High Performance Stereo System For Dense 3D Reconstruction

Georgios-Tsampikos Michailidis, Renato Pajarola, *Member, IEEE,* and Ioannis Andreadis, *Member, IEEE*

***Abstract*—3D stereo reconstruction, a technique which estimates per-pixel depth in a scene, is still a challenging problem mainly due to some prohibitive factors that limit its performance and computational ability. The aim of this paper is to present a new hardware-efficient disparity map computation, which is based on *disparity space image* (DSI) processing using discrete dynamic systems. The hardware architecture of the proposed system was implemented on a high-end field programmable gate array (FPGA) device, offering real-time 3D reconstruction speeds using a hardware aware architecture based on parallelism and process pipelining. The proposed architecture fulfills the requirements of real-world applications regarding resource usage, frame rates and disparity resolution, while its implementation on an *Altera Stratix IV* family FPGA device can extract disparity maps of up to 1280×1024 pixels with up to 128 disparity levels under real- or near real-time conditions at a clock rate of 168MHz. Qualitative and quantitative results also demonstrate its performance and improvement over previous hardware-related studies, making our approach a suitable candidate for applications where timing and processing constraints are critical.**

***Index Terms*—real-time, disparity space image (DSI), FPGA, image processing, 3D reconstruction, stereo vision.**

I. Introduction

ONE of the key sectors of computer vision is stereoscopy or stereo vision, which refers to the perception of depth. Stereo vision is essential to the generation of a three-dimensional representation of the spatial relationship and shape of objects in our surroundings. Stereo vision systems are widely used in a variety of applications that require knowledge about the depth of objects, such as autonomous vehicles steering, security and military applications. Stereo vision uses a stereo camera setup to acquire the initial stereo images (see also Fig. 1a), and the fundamental problem is to compute the depth of the scene structure from these input images. The general process of extracting the depth information of a scene by a stereo vision system involves the determination of corresponding pixels between the input images, calculation of their disparities, and finally estimation of depth by triangulating their position in space (see also Fig. 1b).

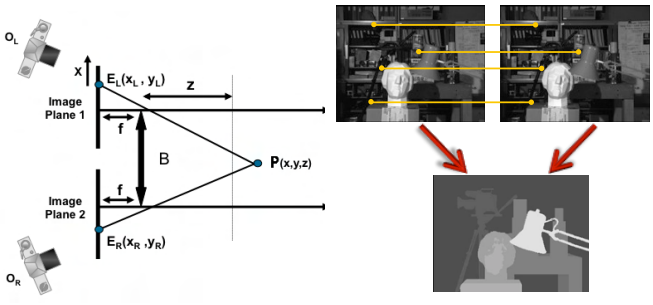
In traditional stereo vision systems, two or more cameras are placed in slightly different horizontal positions, in order to obtain the scene image from different view angles, similar to human binocular vision. For a typical camera stereo setup as in Fig. 1a, the distance to a point $\boldsymbol{P}(x, y, z)$ can be estimated from the two points denoted $\boldsymbol{E}_R$ and $\boldsymbol{E}_L$ (i.e. epipoles) in the two camera image planes. In order to find the corresponding scene point $\boldsymbol{E}_L$ of left image in the right one, we can exploit the epipolar geometry in the stereo image setup. The epipolar geometry defines a plane (i.e. epipolar plane), which is the plane created by a scene point $\boldsymbol{P}(x, y, z)$ and the two optical centers, $\boldsymbol{O}_L$ and $\boldsymbol{O}_R$, and thus given the scene point $\boldsymbol{P}$ observed in the left camera at $\boldsymbol{E}_L(x_L, y_L)$. The corresponding scene point $\boldsymbol{E}_R(x_R, y_R)$ will be constrained to lie on that epipolar plane as well, and thus to be on a corresponding epipolar line of that plane intersecting the right image plane. Therefore, the general 2D correspondence search is reduced to a 1D problem. As a result, the relative depth information from these two images can be obtained, in the form of disparities, by calculating only the horizontal displacement $x_R - x_L = f \cdot B/Z$ of these two corresponding points, which is inversely proportional to the differences in distance to the objects.

In spite of its significance, stereo vision has some serious limitations in real-life applications, due to its inherent ambiguities in the matching process and its increased computational expense. A good survey on the subject can be found in [1], where stereo algorithms are classified into two major categories, global and local methods, with direct implication for hardware design complexity and robustness. The most commonly used global methods attempt to solve the stereo matching problem using *dynamic programming* (DP), *belief propagation* (BP) and *graph cuts* (GC). On the other hand, local correspondence methods are generally categorized into block matching, gradient matching and feature matching algorithms in order to address the correspondence problem [2]. Block matching algorithms are using some statistical methods for determining similarity, like Normalized Cross Correlation (NCC), Sum of Squared Differences (SSD) metric and Sum of Absolute Differences (SAD), gradient matching algorithms seek to determine small local disparities between two images by formulating a differential equation relating motion and image brightness, and feature matching algorithms which attempt to match discrete features among images. Although global methods are very accurate and can produce dense disparity maps, they are computationally more expensive. They often exhibit irregular data access patterns and usually they are unsuitable for real-time applications and hardware implementations. On the other hand, local algorithms yield less accurate disparity maps due to their poor performance on textureless and occluded regions, but they are better suited for real-time stereo matching due to their reduced computational

G.-T. Michailidis and R. Pajarola are with University of Zurich, Switzerland, E-mail: gtmichail@ifi.uzh.ch and pajarola@ifi.uzh.ch.

I. Andreadis is with the Department of Electrical and Computer Engineering, Democritus University of Thrace, Greece, E-mail: iandread@ee.duth.gr.

complexity, their limited data dependencies and their increased suitability for hardware implementations.



(a) (b)

Fig. 1: (a) Stereo vision camera setup, and (b) the determination of corresponding scene points and the creation of the disparity map.

However, the majority of highly-accurate real-time stereo vision systems developed so far have focused on different approaches for solving the stereo matching process, mainly using global metrics and calculations, which increase the computational cost and the processing time. In addition, the majority of current local methods, to which the proposed technique belongs, implement different algorithms that are applied either in the input images, taking advantage of object and intensity differences in order to optimize the matching cost computation/aggregation step and create a plausible disparity map, or after the matching process, in order to optimize/refine the extracted disparity maps and improve the resulting disparity values in a post-processing step [1], [3]. This contribution attempts to diversify from this line and follow a new path for optimizing the stereo matching process, presenting a novel approach for accessing the intermediate level of the disparity estimation process and optimizing the matching costs in a 3D *disparity space image* (DSI) *before* the disparity selection procedure and after the application of the similarity metric, presenting results comparable to that of other methods in shorter time. Although it is local, the big advantage of this approach is the ability to exploit more information, which lie in 3 dimensions and not only in 2, as happens in pre- or post-processing steps of other techniques, as well as the capability to enhance the disparity maps by correcting potential mismatches or correspondences before the final selection of the disparity values by the similarity accumulator, reducing the required processing/refining steps and increasing its performance. It is also independent from the similarity accumulators used in most local algorithms and this advantageous feature makes it suitable for embedding it as an intermediate optimization stage to the majority of local window- or block-based stereo algorithms. It is also worth noticing that the extracted results are produced without using any kind of disparity map post-processing, meaning that any post-processing step can further improve the extracted disparity maps eliminating the remaining false reconstructions.

This efficient behavior is realized by accenting the capabilities of local information in 3D disparity space (i.e. DSI) using a novel local-based *cellular automata* (CA) matching cost optimization technique for eliminating the false reconstructions and optimizing the disparity selection process. Although there are inherent difficulties in refining the DSI using only local information, the proposed technique shows that CA can exploit quite efficiently local information and solve the stereo correspondence problem equally as other approaches.

For the proposed technique, a scalable and highly parallel hardware architecture is also presented, capable of producing dense disparity maps in real-time speeds. The overall disparity estimation process includes pre-processing (i.e. filtering for noise reduction), correspondence matching cost computation, DSI formation, matching cost optimization and decision making. Regarding the scalability issue of the proposed system, it was achieved by using intensively parallelism and pipelining, providing very good results in terms of disparity levels and image sizes. Moreover, the impact of design parameters regarding performance and required hardware resources is explored. Due to its performance, the proposed high frame rate hardware-based system can be applied to many real-time stereo vision applications, including high-speed tracking and mobile robots, object recognition and navigation, biometrics, vision-guided robotics in the automotive industry and many more.

The rest of the paper is organized as follows. Section II discusses related work, Section III gives a short overview of discrete dynamical systems and cellular automata, Section IV presents the proposed solution, while in Section V a detailed description of the hardware architecture and implementation of the proposed system is provided. Section VI presents the experimental results and their evaluation, while conclusions and future work are discussed in Section VII.

