# Reconfigurable Computing Architecture for Accurate Disparity Map Calculation in Real-Time Stereo Vision

**P. Zicari, H. Lam, and A. George**
NSF Center for High-Performance Reconfigurable Computing (CHREC)
Dept. of Electrical and Computer Engineering, University of Florida
Gainesville FL, USA 32611

**Abstract** - *This paper presents a novel hardware architecture using FPGA-based reconfigurable computing (RC) for accurate calculation of dense disparity maps in real-time, stereo-vision systems. Recent stereo-vision hardware solutions have proposed local-area approaches. Although parallelism can be easily exploited using local methods by replicating the window-based image elaborations, accuracy is limited because the disparity result is optimized by locally searching for the minimum value of a cost function. Global methods improve the quality of the stereo-vision disparity maps at the expense of increasing computational complexity, thus making real-time application not viable for conventional computing. This problem becomes even more evident when stereo vision is a single step integrated into a more complete image elaboration flow, where the depth maps are used for further detection, recognition, stereo reconstruction, or 3D enhancement processing. Our approach exploits a parallel and fully pipelined architecture to implement a global method for the calculation of dense disparity maps based on the dynamic programming optimization of the Hamming distance of the Census-transform cost function. The resulting stereo-vision core produces results that are significantly more accurate than existing hardware solutions using FPGAs that are based upon local approaches. The design was implemented and evaluated on an Altera Stratix-III E260 FPGA in a GiDEL PROCStar-III board. Tests were performed on 640×480 stereo images, with a Census transform window size = 3, correlation window size = 5, and disparity ranges of 30 and 50. Our hardware architecture achieved a speedup of about 319 and 512 respectively for the two disparity ranges, when compared to an optimized C++ implementation executed on a 2.26 GHz Xeon E5520 core. High accuracy in the output disparity map, together with high performance in terms of frames per second, make the proposed architecture an ideal solution for 3D robot-assisted medical systems, tracking, and autonomous navigation systems, where accuracy and speed constraints are very stringent.*

**Keywords:** Real-time stereo vision; dynamic programming; FPGA; reconfigurable computing

## 1 Introduction

Accurate and real-time 3D reconstruction from stereo vision is one of the most important research topics for improving computer-vision systems today and is essential in applications such as robotics, automated medical systems, video surveillance, object recognition, people tracking, obstacle detection, and autonomous navigation. Depth information in stereo vision is determined by processing the left and right images acquired by a stereo camera, which is composed of two calibrated cameras aligned at a baseline distance *b*. The stereo-matching problem consists of searching the correspondent points in the left and right images. A preprocessing operation, called rectification, simplifies the matching computation by aligning the acquired left and right stereo images so that the search can be executed on the horizontal scan lines. Fig. 1 shows an example of stereo matching in which the horizontal displacement *D* of the matched points, called disparity, is used to calculate the distance *z* of the real point in the scene from the stereo camera using Eq. 1, where *f* is the focal length.

$$z = \frac{b \times f}{D} \qquad (1)$$

Stereo-vision algorithms are widely recognized as extremely compute-expensive in the image-processing domain. Moreover, this complexity drastically increases when improving the quality of the depth maps. In the last several years, novel algorithmic improvements through software implementations has greatly extended the list of new entries in the Middlebury stereo evaluation table [1], which rates the different matching methods with respect to the accuracy of the disparity results over a set of benchmark stereo images: Tsukuba, Venus, Teddy, and Cones. These images, provided with ground-truth disparity maps, can be used as a common reference for fair comparisons. Although these algorithms can be implemented in a straightforward manner in software, their execution on CPUs is sequential, which does not always provide a viable solution for real-time applications with stringent performance requirements.

This work was supported in part by: the European Commission and the Calabria Region of Italy, through the European Social Fund, Regional Operational Program 2007/2013, Priority IV – Human Capital; and the I/UCRC Program of the National Science Foundation under Grant Nos. EEC-0642422 and IIP-1161022.

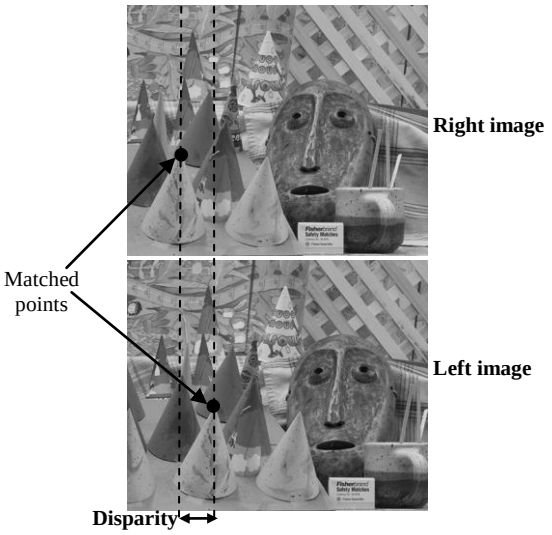


Fig. 1. The disparity of the matched points.

