SLAM using the Rao-Blackwellized particle filter," in *Proceedings of IJCAI Workshop on Reasoning with Uncertainty in Robotics*, pp. 9–16, Edinburgh, UK, 2005.

[30] M. H. Li, B. R. Hong, R. H. Luo, and Z. H. Wei, "Novel method for mobile robot simultaneous localization and mapping," *Journal of Zhejiang University: Science*, vol. 7, no. 6, pp. 937–944, 2006.

[31] Y. Miyajima and T. Maruyama, "A real-time stereo vision system with FPGA," in *Field Programmable Logic and Applications*, vol. 2778 of *Lecture Notes in Computer Science*, pp. 448–457, Springer, Berlin, Germany, 2003.

[32] M. Hariyama, Y. Kobayashi, H. Sasaki, and M. Kameyama, "FPGA implementation of a stereo matching processor based on window-parallel-and-pixel-parallel architecture," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E88-A, no. 12, pp. 3516–3521, 2005.

[33] J. Woodfill and B. von Herzen, “Real-time stereo vision on the PARTS reconfigurable computer,” in *Proceedings of the 5th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, pp. 201–210, April 1997.

[34] K. Ambrosch, M. Humenberger, W. Kubinger, and A. Steininger, “SAD-based stereo matching using FPGAs,” in *Embedded Computer Vision, Part II*, B. Kisacanin, S. Bhattacharyya, and S. Chai, Eds., Springer, London, UK, 2009.

[35] D. K. Masrani and W. J. MacLean, “A real-time large disparity range stereo-system using FPGAs,” in *Proceedings of the 4th IEEE International Conference on Computer Vision Systems (ICVS ’06)*, pp. 13–20, January 2006.

[36] J. Díaz, E. Ros, S. Mota, E. M. Ortigosa, and B. del Pino, “High performance stereo computation architecture,” in *Proceedings of the International Conference on Field Programmable Logic and Applications (FPL ’05)*, pp. 463–468, Tampere, Finland, August 2005.

[37] C. C. Cheng, C. K. Liang, Y. C. Lai, H. H. Chen, and L. G. Chen, “Analysis of belief propagation for hardware realization,” in *Proceedings of the IEEE Workshop on Signal Processing Systems (SiPS ’08)*, pp. 152–157, Washington, DC, USA, October 2008.

[38] S. Park and H. Jeong, “High-speed parallel very large scale integration architecture for global stereo matching,” *Journal of Electronic Imaging*, vol. 17, no. 1, Article ID 010501, 2008.

[39] F. Tombari, S. Mattoccia, L. D. Stefano, and E. Addimanda, “Classification and evaluation of cost aggregation methods for stereo correspondence,” in *Proceedings of the 26th IEEE Conference on Computer Vision and Pattern Recognition (CVPR ’08)*, pp. 1–8, June 2008.

[40] J. Congote, J. Barandiaran, I. Barandiaran, and O. Ruiz, “Realtime dense stereo matching with dynamic programming in CUDA,” in *Proceedings of the 19th Spanish Congress of Graphical Informatics (CEIG ’09)*, pp. 231–234, San Sebastian, Spain, September 2009.

[41] R. Sim, P. Elinas, M. Griffin, A. Shyr, and J. J. Little, “Design and analysis of a framework for real-time vision-based SLAM using Rao-Blackwellised particle filters,” in *Proceedings of the 3rd Canadian Conference on Computer and Robot Vision (CRV ’06)*, Quebec, Canada, June 2006.

[42] A. Cumani, S. Denasi, A. Guiducci, and G. Quaglia, “Robot localization and mapping with stereo vision,” *WSEAS Transactions on Circuits and Systems*, vol. 3, no. 10, pp. 2116–2121, 2004.