II. Related Work

Until recently, the majority of stereo vision methods was mainly restricted to software-based techniques on custom computers. Although these methods are extremely flexible and easy to implement, most of them cannot easily handle the outlined requirements, due to their serial operation. The increased computational complexity and high execution time, caused by numerous iterative calculations, make real-time 3D reconstruction a challenging process. As a result, most of the software-based stereo algorithms require high-end computers and sophisticated code optimization techniques, in order to extract disparity maps in real time. Only in the last decade researchers started to focus on custom hardware-based stereo vision systems implemented on *field programmable gate arrays* (FPGAs) or *application specific integrated circuits* (ASICs).

Some previous works that utilize specialized hardware to reconstruct the 3D scene structure are presented in [4], [5], [6], using Intel MMX processors, graphics processing units (GPUs) and digital signal processors (DSPs), respectively. However, although these approaches solve the 3D reconstruction problem in a computationally efficient manner, they are not suitable for embedded and mobile applications due to their increased power demands. The authors in [7] developed a stereo vision system based on a specialized ASIC processor,

the *DeepSea*, which computes the disparities at 200 frames per second (fps) with a 512×480 input stereo image pair and 52 different disparity levels. Another disparity estimation stereo system, based on an FPGA architecture, is introduced in [8]. This system generates disparity maps of 512×480 stereo images at a frame rate of 30fps, implementing a window-based, scan-line correlation search technique. Another system presented in [9] yields 20fps on 640×480 image sizes, although the performance of the system is limited due to the memory access pattern utilized, which does not provide scalability and performance optimization. In [10] a new system on a Xilinx FPGA platform was developed, which computes trinocular stereopsis using a local-based method. This system runs at approximately 30fps with 640×480 pixel images within a 64 pixel disparity search range. Similar local SAD-based techniques have been implemented in FPGAs in [11] and [12], with different improvements and various results, depending on their configuration and referenced work.

III. Discrete Dynamical Systems and Cellular Automata

A *discrete dynamical system* (DDS) consists of an abstract phase space (or state space) S, whose coordinates describe the state at a set of times T, and a dynamical rule R for evolution $R : S \times T \rightarrow S$ that specifies the future of all state variables, given only the present values of those same state variables. It can be either deterministic, if there is a unique consequence to every state, or stochastic or random, if there is a probability distribution of possible consequences [13].

Cellular automata (CA) are dynamical systems with improved capabilities in massive parallelism, performing complex computations and modeling complicated systems with the help of only local information. CA algorithms are basically computer algorithms that are discrete in space and time and operate on a lattice of sites (in image processing, pixels). CA comprise an array of cells, where each cell can be in one of a finite number of possible states, which is updated synchronously in discrete time steps (clock cycles) according to local transition rules (cell rules). A state of a cell at the next time step is determined by its neighboring cells' current states.

Generally, a CA is characterized by the number of spatial dimensions and a triple $\mathcal{A}=(S, N, \mathcal{F})$, where S is a nonempty set called state set (usually binary), $N \subseteq \mathbb{Z}^2$ is the neighborhood with a width w, where w_j is the width of the j-th side of the array with $j=1, \ldots, n$, and $\mathcal{F}:S^N \rightarrow S$ is the local transition function (rule), where the state of a cell at time $(t+1)$, is computed according to $\mathcal{F}$, given the arguments of $\mathcal{F}$ being the neighbor cells at a given time (t).

In this work, in order to enhance the disparity map with the minimum loss of 3D information, an efficient CA process on a 3D DSI was used, which produces dense disparity maps with minimal false reconstructions. The main idea behind the CA transition rules is that each rule is applied only to specific places in the DSI, in order to find and normalize the unexpected big differences between matching costs in neighboring pixel values, assuming piecewise smooth surfaces

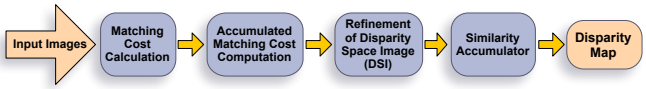


Fig. 2: Block diagram of the proposed system.

and that the spatial smoothness constraint is broken at object (depth) boundaries [1]. In [14] we have already presented a different CA approach, where a semi-dense technique was proposed, using CA in the post-processing stage for disparity map refinement. Although the performance and effectiveness of CA are quite satisfactory, it is inevitable that some wrong replacements occur in the DSI during the iterative process, which is typically applied no more than two or three times in DSI. Thus after exhaustive experiments, the CA rules were modified such that each rule operates as a complemented rule for the previous one and each processing step to be fully pipelined, in order to minimize the inserted errors and increase system's performance.

IV. DSI based 3D Scene Reconstruction

In this section, an efficient DSI processing technique is described, based on a efficient CA structure, in order to calculate an optimized disparity map. An overview of the proposed technique is presented in Fig. 2. After a pre-filtering stage, the stereo correspondence matching cost is calculated for a predefined window size, and the accumulated matching costs are computed by a window-based correlation algorithm. The resulting output comprises the initial DSI, which is then refined by a 2D CA structure, in order to improve the effectiveness of the similarity accumulator. Finally, a similarity accumulator selects the best match for each pixel and assigns to it the proper disparity value. All these stages are fully implemented in hardware on a FPGA device (see Section V).

A. Pre-Filtering

Since in many practical cases the initial intensity values of stereo input images are unreliable, a Laplacian prefilter is applied first on the initial frames for intensity normalization. Then, a weighted mean filter is used to reduce the wrong pixel matches in the stereo correspondence stage. This filter can be described by the following equation:

$$f'(x,y) = 1/4(f(x-1,y) + f(x+1,y)) + 1/2 f(x,y) \quad (1)$$

where f is the original and f' the filtered image. Of course, a two-dimensional filter produces better results, but also increases the computational cost.

B. DSI Calculation and Processing

1) DSI Calculation: The most important and time consuming task for stereo matching algorithms is the identification of corresponding pixels. Although the stereo matching problem is one of the most investigated topics in computer vision, there are still several unpredictable factors that affect the final results, such as noise from light variations, textureless image regions, depth discontinuities at object boundaries and partially occluded areas. These factors can cause lack of information

in disparity calculation and make it impossible to synthesize the correct view.

In order to solve the stereo correspondence problem and compute the disparity map, we have to determine which pixels on the left and right images map to the same point in space. Taking into account speed issues and considering the hardware complexity, the SAD window-based correlation algorithm was used for the disparity map generation, although other metrics, like NCC and adaptive support-weight, could perform better. Assuming that the two input images are already rectified [1], in order to find the corresponding points in the stereo image pair, a block matching method, based on computing the sum of absolute differences, was used:

$$SAD(i,j,d) = \sum_{\mu=-w}^{w} \sum_{v=-w}^{w} |I_l(i+\mu, j+v) - I_r(i+\mu, j+v-d)| \quad (2)$$

where I_l and I_r denote the left and right image pixel values, d is the disparity range, w determines the window size and i, j are the center pixel coordinates (rows, columns) of the window for which the SAD is computed. In our case, w was selected to be 2 for comparison reasons, although the proposed technique can take this value as an input parameter.

The DSI is defined as a 3D space of (x, y, d), where the first two coordinates represent the dimensions of the input images (width and length), while the third one refers to the matching cost for each disparity value that belongs to a predefined disparity range $d_{\max}$. Given two images, the value of the DSI is given by:

$$DSI(x,y,d) = |P_L(x,y) - P_R(x+d,y)|$$
$$where \;\; 0 \le (x+d) \le W \;\; and \;\; d_{min} \le d \le d_{\max} \quad (3)$$

]Each element of DSI is a confidence measure and represents the cost (likelihood) of the correspondence between $P_L(x, y)$ and $P_R(x+d, y)$ of the same epipolar line, where P_L and P_R are the summarized matching cost values of the same epipolar line calculated by the SAD. Once the SAD is computed for all pixels and for all disparity values and the DSI has been formed, a similarity accumulator will indicate the most likely disparity for each pixel.

2) DSI Processing: Although the SAD algorithm is quite robust and simple, it does not exhibit high accuracy and introduces several mismatches. Inherent ambiguities in occluded areas, regions with periodical structure or unstructured regions, produce random incorrect matching costs, which can be located anywhere in the DSI. As a result, the effectiveness of the similarity accumulator is reduced and many random incorrect disparity values are introduced in the extracted disparity map.

Thus, a new CA approach is proposed, in order to refine the DSI and improve the efficiency of the similarity accumulator. Since there are numerous rules that can be applied to improve the quality of the DSI, a considerable effort was devoted to explore the effects of different rules and use only those that have proven to be good in eliminating the wrong matching costs in the DSI.

The cellular structure used in the proposed system consists of k matching cost levels and is on the basis of CA formation (I, V, N, f), where $V{=}0, 1, \ldots, k-1$ is a set of cellular states and k the number of possible states. N represents the type of neighborhood for each transition rule, $I = (a, b)/1 \le a \le W, 1 \le b \le L$ where W and L are the dimensions of the disparity image, while the local transition rule f is from V_n into V [15], where n is the total number of neighbors for each CA rule. The value $DSI^{(t)}(i, j, d)$ of a site (i, j, d), where d is the disparity value and $DSI^{(t)}$ the matching cost value in disparity space image at time step t, is a 3-dimensional cellular automata with a rule that depends only on nearest neighbors and evolves according to equation:

$$DSI^{(t+1)}(i,j,d) = f[h^t(a)] \quad (4)$$

where $h^t(a) = (DSI^{(t)}(\alpha{+}\delta_1), \ldots, DSI^{(t)}(\alpha{+}\delta_n))$ is the neighborhood state function of cell α, for all $\alpha \in W \times L \times d$ and $\delta_{i=1,2,\ldots,n} \in N \to W \times L \times d$ at time t.