This paper presents a novel, FPGA-based hardware architecture for accurate calculation of dense disparity maps in real-time, stereo-vision systems. The elaboration flow design includes matching cost computation, cost aggregation, disparity calculation, and consistency-check validation. Unlike most recent stereo-vision hardware implementations, which are based solely upon local-area approaches [2-7], the proposed solution exploits a global method for the calculation of dense disparity maps based on the dynamic-programming optimization of the Hamming distance of the Census transform cost function. As a result, the parallel and fully pipelined architecture significantly improves the accuracy as compared to recent hardware solutions. Moreover, when compared to the more accurate stereo-matching approaches based on dynamic-programming methods running on CPUs and GPUs [9-14], the proposed stereo-vision architecture outperforms them by one to two orders of magnitude in speed. An implementation of the proposed architecture, running on an Altera Stratix-III EPSE260 FPGA, achieved speedups of about 319 and 512 for disparity ranges of 30 and 50, respectively, when compared to an optimized C++ baseline executed on a 2.26 GHz Xeon E5520 core.

The remainder of the paper is organized as follows. Section II presents the related works. Section III furnishes a detailed description of the approach and the design of the hardware stereo architecture. Section IV reports the experimental results and the comparisons against the most recent hardware and software solutions found in the literature. Finally, conclusions and directions for future research are given in Section V.

## 2 Related Works

In order to retrieve depth information, stereo-matching methods search for correspondences in a pair of right and left images acquired by a stereo camera. A detailed survey of the most recent methods for searching correspondences is presented in [15], where a classification in terms of matching cost, aggregation, and optimization functions is provided. In the plethora of methods present in the literature, local window-based algorithms have been preferred in recent hardware implementations [2-7] for the parallel execution of the repetitive operations on multiple windows. One of the first FPGA implementations of stereo systems is the reconfigurable PARTS engine [7], consisting of 16 Xilinx 4025 FPGAs, and 16 one-megabyte SRAMs. A frame rate of 42 frames per second (fps) was achieved when the Census algorithm was executed on 320×240 stereo images with a disparity range of 24 pixels. In [3], the Local Weighted Phase Correlation matching algorithm was implemented on four Virtex2000 FPGAs, where 256×360 disparity maps were calculated at a rate of 30 fps, with a disparity range of 20 pixels. In [5], a novel stereo-matching algorithm based on the Census transform of gradient images was proposed. The implemented version used a Stratix EP1S60 FPGA and a maximum of 60 fps was reached on a disparity range of 60 pixels. The FPGA stereo-vision system in [2] implemented the Census-based disparity matching over 640×480 stereo images. The disparity was computed by selecting the minimum winning cost over a disparity range of 64 pixels by using 11×11 Census transform windows and 15×15 correlation windows for the pixel aggregation. A frame rate of 230 fps was achieved when implemented on a Xilinx Xc4vlx200 FPGA. In [6], a fast and low-cost, stereo-vision system based upon SAD (Sum of Absolute Differences) was presented for real-time applications. A novel injective consistency check improves the efficiency by greatly reducing area usage with respect to the more common cross-checking methods which require the computation of both left and right disparity maps. In [8], a very different approach based on fuzzy logic was used to reach high frame rates and high accuracy in an Altera Stratix EP1S60 FPGA. Comparison of our proposed architecture with these hardware solutions will be given in Section IV.

Several software implementations of global methods have been presented in the literature, each achieving high-quality results but poor performance. Most global methods search for the optimum disparity distribution, which minimizes a specific global-energy function. Each selected disparity is not the result of a single independent decision as in the local methods, but is the result of a global decision that involves many disparity values considered together. Stereo matching based upon dynamic programming is a well-known class of global methods using scan-line optimization. Different approaches in using dynamic programming in stereo vision have been proposed in the literature [9-14]. The stereo-matching system in [14], rated highly in the Middlebury ranking [1], achieved very high accuracy due to the multi-direction scan-line optimization based on Hirschmüller's semi-global matching method, followed by several refinement steps systematically executed in order to correct the disparity errors in occluded and near-depth discontinuity regions. Unfortunately, the performance (about 10fps) was penalized by the high computational load in its CPU+GPU implementation. In [12], an adaptive aggregation step based

upon the color and proximity weighting of the sum of absolute difference cost function was adopted in conjunction with a dynamic-programming scan-line optimization. Performances of 7.63 and 5.46 fps were achieved when 640×480 images are processed in the disparity ranges of 32 and 48 pixels, respectively, running on a 3GHz PC with an ATI Radeon XL1800 GPU. In [10], a system with a coarse–to-fine refinement approach was implemented on a 2.2 GHz AMD Athlon XP 2800+ CPU, achieving a frame rate of 12.3 fps when processing 640×480 stereo images in a disparity range of 50 pixels. In [13], the GPU-based Orthogonal Reliability-based Dynamic-Programming (GORDP) stereo system used two dynamic-programming passes with a local minimum searching process. Stereo images of 320×240 pixel sizes were processed at 10 fps for disparity ranges of 20 pixels. One of the most accurate dynamic-programming methods for stereo matching was proposed in [9], where a generalized ground control points (GGCP) scheme was introduced together with a two-pass optimization technique for reducing the inter-scan line inconsistency problem. Unfortunately, real-time is far from sustained by their Pentium IV 2.4GHz PC implementation, which calculates the disparity of the Tsukuba, Saw tooth, Venus and Map benchmark stereo images in 4.4, 11.8, 11.1 and 4.9 seconds, respectively. Comparison of our proposed architecture with these software solutions will also be given in Section IV.

# 3 Proposed Approach and Architecture

In order to evaluate the impact of using the dynamic-programming scan-line optimization when applied to the Hamming distance of Census transform cost functions, we performed an analysis over the Middlebury stereo-pair images. These benchmark images provide truth disparity maps as a reference for comparing the algorithmic results. As cited in [15], the percentage of bad pixels $B$ is calculated as in Eq. 2 for the *non-occlusion*, *all* and *discontinuity* regions, where $N$ is the total number of pixels, $D$ and $D_{truth}$ are the computed and the ground truth disparity values, respectively.

$$B = \frac{1}{N} \times \sum_{(x,y) \in \text{Re}\,gion} \left| D(x,y) - D_{truth}(x,y) \right| > 1 \qquad (2)$$

Fig. 2 shows the average percentage of bad pixels over the Tsukuba, Venus, Teddy, and Cones images, for the Census-Hamming, local-area approach and the proposed dynamic-programming approach. The cost function for both approaches uses an aggregation window size of 3×3, a correlation window size of 5×5, and consistency cross check is used to select the valid disparity values. The results show that the dynamic-programming optimization improves the quality of the output maps considerably in the *all* and *non-occluded* regions, while offering similar quality in the discontinuity regions. The average error over the benchmark stereo pairs is reduced by 51.74%, 48.04% and 1.77% for the *non-occlusion*, *all* and *discontinuity* cases, respectively. Note that abrupt changes of disparity inside discontinuity regions are not significantly improved by the applied global optimization method, since it is based on the minimization of an energy function aimed to smooth the depth discontinuity along the scan lines.

Our proposed architecture for dynamic programming in stereo vision is shown in Fig. 3. The datapath structure is fully pipelined and parallelized in order to enable a continuous input flow of left and right pair of pixels and output flow of disparity at each clock cycle. The left and right pixels of the acquired stereo images are serially inputted to the *Matching Cost Function* module for the Census transform and the Hamming distance calculation. The *Dynamic Programming* module searches for the optimum disparity path. Finally, the disparity in output is validated by the *Consistency Cross Check* module, which compares results of the left and right matching processes.

## 3.1 Matching Cost Function Module

The matching cost computation produces the Census transform over $Wc \times Wc$ windows, and then the Hamming distance of the Census vectors over $Wh \times Wh$ aggregation windows. According to [16], each element $CV(x,y,k)$ of the Census vector is the sign bit of the subtraction result between the generic $P(x+i,y+j)$ pixel and the central pixel $P(x,y)$ of the selected $Wc \times Wc$ window calculated as in Eq. 3, with $-(Wc-1)/2 \le i,j \le (Wc-1)/2$ and $k = (y+j-1) \times Wc + x + i$.

$$CV(x,y,k) = sign\,(P(x+i, y+j) - P(x,y)) \qquad (3)$$

The generic aggregated matching cost $C$, with respect to the disparity value $z$ in the disparity range $r$, is calculated as in Eq. 4, by counting all the bit differences between the right $CVR$ and the left $CVL$ Census vectors in the selected $Wh \times Wh$ window. The matching cost based on the Hamming distance of Census transformed images is a non-parametric measure that is insensitive to differences in camera gains and bias [16].

$$C(x,y,z) = \sum_{i=\frac{-(Wh-1)}{2}}^{(Wh-1)/2} \sum_{j=\frac{-(Wh-1)}{2}}^{(Wh-1)/2} \sum_{k=1}^{Wc*Wc} [CVR(x+i, y+j, k) \oplus CVL(x+z+i, y+j, k)] \qquad (4)$$

The *Matching Cost Function* module consists of the *Census Transform* component and the *Hamming Distance* component. The high-level model designs of the two components are shown in Fig. 4 and Fig. 5, respectively. This module for the entire disparity range executes in parallel. Thus, two $C(x, y, 1{:}\ r)$ cost vectors are simultaneously computed for the left and right distinct processing flows at each clock cycle.

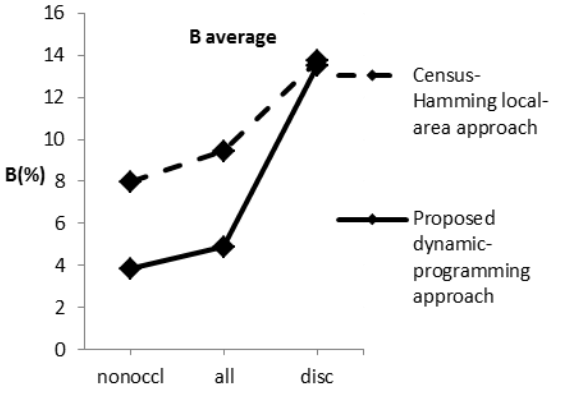


Fig. 2. The average percentage of bad pixels B over the benchmark Venus, Teddy, Cones and Tsukuba stereo images for the Census-Hamming, local-area approach and the proposed dynamic-programming approach.

In order to guarantee the parallel processing of the serially inputted *Nc*×*Nr* left and right images, two *Nc*×(*Wc*-1)+*Wc* pixel buffers (implemented as shift registers) are used. After a latency of *Nc*×(*Wc*-1)+*Wc* clock cycles, an entire *Wc*×*Wc* pixel window is inputted to the *Census Transform* component at each clock cycle. The sign bits outputted from the *n*=*Wc*×*Wc* parallel subtraction circuits of the *Census Transform* are then inputted into the *Census Buffer*, which uses *Nc*×(*Wh*-1)+*r*+*Wh*-1 *n*-bit registers connected as shown in Fig. 5. After *Nc*×(*Wh*-1)+*r*+*Wh*-1 clock cycles from the first Census vector input, one reference window and *r* candidate windows are outputted from each *Census Buffer* at each following clock cycle. The Hamming distance between a reference window and each candidate window is calculated in parallel by 2×*r* *Hamming Distance (HD)* blocks. An *HD* block includes a bank of XOR-gates and a final tree of pipelined adders. Tree adders at the first level have multiple single-bit operands, while adders at the other levels have two operands, with the input precision incremented by one bit at each level.