Thus, based on the CA, the following transition rules were considered for the DSI refinement stage:
i) For each disparity level and for each element of the same disparity value, set as:

$$DSI^{(t+1)}(i,j,d) = \frac{1}{9} \sum_{m=-1}^{1} \sum_{n=-1}^{1} DSI^{(t)}(i{+}m, j{+}n, d)$$

where N is the Moore neighborhood of a 3×3 pixel mask.

Since pixelwise cost calculations are generally ambiguous and wrong matches can easily have a lower or higher cost than the correct one, this rule eliminates the amount of cost variations that are unrepresentative of their surroundings in a small neighborhood of the same disparity plane. These variations may be caused by different factors, such as the noise in input stereo image pair.
ii) For each element of DSI with same (x, y) coordinates and for different disparity levels,

if $DSI^{(t)}(i,j,d{-}1) > \frac{1}{2} DSI^{(t)}(i,j,d)$ **or**
$DSI^{(t)}(i,j,d{+}1) > \frac{1}{2} DSI^{(t)}(i,j,d)$ **then**
$DSI^{(t+1)}(i,j,d) = 0.8 \cdot DSI^{(t)}(i,j,d)$
else if $DSI^{(t)}(i,j,d{-}1) < \frac{1}{2} DSI^{(t)}(i,j,d)$ **or**
$DSI^{(t)}(i,j,d{+}1) < \frac{1}{2} DSI^{(t)}(i,j,d)$ **then**
$DSI^{(t+1)}(i,j,d) = 0.6 \cdot DSI^{(t)}(i,j,d)$
end if

where the neighborhood N consists of the following cells $N{=}\{DSI^{(t)}(i,j,d{-}1), DSI^{(t)}(i,j,d), DSI^{(t)}(i,j,d{+}1)\}$.

This rule introduces a smoothness penalty for neighboring disparity changes, based on the fact that the matching costs between neighboring pixels that belong to the same surface (object) should have small variations. In many cases, pixelwise calculated matching cost values from a local window-based stereo correspondence algorithm yield non-unique or wrong correspondences due to textureless areas and ambiguities, leading to wrongly selected disparity values by the similarity accumulator. Assuming that the scene is formed by piecewise-smooth and Lambertian surfaces, the intensity differences $|I_l(i, j) - I_r(i, j)|$, from which the matching costs are computed, should be uniform in smooth areas, while the disparity discontinuities that lie on object boundaries should be aligned with equivalent intensity discontinuities (i.e. strong matching cost variations). Using these empirically determined

factors, it is possible to smooth the unwanted extreme variations in uniform areas but still keep them discretized in object boundaries and in areas with local gradients.
iii) The next CA rule is applied to each element of the DSI with the same disparity level.

```
k, p := 0;
for m, n := −2 to 2 do
    if DSI^(t)(i+m, j+n, d) ≤ (1/2)DSI^(t)(i, j, d) then
        k = k + 1;
    else if DSI^(t)(i+m, j+n, d) ≥ (1/2)DSI^(t)(i, j, d) then
        p = p + 1;
    end if
end for
if k ≥ mod_val then
    DSI^(t)(i, j, d)=0.4DSI^(t)(i, j, d)
else if p ≥ mod_val then
    DSI^(t)(i, j, d)=1.2DSI^(t)(i, j, d)
end if
```

where mod_val represents the number of times that the mode value of a 5×5 Moores neighborhood appears on this neighborhood. Furthermore, in the second and third CA rules, the scale factors were estimated after extensive experiments, based on observations about how the DSI modifications affect the extracted disparity maps and what possible areas in 3D space may cause false reconstructions. These rules target to smoothen the matching costs in a local neighborhood, while it should be also noted that the they were explicitly implemented after extensive testing to produce the maximum possible performance, according to the trade-off between accuracy and speed of the proposed technique.

Once the processing of the DSI is completed, a similarity accumulator indicates the most likely disparity value for each pixel (x, y) in the plane. In order to compute the disparity map, a search in the DSI for all disparity levels ($d_{\min}$ up to $d_{\max}$) is performed for every pixel. The disparity value where $D(x, y, d)$, with $d_{\min} \leq d \leq d_{\max}$, is the minimum for a pixel, is given as the corresponding pixel value for disparity map:

$$D(i, j) = \arg\min_{d \in [d_{\min}, d_{\max}]} DSI(i, j, d) \quad (5)$$

V. Hardware Implementation

The proposed system architecture follows an area-based similarity technique to extract the final disparity maps, combined with a CA DSI optimization operation. As in many other related works [14], [4], we assume that the intensities of corresponding points are the same and that the input images are captured from calibrated stereo cameras.

A. System Overview

The computational complexity of stereoscopic algorithms can easily be calculated. The main feature of stereo computation methods is their repetitiveness and their time-consumption, especially for software-based techniques on conventional computers. These repetitive calculations cause many memory references, making it difficult to meet real-time stereo vision performance. Thus, in order to achieve real-time performance while reducing the aforementioned computational effort, a parallel-pipelined hardware-based stereo architecture was designed, which was implemented on a single FPGA device of the Stratix IV family of Altera Devices. The typical operating clock frequency was found to be 168 MHz. It should be noted here that the proposed hardware architecture is also parametrizable in terms of the input image size, the correlation window size of the window-based stereo matching technique, and the levels of the disparity range of the final disparity map. The hardware design architecture can be seen in Fig. 3.

The proposed architecture is divided into three major tasks, pre-processing, stereo correspondence computation and DSI processing, and consists of corresponding hardware units: the Pre-Processing Unit (PPU), the DSI Creation Unit (DSI_CU) and the DSI Processing Unit (DSI_PU). The system also consists of a High-level Control Unit (HCU) that coordinates the different operations performed in the architecture, optimizes memory accesses through the Memory Logic module according to the algorithm requirements, and synchronizes data transfers between the PPU, the DSI_CU and the DSI_PU. It is also used to manage the data flow of the overlapping pixels between the internal memory and the different process stages of the system, in order to reduce the clock cycles needed to load image pixels into the processing units. Furthermore, from this unit, the user can also select the configuration parameters of the 3D reconstruction procedure, customizing the architectures functionality.

The whole stereo system architecture is based on intensive use of parallelism and pipelining design techniques, minimizing the area cost of the FPGA implementation and maximizing the total throughput of the system. In most cases, these techniques can be implemented without significant additional FPGA resources, at the expense of some additional system latency, due to the large amount of programmable registers found in most modern FPGA architectures. In addition, in order to achieve real-time throughput, many processing steps of the algorithm have been mainly realized as simple highly parallel dedicated processing elements, while in order to reduce latency, the processing units of the system have been designed with a throughput of one pixel per clock cycle. Furthermore, the main units of the system are implemented as combined entities, which enables data transitions between the processing elements without requiring many bus accesses. For data transitions between the different implemented hardware components has been used also a multi-buffering process, which allows the parallel and pipelined processing of data, avoiding writing to memory elements while other components still read from them. As a result, the final disparity output is generated in real-time, after a small initial pipeline latency, which is in the order of a few microseconds.

B. Pre-Processing Unit (PPU) Architecture

In order to reduce the effects of noise, a one-dimensional weighted mean filter is first applied on the input images of the stereo system. The simplicity of this linear filter is twofold: it

meets the requirements for image pre-processing, as well as the high speed imposed by the application. The architecture of the PPU unit is shown in Fig. 4a. The two color input images are initially stored in the internal memory of the system, in order to eliminate the necessity for fetching data from any external memory device. The color components of each image (R, G and B) are separated through a bus splitter, in order to be routed and processed in each one-dimensional weighted mean filter in a parallel manner. Its circuitry can be seen in Fig. 4b. Parallelizing this procedure (due to its low resource utilization), the two images are processed simultaneously, so six identical filters have been implemented. The main features of this unit are its low area cost and its fully parallel-pipelined architecture.

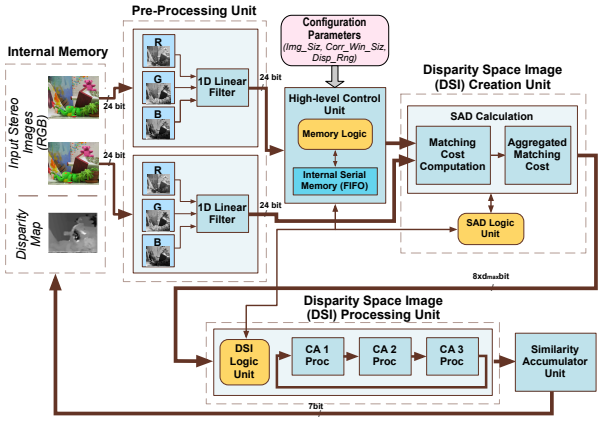


Fig. 3: Hardware architecture of the proposed real-time stereo vision system.