## 3.2 Dynamic Programming Module

In our design, the *Dynamic Programming* module searches for the minimum cost path on the basis of a scan-line optimization. For each row, *r* different disparity paths are calculated by building the energy function matrix *E* and the matrix *P* of disparity paths. According to [15], each element $E(x,y,z)$ of the energy function matrix is calculated iteratively as shown in Eq. 5, with $1 \le x \le Nc$, $1 \le y \le Nr$ and $0 \le z < r$; $\delta(x,y,z)$, taking into account the depth discontinuity through the constant term $\lambda$ as shown in Eq. 6.

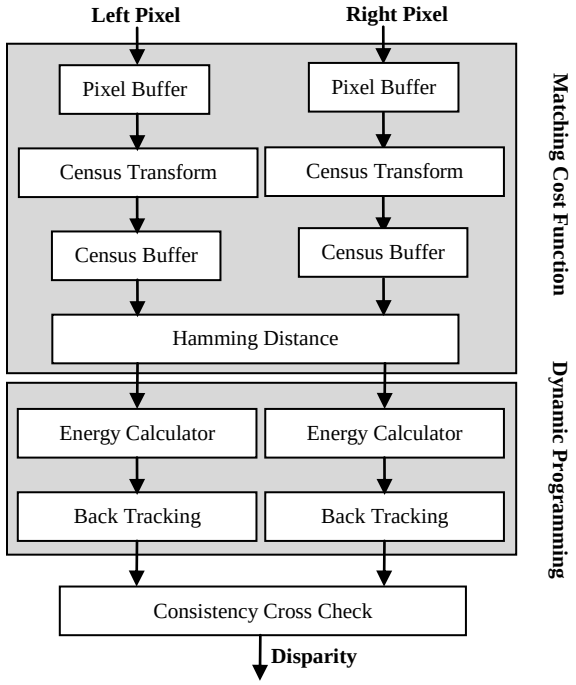


Fig. 3. Proposed Stereo-Vision Architecture.

$$E(x,y,z) = C(x,y,z) + \delta(x,y,z) \tag{5}$$

$$\delta(x,y,z) = \min\{\, C(x-1,y,z-1) + \lambda, C(x-1,y,z), C(x-1,y,z+1) + \lambda \,\} \tag{6}$$

According to [12], each element $P(x,y,z)$ is calculated as in Eq. 7 by selecting the minimum arguments calculated in Eq. 6 with respect to the index *z*.

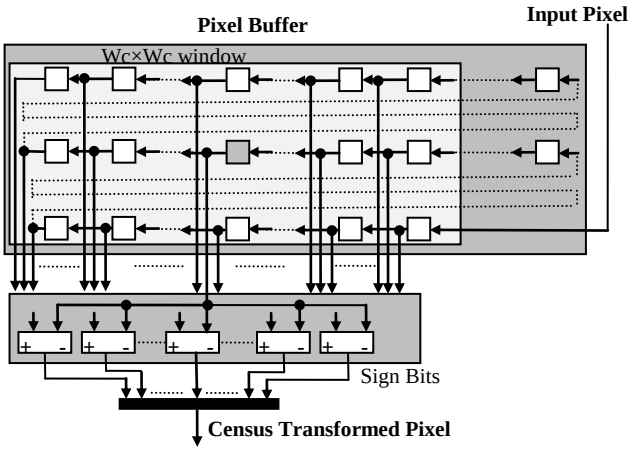


Fig. 4. Census Transform Component.

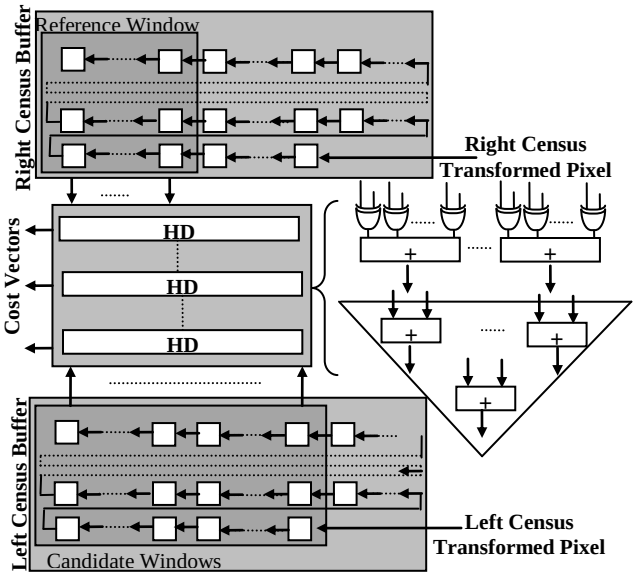


Fig. 5. Hamming Distance Component.

$$P(x,y,z)=\begin{cases} z-1 & \text{when } \delta(x,y,z)=C(x-1,y,z-1)+\lambda \\ z & \text{whe n } \delta(x,y,z)=C(x-1,y,z) \\ z+1 & \text{when } \delta(x,y,z)=C(x-1,y,z+1)+\lambda \end{cases} \quad (7)$$

The matrix $P$ keeps track of all the possible $r$ disparity paths for each row. The disparity map $D$ is calculated as shown in Eq. 8. The last disparity in each scan line is the $z$ position of the minimum energy value, while all the previous disparity values in each path are retrieved by back-tracking through the matrix $P$.

$$D(x,y)=\begin{cases} \arg(\min_{0\le z<r} E(x,y,z)) & \text{when } x = Nc \\ P(x+1,y,D(x+1,y)) & \text{when } 0 < x < Nc \end{cases} \quad (8)$$

The design of the *Dynamic Programming* module is shown in Fig. 6. The optimal disparity path is calculated by iteratively processing the cost vector $c=C(x, y, 0: r\text{-}1)$ and the energy vector $e=E(x, y, 0: r\text{-}1)$ in a row-scanning order. At each clock cycle, a cost vector $c$ is inputted to the *Dynamic Programming* module. The *EF Block* calculates the energy function as in Eq. 5. The *Disparity Path Storage Block* is used to store the $r$ possible disparity paths. The *Min Tree Block* selects the best path which minimizes the energy function of the entire path, furnishing in the output the last disparity value. All of the previous disparity values in the optimum path are retrieved by the back-tracker *BT Block* in an inverted order. The disparity values are then queued in the *Disparity Storage Block* to be outputted in the right order by the forward tracker *FT Block*. As the energy of an entire path represents the energy accumulated in each single step, the energy function block is realized as a bank of $r$ special accumulators as shown in Fig. 6. The aggregated matching costs are accumulated at each clock cycle. To take into account the depth discontinuity, the minimum value among the adjacent energy values is selected and appropriately corrected by the constant $\lambda$.

One of the main disadvantages of the dynamic-programming approach is the considerable amount of resources needed to store all of the possible disparity paths during the scan-line optimization. In fact, the optimum among all the disparity paths can be selected only at the end of the scan line after that the global energy is computed. If the optimization is performed on an entire image row, $r$ paths of the row length need to be saved until the energy function for the entire row is calculated, thus requiring the storage of $Nc \times r$ disparity values. In our design, in order to reduce resource usage, instead of saving the disparity values, 2 bits of information are stored for tracking each $s$=+1, 0, -1 variation step with respect to the previous disparity value in the path, as shown in Eq. 9.

$$s(x,y,z)=\begin{cases} -1 & \text{if } \delta(x,y,z)=C(x-1,y,z-1)+\lambda \\ 0 & \text{if } \delta(x,y,z)=C(x-1,y,z) \\ +1 & \text{if } \delta(x,y,z)=C(x-1,y,z+1)+\lambda \end{cases} \quad (9)$$

The variation step $s$ is calculated by the *Min* block inside the energy accumulator of the *EF Block*. In this way, after the *Min Tree Block* calculates the last disparity value of the path, all of the previous disparity values in the optimum path are back-tracked by the *BT Block*, by appropriately incrementing, disabling, or decrementing a counter register initially loaded with the last disparity value. The *BT Block* uses a multiplexer to select the next disparity step in the optimum path. After back-tracking, the disparity flow is inverted with respect to the left-to-right input order; thus a further step is required. The disparity variation steps of the optimum path are queued in the *Disparity Storage Block*. The *FT Block* then forward-tracks the disparity values starting from the first disparity value of the current row, outputted by the *BT Block* at the end of the back-tracking phase. The counter in the *FT Block* is controlled by the disparity variation step $s$ outputted from the *Disparity Storage Block*. The *Disparity Path Storage Block* and the *Disparity Storage Block* are two shift-register bidirectional buffers storing the disparity steps in a LIFO (Last Input First Output) order. While the former stores all the possible paths into $r$ stack lines, the latter stores only the optimum path in just one stack line. In order to use the same structure for contemporary pushing and pulling the disparity steps of two consecutive paths without stopping the pipelined processing flow, in each stack line the registers are interleaved by multiplexers. The select signals of the multiplexers are used to control the direction of the push-pull operation like a piston moving a cylinder back and forth in an engine. As the result of this design, the amount of resources for storage was reduced by 80% in the *Dynamic Programming* module, thus reducing by 60% the total amount of storage for the entire design.