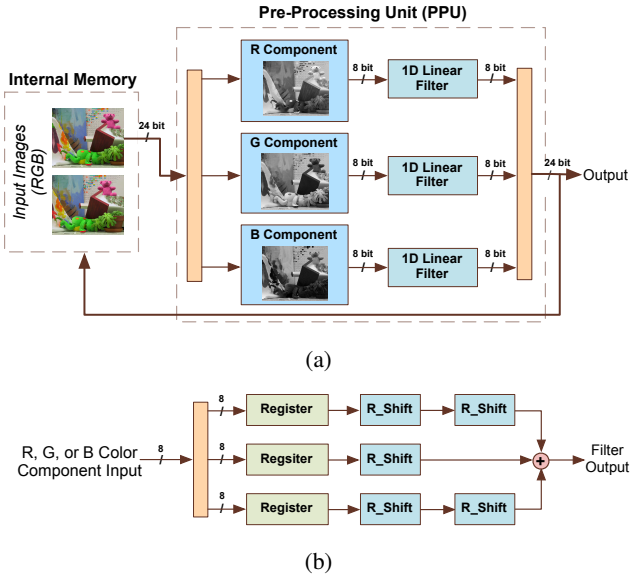


Fig. 4: (a) Pre-Processing Unit (PPU), and (b) one-dimensional linear pre-processing filter architecture.

C. DSI Creation Unit (DSI_CU) Architecture

In the DSI_CU unit, the matching cost of the corresponding pixels is calculated with the SAD window-based technique, and the 3D space of DSI is created for the disparity optimization procedure.

SAD can be directly implemented in hardware using only addition and subtraction operations. Taking into account the speed issues of real-time stereo applications, the proposed hardware implementation of the DSI_CU unit is presented in Fig. 5. As it is shown, the DSI_CU unit is divided into three basic stages, the Color Component Analysis, the Absolute Difference Calculation Module and the Sum of Absolute Differences Module. The architecture in Fig. 5 also consists of a Control Logic unit, which comprises a part of the HCU unit, in order to read/write data from/to the different modules of th‘e DSI_CU and coordinate the accesses from/to the internal memory. In the proposed architecture, there is also a Memory Module with two banks of registers (window and scanline registers) for temporarily storing the pixel values needed to the SAD computation process, along with a collection of intermediate pipeline registers/buffers, adders and subtractors to calculate the SAD values.

After the pre-processing stage, the SAD for a 5×5 pixel window is calculated and the DSI of the input stereo images is formed. The input of DSI_CU unit is the two 24-bit color depth filtered images, where each color component has 8-bit depth with a total of 255 different intensity values. The output of this unit is the calculated 8-bit SAD value, for all possible positions of the shifting window, and it is given in a single clock cycle (after an initial latency in the pipelining stage). To achieve this, a highly pipelined structure has been utilized, using adder and subtractor modules in a binary tree structure.

The Control Logic of the DSI_CU unit reads the pixel data of the filtered images in a row-wise mode, until all data needed to perform the SAD calculation along an entire scanline are fed into the Memory Arrangement. To reduce the computational cost and the hardware resources utilization for the SAD calculations, each of the RGB color component data was separately fed to the SAD Computation Unit through a multiplexer arrangement. This architectural arrangement improves the efficiency of the unit, minimizing the hardware resource utilization and leading to significant reduction in power consumption, although a small decrease in system's output frame rate is unavoidable, due to an additional delay in multiplexing stage. The DSI_CU Memory Arrangement uses three banks of 25+5 8-bit window registers for a working window of 5×5 for the left image, for all three color components and a bank of $3\cdot(W+1)\times L$ 8-bit scanline registers for the right image inputs. Through the Control Logic of the unit, an additional input signal is fed into the SAD Computation Unit, in order to determine the disparity range for the SAD calculation. In this way, the operating disparity range is customized and the performance of the system may be increased, when small disparity ranges are selected.

The pixel data needed for the SAD calculation, which have been temporarily stored in Memory Arrangement, are initially fed into the Absolute Differences Calculation module, in each clock cycle. This stage consists of $d_{\max}$ subtractor modules, which compute in a parallel-pipelined manner the absolute differences between the fixed window of the left image and

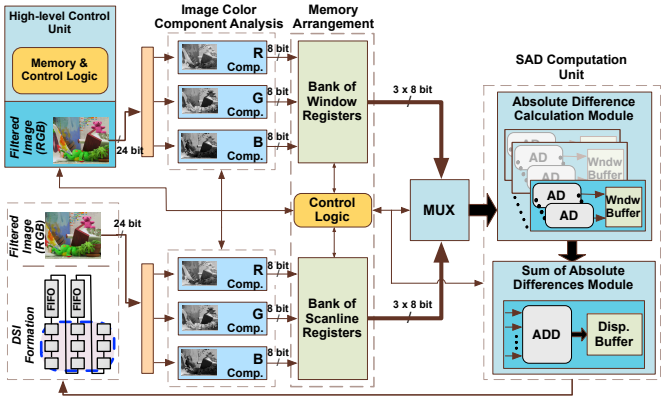


Fig. 5: DSI Creation Unit (DSI_CU) architecture.

the shifting window of the right image. Each subtractor module receives as input $2{\cdot}W^2$ 8-bit data words, computes the absolute difference of the pixel values and extracts an 8-bit vector, which is next added bitwise using binary tree adders in the Sum of Absolute Differences module.

On every clock cycle, the Control Logic unit shifts the current correlation window to the right and feeds the SAD Computation Unit with $3 \cdot W$ 8-bit new values from the bank of scanline registers. The other pixel data from the window and scanline registers remain the same as for the previous operation, and the SAD calculation is performed again for the new pixel data. After $d_{\max}$ right transitions, a new pixel value is added to the bank of window registers and an old one is shifted out in a FIFO (First-In-First-Out) pipeline architecture.

The SAD is computed for a given pixel, for all of the disparity ranges according to Eq. (2), and the final value, which is 13-bit long, is used to form the DSI. The created 3D space of DSI will be optimized then in the DSI_PU unit, through a hardware-based parallel-pipelined 2D CA structure.

D. DSI Processing Unit (DSI_PU) Architecture

1) CA 1 Processing Unit Architecture: The hardware implementation of the first CA structure is shown in Fig. 6. After the matching cost computation from the DSI_CU unit, the first CA rule is applied to a $3{\times}3$ Moore neighborhood in the first disparity image plane for DSI refinement. Since only 3 scanlines are needed to perform the operation described in Eq. (5), an efficient memory architecture with 2 FIFO memories which can store $2{\cdot}(L{-}3)$ matching cost values and 9 registers was used to increase processing speed, as shown in Fig. 7. As the working window of the first CA rule moves over the image, $2{\cdot}W$ overlapping pixels (where W is the width of the working window) exist between adjacent windows. The implemented memory architecture was used to temporarily store these pixels to reduce the clock cycles needed to load the matching cost values into the CA 1 PU. After an initial latency for filling up this memory module with the first $2L{+}3$ matching cost values from DSI_CU, in every pixel transition to the right of the disparity image plane, only the bottom right pixel of the working window is new and needs to be fed into the memory module. Thus, the output of this memory architecture is provided once per clock cycle, enabling a fast transition of the required working window to the next adjacent pixel.

After an initial latency required for filling up the FIFO memories and the 9 registers, the first 9 matching cost values are stored in a parallel fashion in equivalent 13-bit registers into the CA 1 PU, in order to be added in the PARAL_ADDER module according to Eq. (5). The addition process is fully parallelized for faster computation, so at the input of this module, the bitwise analyzers treat their input signals as a vector of bits rather than a single value, helping to load faster the input multi-line signals to the PARAL_ADDER module. As a result, 13 addition elements of 9-bit enter to PARAL_ADDER and the output value of this circuit is given in every clock cycle. The resulting summation value is divided in the Division module and the final output is inserted into the DSI_REP module, where the REPL Control Logic synchronizes the replacement of the matching cost value of the central pixel of Moore's neighborhood with the refined value extracted by CA 1 PU. The extracted values are fed into the next processing stage for further refinement.

2) CA 2 Processing Unit Architecture: The second CA structure performs the matching cost value processing along the d axes for further DSI refinement. More specifically, three directly adjacent neighboring pixels $(d{-}1, d, d{+}1)$ are needed for each comparison and a total of $L{\times}H{\times}d$ iterations are required for all DSI pixel processing. Fig. 8 shows the fully parallel-pipelined architecture of CA 2 PU, which is divided into four stages: DSI value fetching, matching cost analysis & division, matching cost comparison and DSI replacement. In addition, there are two Replacement Control Units (RCUs) for each part of the second CA rule, replacing the corresponding DSI values with the refined ones according to CA rule. The main components of the proposed architecture is a Control & Replacement Logic module, which is part of the HCU and coordinates the operations performed in the CA architecture and an on-chip memory block (bank of registers) for temporarily storing the matching cost values of DSI.

For CA 2 PU, three adjacent disparity image planes should be buffered, before the CA operation initiates. To achieve this, a DSI Memory Block was used, consisting of two image plane buffers along with four registers. Fig. 9 shows the hardware architecture of the proposed DSI memory block. After $2{\times}H{\times}L{+}2$ clock cycles, the In_Register and the two image plane buffers of DSI memory block are fulfilled with the first DSI values. Since CA 2 PU requires three input pixels with the same coordinates in the $x{-}y$ plane and different disparities (i.e. pixels $P(i,j,d)$, $P(i,j,d{-}1)$ and $P(i,j,d{-}2)$) from neighboring disparity image planes, the values of the three left top-most registers (two from the first two image planes $d{-}1$ and $d{-}2$, and the In_Register) of the DSI Memory Block can be used to feed this unit.

The First and the Second RCU units implement the first and the second part of second CA rule respectively, using each of them two Binary Comparator modules for the comparison of the middle with the other two 13-bit input signals, and one logic OR module for the final replacement decision. Binary comparison is one of the most computationally demanding tasks in the proposed stereo system, so a hardware-efficient

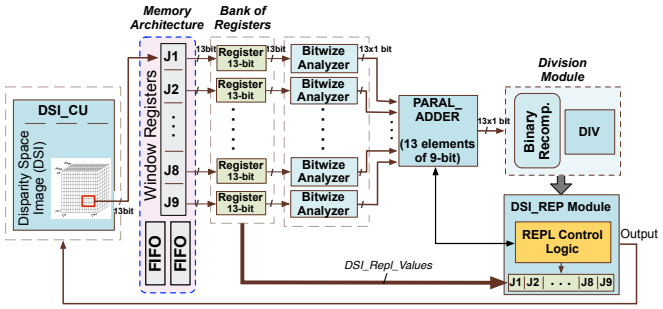


Fig. 6: Hardware architecture for the CA 1 Processing Unit.

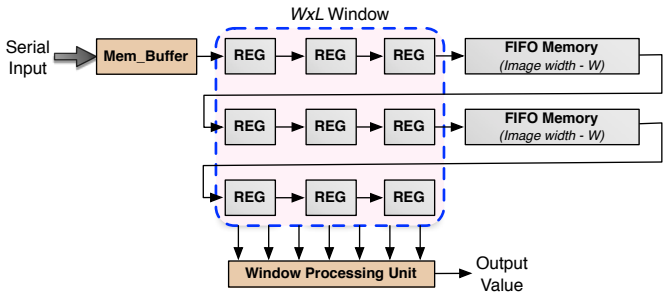


Fig. 7: Memory architecture for CA 1 Processing Unit.

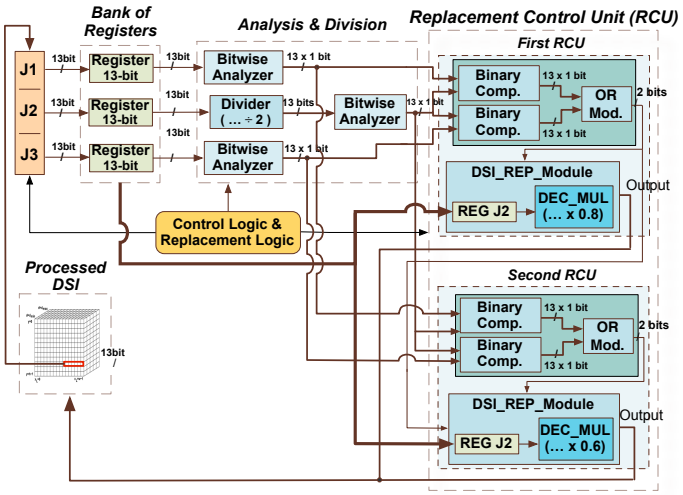


Fig. 8: Hardware architecture for the CA 2 Processing Unit.

implementation was designed using 2 13-bit input signals. Fig. 10 shows a simplified architecture of the proposed Binary Comparator implementation, using a comparison example with two 4-bit input signals, in order to demonstrate better the functionality of the proposed architecture. It is also worth noticing that the aforementioned circuit architecture is quite straightforward, using only logic gates for its implementation, and no complex calculations, like multiplications. This beneficial architecture reduces the hardware complexity, simplifying the implementation for larger scale comparators with more inputs, and making the 3D stereo reconstruction very fast. Moreover, the reduced size of this circuit allows for a parallel implementation of more than one comparators, giving the ability of comparing more than two pixel values at a time.

The output values of Analysis & Division stage are fed

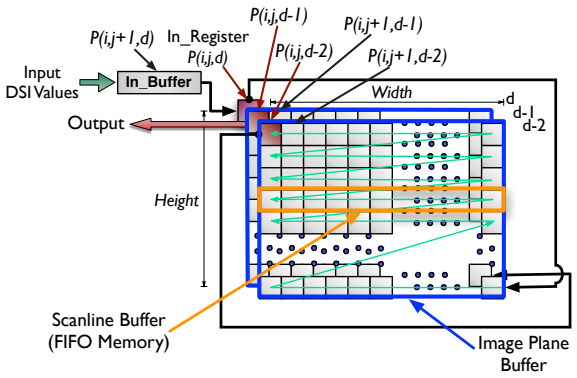


Fig. 9: Hardware architecture of the DSI Memory Block.

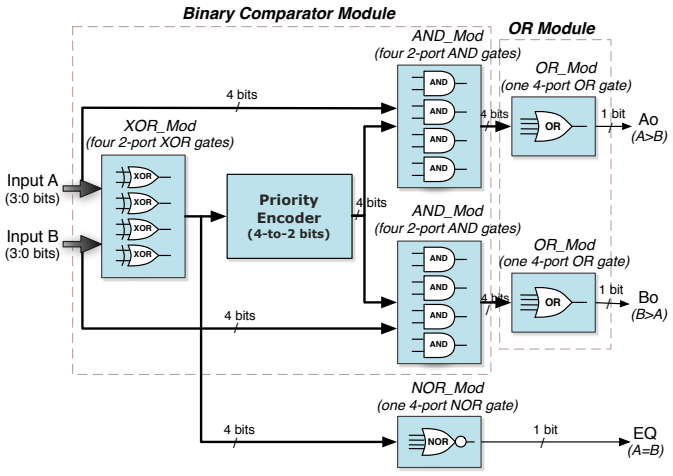


Fig. 10: The implemented binary comparator.

synchronously into the RCU units, they are compared in a parallel mode and the extracted results are driven into the DSI_REP modules in order to be replaced by the proper value.

3) CA 3 Processing Unit Architecture: Following the process described in the third CA rule, 25 pixel values of a 5×5 pixel Moore neighborhood in the same disparity image plane are compared with the neighborhood's central pixel, and the extracted results are summed for further comparison with the number of times (frequency) of the mode value presented in the Moore's neighborhood (mode_frq). Fig. 11 shows the fully parallel-pipelined hardware architecture of the CA 3 PU, which is divided into six stages: DSI value fetching, matching cost Analysis & Division, mask matching cost value comparison and counting, and DSI replacement. In addition, according to the third CA rule, there are also two Replacement Control Units (RCUs) and a Control & Replacement Logic module, which is part of the HCU and coordinates the data transfers and operations performed in the CA architecture. An on-chip memory block (bank of registers) is also included for temporarily storing the input matching cost values.

So, after an initial latency for filling up the memory module with 4 FIFO memories and 25 registers (i.e. an extension of architecture described in Fig. 7), an input register arrangement was used to store the 25 image pixels of the 5×5 working window. In Comp_Mod of Sum_Comp module the comparison between the neighboring pixel values of the mask with the cen-

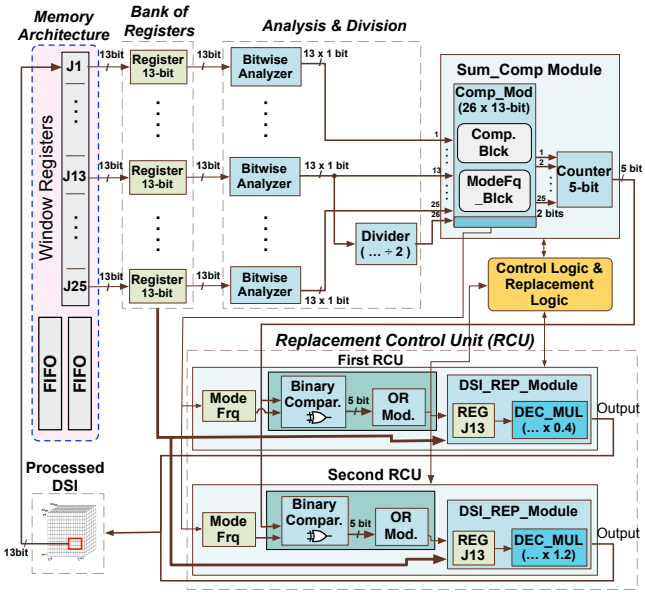


Fig. 11: Hardware architecture for the CA 3 Processing Unit.

tral pixel takes place (Comparator_Blck) using the hardware architecture presented in Fig. 10 with 25 13-bit input signals and equivalent 2-bit outputs.