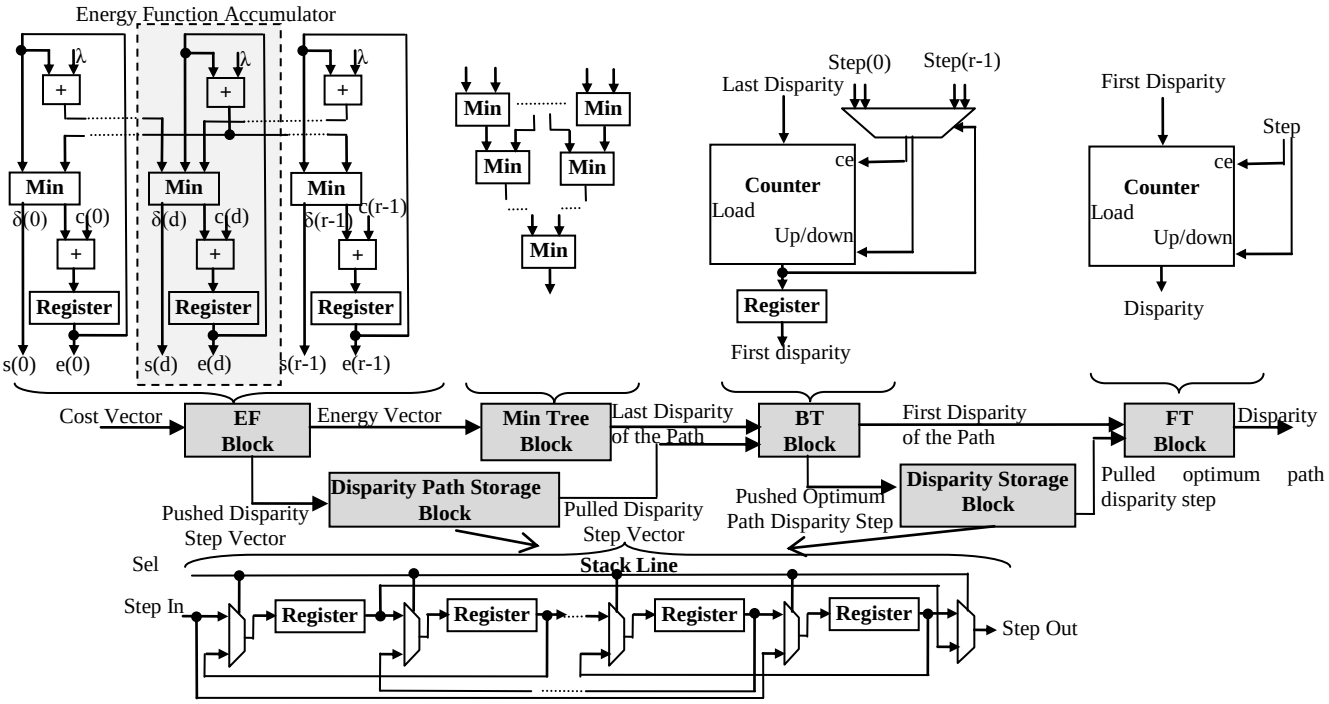


Fig. 6. Dynamic Programming Module.

### 3.3 Consistency Cross Check

Post-processing is adopted in order to reduce the matching errors that could be caused by occlusions and false matching. The cross check validates the consistency of the right and left results. The matching is considered valid by the cross check method when the right *Dr* and the left *Dl* disparity values satisfy the condition shown in Eq. 10. Only if this is the case, the disparity is flagged as a correct result.

$$Dr(x, y) = Dl(x + Dr(x, y), y) \qquad (10)$$

The stereo-vision system executes the right and left matching processes in parallel. Thus, one right and one left disparity value are simultaneously ready at the output from the *Dynamic Programming* module at each clock cycle. As shown in Fig. 7, the disparity values are appropriately buffered in *r* registers, which are left-shifted at each clock cycle. A bank of *XNOR* logic gates inside the *Comparator* block are used to compare each right disparity with its matched left disparity, selected by a multiplexer. An active-high *valid* signal is outputted when the disparity passes the consistency check.

## 4 Experimental Results and Comparisons

In order to support different image-processing requirements and FPGA platforms, the proposed stereo-vision architecture for the disparity-map calculation was designed in VHDL as an IP core that can be parameterized in terms of image size, Census-transform size, aggregation-window size, and disparity range. Two versions of the proposed architecture have been implemented on an Altera Stratix-III E260 FPGA, the results of which are shown in the first row of Table I, calculating the disparity in the 30 and 50 pixel ranges, respectively. The Census transform is executed over 3×3 windows; the Hamming distance is calculated for aggregation windows of size 5×5; the input image size is 640×480 with 8-bit gray level pixels; and the depth discontinuity constant λ is fixed to 7. For the *r*=30 version, the complete circuit occupies 33,881 combinational ALUTs, 949 memory ALUTs, 101,802 dedicated logic registers, 102,288 total registers, and 493,683 total block-memory bits. For the *r*=50 version, it occupies 50,402 combinational ALUTs, 320 memory ALUTs, 157,005 dedicated logic registers, 157,491 total registers, and 505,355 total block-memory bits. The remainder of Table I is used to compare our solution with other works available in the literature. Comparison is made with both hardware solutions and CPU/GPU solutions.

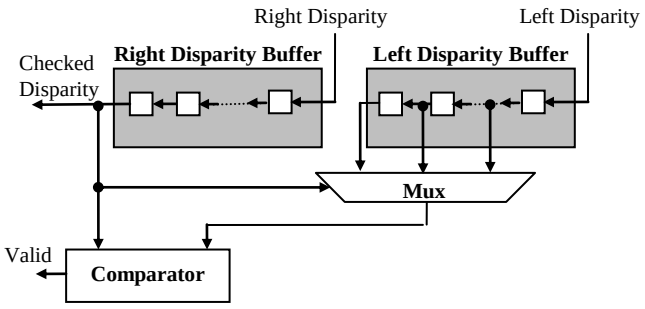


Fig. 7. Consistency Cross Check.

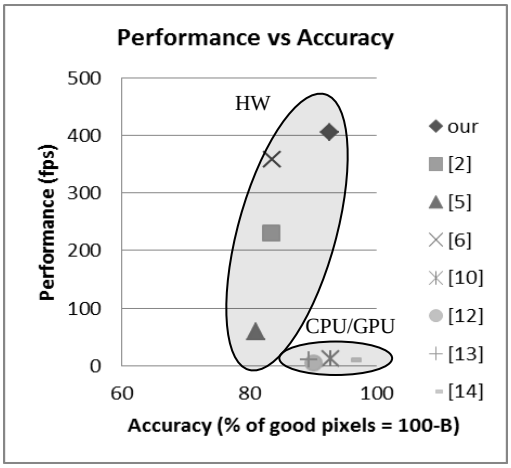


Fig. 8. Performance vs. Accuracy graph.

As shown in Table I, our solution significantly outperformed the cited hardware implementations [2, 5 and 6] in result quality, as indicated by the percentages of bad pixels of the validated disparity maps over Tsukuba, Venus, Teddy, and Cones stereo images in the last column. For the *non-occlusion* cases, the improvement ranges from 32% to 84%. For the *all* cases, the improvement ranges from 43% to 78%. These results are expected because of our use of a global method for the calculation of dense disparity maps based upon the dynamic-programming optimization, as compared to the local-area approaches used by other hardware solutions. For the *discontinuity* cases, except for one result in [6] which is even better than the proposed one, the rest of the results show improvement from 16% to 63%. As noted previously, the improvement in this case is expected to be less because the abrupt changes of disparity inside discontinuity regions are not significantly improved by the applied global-optimization method since it is based on the minimization of an energy function aimed to smooth the depth discontinuity along the scan lines. The design was compiled and downloaded into a GiDEL PROCStar-III board for testing. As the proposed architecture exploits massive parallelism, an increase in the disparity range improves the speedup performance with respect to a software baseline implementation. Performance was measured with speedups of about 319 and 512, for the 30 and 50 disparity ranges respectively, as compared to optimized C++ code executed on a 2.26 GHz Xeon E5520 core. The maximum frame rate of 406 fps was achieved for both IP cores. Compared to the cited software implementations in the literature [10, 12, 13, 14, and 17], our proposed stereo-vision architecture outperformed them by one to two orders of magnitude in terms of frame rate.

TABLE I. COMPARISON OF PERFORMACE RESULTS

| System | Image Size [Pixel] | Device | Disparity Range [Pixel] | Resources | Frame Rate [fps] | B [%] Tsukuba Nocc | Tsukuba All | Tsukuba Disc | Venus Nocc | Venus All | Venus Disc | Teddy Nocc | Teddy All | Teddy Disc | Cones Nocc | Cones All | Cones Disc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Our proposed stereo-vision architecture | 640×480 | FPGA Altera Stratix-III E260 | 30 | 33,881 Comb. ALUTs 949 Mem. ALUTs 101,802 Logic registers 102,288 Tot registers 493,683 Mem. Bits | 406 | 4.39 | 5.21 | 15.54 | 2.41 | 2.96 | 13.81 | 5.13 | 6,54 | 15.76 | 3.30 | 4.75 | 8.63 |
| | | | 50 | 50,402 Comb. ALUTs 320 Mem. ALUTs 157,005 Logic registers 157,491 Tot registers 505,355 Mem. Bits | | | | | | | | | | | | |
| [2] | 640×480 | FPGA Xilinx Xc4vlx200 | 64 | 12 DSP 322 18Kb-BRAM 51,191 Slices | 230 | 9.79 | 11.56 | 20.29 | 3.59 | 5.27 | 36.82 | 12.50 | 21.50 | 30.57 | 7.34 | 17.58 | 21.01 |
| [5] | 750×400 | FPGA Altera Stratix EP1S60 | 60 | 38,944 Logic Elements 557,056 Mem. Bits | 60 | 22.7 | 23.7 | 26.2 | 15.1 | 15.9 | 25.3 | 13.9 | 20.7 | 29.0 | 5.96 | 12.7 | 15.5 |
| [6] | 640×480 | FPGA Xilinx XC4VLX60 | 30 | 63 DSP 64 BRAM 12,974 Slices | 358 | 9.99 | 10.64 | 12.88 | 11.42 | 12.28 | 16.61 | 17.19 | 19.38 | 25.75 | 17.12 | 20.18 | 23.61 |
| | 1280×720 | | | 63 DSP 128 BRAM 15,728 Slices | 97 | | | | | | | | | | | | |
| [10] | 640×480 | CPU | 32 | AMD Athlon XP 2800+ | 15.2 | n.a.* | 4.12 | n.a.* | n.a.* | n.a.* | n.a.* | n.a.* | n.a.* | n.a.* | n.a.* | n.a.* | n.a.* |
| | | | 50 | | 12.3 | | | | | | | | | | | | |
| [12] | 640×480 | CPU + GPU | 32 | 3GHz PC + ATI Radeon XL1800 | 7.63 | 2.05 | 4.22 | 10.6 | 1.92 | 2.98 | 20.3 | 7.23 | 14.4 | 17.6 | 6.41 | 13.7 | 16.5 |
| | | | 48 | | 5.46 | | | | | | | | | | | | |
| [13] | 320×240 | CPU + GPU | 20 | 3 GHz P4 CPU 512 MB RAM ATI Radeon X800 | 10 | 1.34 | 3.36 | 7.10 | 2.73 | 3.81 | 10.1 | 9.03 | 16.8 | 18.4 | 13.1 | 20.1 | 20.1 |
| [14] | 512×384 | CPU + GPU | 60 | Core2, 2.20GHz NVIDIA GeForce GTX 480 | 10 | 1.07 | 1.48 | 5.73 | 0.09 | 0.25 | 1.15 | 4.10 | 6.22 | 10.9 | 2.42 | 7.25 | 6.95 |
| [17] | 640×480 | MIMD many-core architecture | 48 | Tilera TILEPro64 | 71.5 | n.a.* | | | | | | | | | | | |

* n.a. = not available

Compared to a CPU-only implementation [10] of a stereo-matching system based upon a coarse to fine approach of the dynamic programming on a 2.2 GHz AMD Athlon XP 2800+ CPU, our speedup in terms of fps is almost 33 times faster. Compared to an MIMD (Multiple Instruction, Multiple Data) many-core implementation [17], in which the disparity map calculation of the SSD (Sum of Squared Differences) correlation metric was executed on the Tilera TILEPro64 architecture, consisting of 64 32-bit processing cores, our speedup is almost 6 times faster. Compared to works that include CPU and GPU [12, 13, 14], our speedup ranges from about 40 to 74. In summary, a performance vs. accuracy graph of the systems reported in Table I is shown in Fig. 8, where performances are reported in terms of frames per second, and accuracy is the average of the percentage of good pixels calculated as (100-*B*) over the *Nocc*, *All* and *Disc* regions. As shown in this figure, our solution provides the highest speed performances while approaching the level of accuracy of the software CPU/GPU global-methods implementations.