For mode_frq calculation, the ModeFq_Blck was implemented, as shown in Fig. 12. Each Neighb_Comp sub-block is a comparator which compares one input pixel value with the other 24 of the 5×5 neighborhood, and if it stands more than 12 times then the output is assigned as the logic one. The priority encoder generates an output based on the highest Neighb_Comp sub-block output that emits a logic one. The priority encoder's output is fed into the mux, selecting the mode value of the neighborhood. After an initial latency of 4 clock cycles, the output of this sub-module appears once per clock cycle, while the extracted mode value is compared with the output of the counter in the next pipeline stage. The RCU, which comprises the final stage of CA 3 PU, implements the outer comparisons of the condition parts of third CA rule and the resulting product is fed into the next processing stage.

When the DSI operation is finished, the Similarity Accumulator Unit selects the proper disparity value for each pixel.

VI. Evaluation and Experimental Results

The hardware architecture presented in this work has been implemented on an FPGA device of the Altera Stratix IV family, with Altera Quartus II schematic editor. Table I summarizes the resource utilization report for the Altera FPGA. The operating clock frequency was found to be 168 MHz under the configuration presented in Table I, while in this table it is also presented the overhead caused by each separate processing unit of the device, keeping image size, window size and disparity range (a mid-range of 70 disparity levels was selected) constant. From Table I it is worth noticing the low device resource utilization of the proposed architecture mainly caused by the reduced design complexity (i.e. the majority of circuitry consists of logic gates) and the multiplexing schema of RGB color component in DSI computation unit.

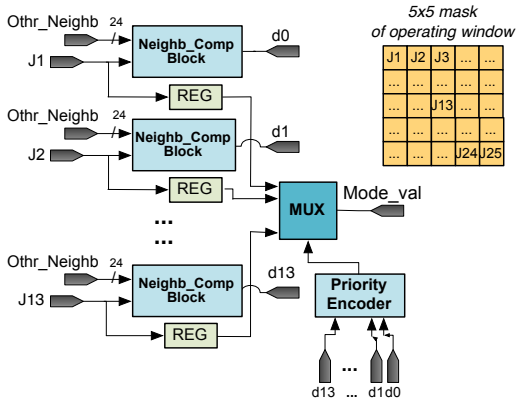


Fig. 12: Hardware architecture for the ModeFq_Blck sub-module.

Furthermore, the DSI calculation and refinement stages are based on local (i.e. window-based) image processing, so the size of the input stereo image pairs does not affect the device utilization directly. However, the size of the input image is directly related to the size of the on-board memory and the total latency/delay caused by the data transfers between the different units of the system. In addition, the operating disparity range and the window size of SAD linearly affect the consumed logic resources, while other resources are comparingly less affected. In addition, from Table I it can be inferred that the proposed hardware implementation leaves enough resources in the targeted FPGA device, such as required for the implementation of other additional processing stages that may improve the system's performance.

The evaluation results presented in this section are based on the test procedure reported by Scharstein and Szeliski [1] available at www.middlebury.edu/stereo. Many methods from the Middlebury test procedure, although being state-of-the-art in accuracy, have not been included in our comparison, since this paper focuses on real-time or near real-time hardware-based local methods, while these methods are far from such requirements. The evaluation data that were used as benchmarks are calibrated stereo image pairs from Middleburry's database, and consist of standard data sets, as well as some new Middlebury stereo image pairs. Representative quantitative and qualitative results for some image pairs from each category are presented. In the current experiments, fixed-value boundary conditions are applied, in which transition rules are only applied to non-boundary cells.

The resulting disparity maps from the proposed system are presented in Figs. 13 and 14. In Fig. 13 different stereo image pairs of varying image sizes and different disparity ranges are shown. As can be seen, the proposed system provides adequate disparity accuracy, considering the real-time performance. Fig. 14 provides a qualitative evaluation of our extracted results against other related architectures. Although many authors do not provided quantitative/qualitative results for their disparity maps, Fig. 14 confirms the performance of the proposed architecture. For example, the table and the area around it in the Tsukuba disparity map are more accurately reconstructed than with other approaches, as well as

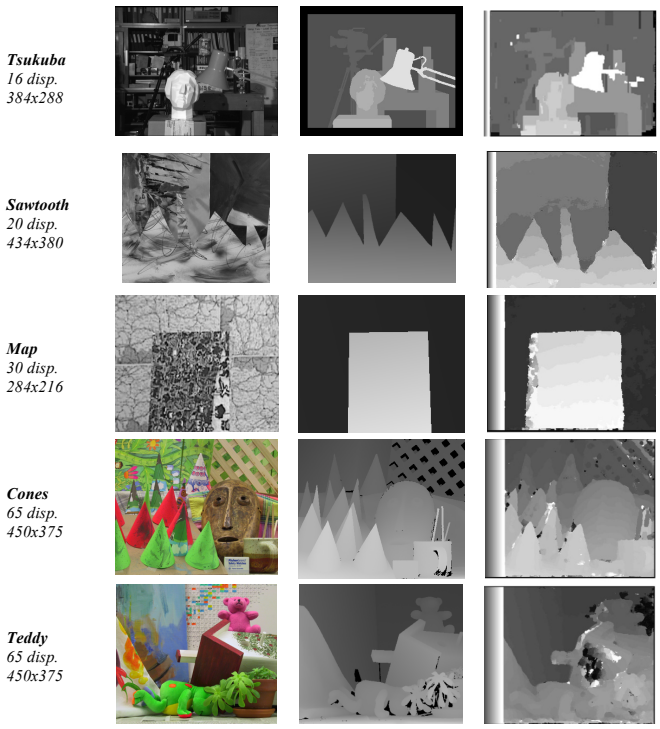


Fig. 13: Resulting disparity maps of stereo image pairs Tsukuba, Sawtooth, Map, Cones and Teddy. (a) Input images, (b) ground truth and (c) final disparity maps.

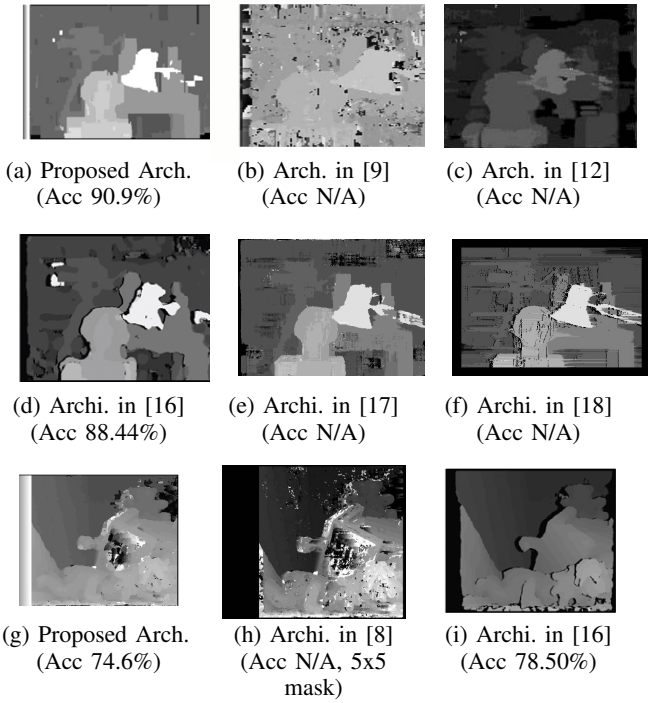
TABLE I: Resource utilization of target device.

Device (Altera Stratix IV)	Total Registers (%) (424,960)	Logic Utilization (%) (424,960)	Total ALUTs (%) (424,960)	Total LABs (%) (21,248)	Total Pins (%) (1,112)
DSI_CU	15,2	14,7	4,9	6,3	4,4
DSI_PU	9,8	6,8	6,3	4,1	3,5
Total	25,6	23,2	12,6	11,4	8,5

the camera on the tripod. Of course, some errors still remain in the final disparity maps, which are caused either by the wrong replacements of CA rules, or by mismatches due to the SAD algorithm. Quantitative results regarding accuracy under various configurations are given in Table II. The evaluation metric was the error rate ε, indicating the percentage of bad pixels whose absolute disparity error is greater than 1. It was calculated with respect to the ground truth maps from the Middleburry evaluation database using the cost function:

$$\varepsilon = \frac{1}{N} \sum_{i,j} |d_{\text{comp}}(i,j) - d_{\text{ground}}(i,j)| > 1 \qquad (6)$$

The results in Table II demonstrate that as the disparity range of the input images increases, the percentage of bad pixels in the resulting disparity maps also increases. This is caused mainly by the nature of the DSI calculation and optimization procedures, which were focused on local pixel processing. A global DSI refinement process, based also on CA, could potentially perform better at the expense of increased computational complexity. More specifically, changing the features, and particularly, the size of the cell neighborhood (in order to have a global behavior), the size of the CA rule will change according to equation $Rulesize{=}2^{2^n}$. This means that the rule space will increase exponentially as the cell neighborhood size increases, leading to similar increase in computing complexity of the hardware design.



(a) Proposed Arch. (Acc 90.9%) (b) Arch. in [9] (Acc N/A) (c) Arch. in [12] (Acc N/A)

(d) Archi. in [16] (Acc 88.44%) (e) Archi. in [17] (Acc N/A) (f) Archi. in [18] (Acc N/A)

(g) Proposed Arch. (Acc 74.6%) (h) Archi. in [8] (Acc N/A, 5x5 mask) (i) Archi. in [16] (Acc 78.50%)

Fig. 14: Qualitative evaluation for Tsukuba and Teddy stereo image pairs for different approaches.

TABLE II: Quantitative results of the proposed system, regarding accuracy under various configurations.

	Disparity range	Error ε (%)
Tsukuba (384×288 pxl)	16	9.1
Sawtooth (434×380 pxl)	20	10.7
Map (284×216 pxl)	30	17.3
Cones (450×375 pxl)	65	24.7
Teddy (450×375 pxl)	65	26.4

Tables III and IV list the performance evaluation results of the proposed system compared to other stereo vision systems with similar window matching techniques or with other disparity computation methods, respectively. Configuration parameters such as image size and disparity range are also provided, while a normalized performance index (NPI) in terms of $image\_size{\times}disparity\_range{\times}fps/frequency$ is calculated for evaluation purposes. As demonstrated, the proposed hardware implementation shows improvements over the majority of previous systems in terms of performance and processing rates, including those that produce real- or near real-time results with high disparity levels. Although there are other systems that achieve better performance indices, e.g. [11], [17] and [21], other factors, such as the different configuration parameters, the total amount of calculations per

TABLE III: Performance evaluation results of the proposed system compared to other related hardware-based stereo vision systems with similar window matching techniques.

	Implemented Device	Matching Method	Image Size	Disparity Range	Window Size	Frequency (MHz)	Frames per Second (fps)	Normalized Performance Index
Hile [8]	N/A	SAD	521×480	32	N/A	N/A	30	N/A
Miyajima [9]	FPGA (Xilinx)	SAD	640×480	80	5×5	40	26	15.97
Arias-Estrada [19]	FPGA (Xilinx)	SAD	320×240	16	7×7	66	71	1.32
Lee [11]	FPGA (Xilinx)	SAD	640×480	64	32×32	10	30	58.98
Hariyama [12]	FPGA (Altera)	SAD	64×64	64	8×8	86	5063	15.43
Kuhn et al. [20]	ASIC	SSD/Census	256×192	25	Census & Corr.	75	50	0.81
Ambrosch [21]	FPGA (Altera)	SAD	450×375	100	9×9	110	600	92.04
DeepSea [7]	ASIC	Census	512×480	52	N/A	60	200	42.59
Jia et al. [10]	FPGA (Xilinx)	SSAD	640×480	64	N/A	N/A	30	N/A
Gudis et al. [22]	FPGA (Altera)	Pyramid decomposition	$2k$×$2k$	N/A	N/A	200	N/A	N/A
Proposed Arch.	FPGA (Altera)	SAD	640×480	70	5×5	168	114	14.59

TABLE IV: Performance evaluation results of the proposed system compared to other hardware-based stereo vision systems.

	Archit. in [23]	Archit. in [24]	Archit. in [16]	Archit. in [17]	Archit. in [18]	Archit. in [25]	Archit. in [26]	Archit. in [27]	Proposed Archit.
Device	FPGA (Altera Stratix S80)	FPGA (Xilinx Virtex II)	GPU (Nvidia GeForce 7900)	FPGA (Xilinx Virtex II Pro)	FPGA (Xilinx Virtex V)	FPGA (Xilinx Virtex II Pro)	FPGA (Xilinx Virtex V)	FPGA (Xilinx Virtex IV)	FPGA (Altera Stratix IV)
Image Size	640×480	1280×690	512×512	640×480	640×480	640×480	1280×720	640×480	640×480
Disp. Range	128	9, 15, 29	64	128	128	25	96	64	128
Frames/sec	30	52	38	63.54	30	8.9	2.5	230	47
Norm. Perform. Index	N/A	14.95	N/A	30.96	8.86	0.68	1.1	18.41	11.27

second (i.e. processing rate) or the complexity-accuracy and quality-performance trade-offs should also be taken into account. For example, from Table III, our architecture produces 640×480 disparity maps with 70 disparity levels and 114 fps, which makes a total amount of about 2.45 billion disparity calculations per second, while from Table IV arises that it makes around 1.85 billion calculations per second. Other architectures that their performance (i.e. NPI) outperforms the proposed one several times, such as the system in [11] (almost 4 times) and the system in [17] (almost 3 times) have a total amount of almost 590 million and 2.5 billion calculations per second, respectively, which shows that our system can process almost 4 times more data than the system in [11] and only about 25% less than the system in [17]. In addition, with respect to other approaches, the proposed one maximizes the trade-off between accuracy and performance, producing more accurate disparity maps than almost all other techniques with higher NPI or processing rates. For example, although the system in [21] outperforms the proposed one several times, the error rate of its extracted disparity maps for Teddy image pair is almost 44% (against 25% of the proposed one). Furthermore, comparing the Tsukuba image pair results with the results extracted by systems in [11] and [17], it is obvious that they are more cluttered and the object boundaries are more distorted than the proposed one, since they suffer from inter-scanline inconsistencies. The proposed technique can eliminate these errors using an efficient CA architecture, proving that it exhibits high potentials for real-time applications with tight time constraints and increased processing and accuracy demands.

Our observations show, that the frequency decreases as the disparity range increases, suggesting that if we want to allow more than 40 disparity levels, we need to introduce more pipeline stages at the expense of increasing the resource usage.

In Table V, the impact of the input image size on the performance of the proposed system is presented, keeping the disparity range and window size constant. Bandwidth limitations affect also significantly real-time systems, so the performance of the system's processing units is also evaluated when the bandwidth is 32-bits/cycle. It can be observed that the performance (i.e. output frame rate) of the system is almost inversely proportional to the input image size, while it exhibits real-time response even for large size input stereo images. However, increasing the disparity range will lead to a decrease in the system's frequency, indicating that more hardware resources and more pipeline stages are necessary for using more disparity levels. More specifically, Table VI shows the DSI_CU's and DSI_PU's resource and performance results for different disparity level configurations up to 128. As it is shown, there is an almost linear increase in the utilization of the FPGA resources, which is caused mainly from the additional elements used for calculating the SAD values, which are increasing when disparity range becomes larger, and the additional parallel components (e.g. comparators, slice registers, buffers, etc.) used for the different pipelining stages and the CA operations. The operating frequency is also

TABLE V: Frame rate output (fps) of the proposed architecture.

	System's unit performance		
Image Size (pixels)	**DSI_CU**	**DSI_PU**	**Overall FPGA System**
320×240	1467	461	455
640×480	398	120	114
800×600	259	79	73
1024×768	163	50	45
1280×1024	98	32	27

TABLE VI: Resource utilization for different disparity level configurations.
(Image size = 1024×768, window size = 7×7)

Disparity range	**20**	**40**	**70**	**128**
Total registers(%)	22,7	23,4	24,1	25,2
Logic Utilization(%)	18,6	19,4	20,5	22,3
Frequency (MHz)	177,2	174,3	168,1	156,4

TABLE VII: Resource utilization for different window size configurations.
(Image size = 1024×768, disparity range = 70)

Window size	**3×3**	**5×5**	**7×7**	**9×9**
Total registers(%)	22,8	23,5	24,1	24,8
Logic Utilization(%)	19,5	19,9	20,5	21,4
Frequency (MHz)	186,1	178,7	168,1	154,7

TABLE VIII: Resource utilization for different image size configurations.
(window size = 7×7, disparity range = 70)

Image size	320×240	640×480	800×600	1024×768	1280×1024
Total registers(%)	21,2	22,3	23,2	24,1	25,4
Logic Utilization(%)	17,6	18,5	19,2	20,5	22,7
Frequency (MHz)	172,1	171,4	170,1	168,1	165,7

provided, indicating that when the disparity range increases, the frequency decreases, meaning that the hardware overhead becomes higher at high disparity ranges.

It is also worth noticing that keeping the disparity range and image size constant while increasing the window size, there is also an increase in the utilization of the FPGA, since more image lines are needed in every processing stage to be fed to the bank of scanline registers and to the memory arrangements, in order to compute the SAD values and the results from each CA PU. Table VII illustrates the systems resource and performance results for different window size configurations while image size and disparity levels remain the same.

As it is indicated, an increase in the resource utilization components is caused mainly by the increase of the window size of the correlation mask, and more scanline register, buffers and other hardware components (i.e. adders, subtractors, etc.) for the intermediate pipeline stages are needed. The operating frequency also decreases as the correlation window size increases. In addition, Table VIII shows the systems resource and performance results for different image size configurations, keeping window size and disparity range constant. In this case, we notice also an increase to the resources of the system as the image size increases, since image size affects strongly the amount of hardware components used in the system. The system clock frequency is also slightly affected, having a small decrease as the size of input images increases.