# 5 Conclusions

This paper presents a novel hardware architecture using FPGA-based reconfigurable computing (RC) to calculate dense disparity maps by exploiting a global method based on the dynamic-programming optimization of the Hamming distance of the Census-transform cost function. Recent stereo-vision hardware solutions exploit parallelism by replicating the window-based image elaborations in local-area approaches, but accuracy is limited because the disparity result is optimized by locally searching for the minimum value of a cost function. Our proposed solution, based on global methods and a parallel and fully pipelined architecture, significantly improves the accuracy as compared to recent hardware solutions. Moreover, when compared to the more accurate stereo-matching approaches based on dynamic-programming methods executing on CPUs and GPUs, the proposed stereo-vision architecture outperforms them by one to two orders of magnitude in speed. As a result, our solution provides the best performance while approaching the level of accuracy of the software CPU/GPU global-methods implementations. High accuracy together with high performance make the proposed stereo-vision architecture an ideal solution for 3D robot-assisted medical systems, tracking, and autonomous navigation systems, where accuracy and speed constraints are very stringent.

# 6 References


[1] Middlebury Stereo-Vision page, available at http://vision.middlebury.edu/stereo/

[2] S. Jin, J. Cho, X. D. Pham, K. M. Lee, S. K. Park, M. Kim, and J. W. Jeon, “FPGA Design and Implementation of a Real-time Stereo Vision System”, IEEE Trans. on Circuits and Systems for Video Technology, vol. 20, pp.15 – 26, 2010.

[3] A. Darabiha, J. Rose, and W. J. Maclean, “Video-rate stereo depth measurement on programmable hardware,” in Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit., Madison, WI, vol. 1. Jun. 2003, pp. 203–210.

[4] Masrani, D.K.; MacLean, W.J.; "A Real-Time Large Disparity Range Stereo-System using FPGAs", Computer Vision Systems, 2006 ICVS '06. IEEE International Conference on, pp. 13- 13, 04-07 Jan. 2006.

[5] K. Ambrosch, W. Kubinger, "Accurate hardware-based stereo vision", Computer Vision and Image Understanding, Vol.114, pp.1303-1316, 2010.

[6] P. Zicari, S. Perri, P. Corsonello, G. Cocorullo, “Low-cost FPGA stereo vision system for real-time disparity maps calculation”, Microprocessors and Microsystems, Vol. 36, pp. 281-288, February 2012, Elsevier.

[7] J. Woodfill and B. V. Herzen. “Real-Time Stereo Vision on the PARTS Reconfigurable Computer”, IEEE Symposium on FPGAs for Custom Computing Machines, pp. 201–210, Napa Valley, CA, USA 1997.

[8] C. Georgoulas, I. Andreadis, "A real-time fuzzy hardware structure for disparity map computation", Journal of Real-Time Image Processing, DOI 10.1007/s11554-010-0157-6, 2010.

[9] J. C. Kim, K. M. Lee, B. T. Choi, and S. U. Lee., “A dense stereo matching using two-pass dynamic programming with generalized ground control points”, Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1075–1082, 2005.

[10] S. Forstmann, J. Ohya, Y. Kanou, A. Schmitt, and S. Thuering, “Real-time stereo by using dynamic programming”, Proc. of CVPR Workshop on Real-time 3D Sensors and Their Use, 2004.

[11] M. Gong and Y.-H. Yang, “Near real-time reliable stereo matching using programmable graphics hardware”, Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR), pp. 924–931, 2005.

[12] L. Wang, M. Liao, M. Gong, R. Yang, D. Nister, “High-quality Real-time Stereo using Adaptive Cost Aggregation”, 3D Data Processing, Visualization, and Transmission, Third International Symposium on, pp. 798 – 805, 14-16 June 2006.

[13] M. Gong and Y.H. Yang, “Real-Time Stereo Matching using Orthogonal Reliability-Based Dynamic Programming Algorithm,” IEEE Trans. on Image Processing, Correspondence, Vol. 16, 2007, pp. 879-884.

[14] X. Mei, X. Sun, M. Zhou, S. Jiao, H. Wang and X. Zhang, “On Building an Accurate Stereo Matching System on Graphics Hardware”, GPUCV'11: ICCV Workshop on GPU in Computer Vision Applications, 2011.

[15] D. Scharstein and R. Szeliski. “A taxonomy and evaluation of dense two-frame stereo correspondence algorithms”, International Journal of Computer Vision, 47(1/2/3): 7-42, April-June 2002.

[16] R. Zabih, J. Woodfill. “A non-parametric approach to visual correspondence”, IEEE Transaction on Pattern Analysis and Machine Intelligence, 1996.

[17] Safari, S.; Fijany, A.; Diotalevi, F.; Hosseini, F.; "Highly parallel and fast implementation of stereo vision algorithms on MIMD many-core Tilera architecture", Aerospace Conference, 2012 IEEE, pp.1-11, 3-10 March 2012.