With respect to I/O bandwidth, it affects the performance of the proposed system mainly when an external data resource (e.g. memory) is used, where the delay to fill the scanline buffers with the temporarily stored pixels needed for the window correlation is almost proportionally connected to the bandwidth of the system. In addition, the performance of the system is also decreased as the I/O bandwidth decreases, since the data flowing into the system limits its throughput.

VII. Conclusions

In this work, a new hardware based architecture for real-time disparity map computation was developed, targeting real-time 3D reconstruction applications. A fast stereo matching algorithm was used for DSI formation and an efficient DSI optimization technique, based on a highly effective CA structure, was implemented on a single FPGA device. Despite the simple construction of CA, it has been shown that they are capable of highly complex behavior, due to their inherent parallelism and local interconnection.

Timing and processing constraints for real-time applications were met using a fully parallel-pipelined architecture, adding only to output latency of the system and ensuring increased output frame rates with accurate disparity maps. Both qualitative and quantitative experimental results verify the efficiency of the proposed stereo vision system under a variety of configurations concerning the quality of the produced disparity maps and the frame rate output.

References

[1] 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, pp. 7–42, 2002.

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

[3] C. Ttofis and T. Theocharides, "Hardware design considerations for edge-accelerated stereo correspondence algorithms," *VLSI Design*, vol. 2012, p. 4, 2012.

[4] R. Šára, "Finding the largest unambiguous component of stereo matching," in *Proceedings on Computer Vision (ECCV '02)*, 2002, pp. 900–914.

[5] H. Hirschmuller, "Improvements in real-time correlation-based stereo vision," in *Proceedings IEEE Stereo and Multi-Baseline Vision (SMBV '01)*, 2001, pp. 141–148.

[6] R. Giryes, A. Bronstein, Y. Moshe, and M. Bronstein, "Embedded system for 3d shape reconstruction," in *European DSP Education and Research Symposium (EDERS '08)*, 2008, pp. 265–272.

[7] J. Woodfill, G. Gordon, and R. Buck, "Tyzx deepsea high speed stereo vision system," in *Conference on Computer Vision and Pattern Recognition Workshop (CVPRW '04)*, 2004, pp. 41–41.

[8] H. Hile and C. Zheng, "Stereo video processing for depth map," Technical Report, University of Washington, Tech. Rep., 2004.

[9] Y. Miyajima and T. Maruyama, "A real-time stereo vision system with fpga," *Field Programmable Logic And Application*, pp. 448–457, 2003.
[10] Y. Jia, X. Zhang, M. Li, and L. An, "A miniature stereo vision machine (msvm-iii) for dense disparity mapping," in *Proceedings International Conference on Pattern Recognition (ICPR '04)*, vol. 1, 2004, pp. 728–731.
[11] S. Lee, J. Yi, and J. Kim, "Real-time stereo vision on a reconfigurable system," *Embedded Computer Systems: Architectures, Modeling, and Simulation*, pp. 225–236, 2005.
[12] 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. 88, no. 12, pp. 3516–3522, 2005.
[13] J. Meiss, *Differential Dynamical Systems (Monographs on Mathematical Modeling and Computation)*. Society for Industrial and Applied Mathematics, 2007.
[14] G.-T. Michailidis and I. Andreadis, "A real-time stereo correspondence algorithm based on 2D cellular automata," in *International Workshop on Advanced Image Technology*, 2010.
[15] A. Popovici and D. Popovici, "Cellular automata in image processing," in *Fifteenth International Symposium on Mathematical Theory of Networks and Systems*, 2002.
[16] J. Lu, S. Rogmans, G. Lafruit, and F. Catthoor, "High-speed dense stereo via directional center-biased windows on graphics hardware," in *3DTV Conference*, 2007, pp. 1–4.
[17] S. Sabihuddin, J. Islam, and W. MacLean, "Dynamic programming approach to high frame-rate stereo correspondence: A pipelined architecture implemented on a field programmable gate array," in *Canadian Conference on Electrical and Computer Engineering (CCECE '08)*, 2008, pp. 001 461–001 466.
[18] C. Banz, S. Hesselbarth, H. Flatt, H. Blume, and P. Pirsch, "Real-time stereo vision system using semi-global matching disparity estimation: architecture and fpga-implementation," in *International Conference on Embedded Computer Systems (SAMOS '10)*, 2010, pp. 93–101.
[19] M. Arias-Estrada and J. Xicotencatl, "Multiple stereo matching using an extended architecture," in *Field-Programmable Logic and Applications*, 2001, pp. 203–212.
[20] M. Kuhn, S. Moser, O. Isler, F. Gurkaynak, A. Burg, N. Felber, H. Kaeslin, and W. Fichtner, "Efficient asic implementation of a real-time depth mapping stereo vision system," in *IEEE Midwest Symposium on Circuits and Systems*, vol. 3, 2003, pp. 1478–1481.
[21] K. Ambrosch, M. Humenberger, W. Kubinger, and A. Steininger, "Sad-based stereo matching using fpgas," *Embedded Computer Vision*, pp. 121–138, 2009.
[22] E. Gudis, G. van der Wal, S. Kuthirummal, and S. Chai, "Multi-resolution real-time dense stereo vision processing in FPGA," in *IEEE International Symposium on Field-Programmable Custom Computing Machines*, 2012, pp. 29–32.
[23] D. Masrani and W. MacLean, "A real-time large disparity range stereo-system using fpgas," in *Proceedings International Conference on Computer Vision Systems (ICVS '06)*, 2006, pp. 13–13.
[24] 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.
[25] S. Hadjitheophanous, C. Ttofis, A. Georghiades, and T. Theocharides, "Towards hardware stereoscopic 3D reconstruction a real-time FPGA computation of the disparity map," in *Proceedings Design, Automation & Test in Europe Conference & Exhibition*, 2010, pp. 1743–1748.
[26] J. Pérez, P. Sanchez, and M. Martinez, "High memory throughput fpga architecture for high-definition belief-propagation stereo matching," in *International Conference on Signals, Circuits and Systems (SCS '09)*, 2009, pp. 1–6.
[27] S. Jin, J. Cho, X. Dai Pham, K. Lee, S. Park, M. Kim, and J. Jeon, "Fpga design and implementation of a real-time stereo vision system," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 1, pp. 15–26, 2010.

Georgios - Tsampikos Michailidis received the Diploma and the M.Sc. degrees from the Department of Electrical and Computer Engineering, Democritus University of Thrace (DUTH), Xanthi, Greece, in 2008 and 2010, respectively. He is currently pursuing his Ph.D degree at the Department of Informatics at the University of Zürich. He is member of the Technical Chamber of Greece (TEE). His research interests mainly focus on the area of machine vision, 3D computer graphics and scientific visualization, intelligent systems and on the design of digital hardware architectures for real-time and low-power computer vision applications. He was also the recipient of the Best Paper Award EUREKA 2009.

Renato Pajarola received a Dr. sc. techn. in Computer Science in 1998 from the Swiss Federal Institute of Technology (ETH) Zurich. After a postdoc in the Graphics, Visualization & Usability Center at Georgia Tech he joined the University of California Irvine in 1999 as an Assistant Professor where he founded the Computer Graphics Lab. Since 2005 he has been leading the Visualization and Multi-Media Lab at the University of Zürich and is now a Full Professor in the Department of Informatics. His research interests include real-time 3D graphics, scientific visualization and interactive 3D multimedia. He has published a wide range of peer-reviewed research articles in geometric modeling, real-time 3D graphics and interactive scientific visualization, and is a frequent committee member and reviewer for top conferences and journals. He is a member of the IEEE Computer Society, ACM and SIGGRAPH.

Ioannis Andreadis received the Diploma degree from the Department of Electrical and Computer Engineering, Democritus University of Thrace (DUTH), Xanthi, Greece, in 1983 (1978 1983 IKY Scholarship), and the M.Sc. and Ph.D. degrees from the University of Manchester, Manchester, U.K., in 1985 and 1989, respectively. He joined the Department of Electrical and Computer Engineering, DUTH, in 1993. He supervised 10 Ph.D. theses, 16 M.Sc. theses, and 60 Diploma dissertations. He is the author of an Institute of 2010. He has published 200 refereed publications in Physics Select Paper in book chapters, international journals, and conferences. His research interests are mainly in electronic systems design, intelligent systems, and machine vision. Prof. Andreadis is a Fellow of the Institution of Engineering and Technology (IET), London, U.K. He has been an Associate Editor of the Pattern Recognition Journal since 1996 and a member of the Technical Chamber of Greece since 1983. He was a member of the Board of Governors of the European Commission Joint Research Center from 2008 to 2010. He has served as a Reviewer for leading scientific journals in his areas of interest and a member of Program Technical Committees of premier conferences such as ICIP and ICASSP. He was the recipient of the IET Image Processing Premium Award in 2009, the Best Paper Award (Computer Vision & Applications) in PSIVT 2007, and the Best Paper Award in EUREKA 2009.