High-Quality Real-Time Hardware Stereo Matching Based on Guided Image Filtering

Christos Ttofis and Theocharis Theocharides
KIOS Research Center for Intelligent Systems and Networks
Department of ECE, University of Cyprus
{ttofis.christos, ttheocharides}@ucy.ac.cy

***Abstract*— Stereo matching is a vital task in several emerging embedded vision applications requiring high-quality depth computation and real-time frame-rate. Although several stereo matching dedicated-hardware systems have been proposed in recent years, only few of them focus on balancing accuracy and speed. This paper proposes a hardware-based stereo matching architecture that aims to provide high accuracy and concurrently high performance in embedded vision applications. The proposed architecture integrates a compact and efficient design of the recently proposed guided image filter; an edge-preserving filter that reduces the hardware complexity of the implemented stereo algorithm, while at the same time maintains high-quality results. A prototype of the architecture has been implemented on a Kintex-7 FPGA board, achieving 60 fps for 720p resolution images. Moreover, the proposed design delivers leading accuracy when compared to state-of-the-art hardware implementations.**

Keywords—Stereo Matching; Embedded Systems; FPGAs;

I. INTRODUCTION

Stereo matching, the task of matching the images taken from a stereo camera and extracting the depth of objects in a scene [*1*], is commonly employed in embedded vision applications such as intelligence surveillance, autonomous vehicles and mobile robots [*1*]. Such applications need to satisfy real-time processing speed, high matching accuracy and low-power consumption constraints. The matching algorithm and the implementation platform are both factors that play a significant role in satisfying the requirements of an embedded stereo matching system. *Global* stereo matching algorithms produce very accurate results [*2*], but rely on the high-end hardware resources of multi-core CPUs and/or GPU platforms to achieve real-time processing. Such platforms therefore appear to be unsuitable for the realization of stand-alone stereo matching systems, and also consume excessive power, which is not desirable in battery-powered mobile and embedded devices. In contrast, *local* algorithms can be greatly benefited by the use of parallel structures implemented on either FPGAs or ASICs, providing the necessary computational power and energy efficiency for embedded vision applications [*3*].

To this end, several real-time stereo matching hardware systems have been developed during the past decade. However, the majority of them have implemented local algorithms that rely on standard block-based aggregation with a fixed support window [*3*]. While these algorithms can achieve very high frame rates when implemented in hardware [*3*], they lead to low matching accuracy [*4*]. As such, a few attempts have been made recently to implement dedicated hardware architectures of more accurate algorithms, such as Semi Global Matching (SGM) [*5*], [*6*] and Adaptive Support Weight (ADSW) [*7*], [*8*], [*9*]. For the past few years, hardware implementations based on SGM and ADSW algorithms have become the preferred solution towards higher matching accuracy in embedded vision applications. However, existing implementations have made several modifications and simplifications to adapt the algorithms for real-time processing, resulting in noticeable quality reduction compared to the original software algorithms. In addition, their high memory and hardware demands might limit their scalability to higher resolution images. Recently, the idea to utilize the Guided Image Filter (GIF) [*10*] in local ADSW stereo matching algorithms has been proposed to reduce the complexity of the cost aggregation step. Such software implementations have yielded promising results [*11*].

Motivated by the results of the software implementation presented in [*11*], this paper proposes a fully pipelined, parallel and scalable stereo matching hardware architecture that integrates the recently proposed GIF. There are two main novel contributions in this paper. First, it presents a new and efficient hardware design of the GIF (that can be potentially adopted in other uses of the filter). Second, it explores and concurrently discusses the hardware design parameters and optimizations involved in integrating the GIF hardware architecture in the cost aggregation step of ADSW-based hardware stereo matching systems. The latter reduces the overall hardware complexity of cost aggregation, which in turn allows real-time stereo matching of high definition images (HD), as well as improvements of the overall matching accuracy, thanks to the edge-preserving property of the GIF. The paper furthermore presents a complete prototype of the proposed GIF-based stereo matching architecture on a Kintex-7 FPGA board. The prototype was built to support 720p@60Hz HD video sequences (captured from a custom-built stereoscopic camera system) and various disparity ranges. Additionally, qualitative and quantitative evaluation, based on Middlebury benchmark image sets [*2*], shows that the developed FPGA prototype provides competitive accuracy compared to existing state-of-the-art stereo matching hardware systems.

In the following: Sections II & III provide background and related work, while Section IV presents the proposed hardware implementation. Section V shows results and comparison with related work. Finally, Section VI concludes the paper.

II. BACKGROUND ON STEREO MATCHING

A. Overview & Classification of Stereo Matching Algorithms

Stereo matching is a technique aimed at inferring depth information of a scene from a pair of stereo images (usually called *reference* and *target* images) [*3*]. The depth is determined by locating corresponding pixels in the stereo

This work was co-funded by the Republic of Cyprus through the Research Promotion Foundation and the EUREKA Organization under the Eurostars Programme (Project: E! 5527 RUNNER).

978-3-9815370-2-4/DATE14/©2014 EDAA

images. Given that the input images are rectified [12], the correspondence of a pixel at coordinate (x, y) of the reference image, can only be found at the same vertical coordinate y, and within a maximum horizontal bound, called *disparity range D* ($d_m - d_M$), in the target image. The disparity is then computed as the location difference of corresponding pixels in both images. The disparities of all pixels form a *disparity image*, or *disparity map*, from which depth information can be extracted.

According to [2], stereo matching algorithms mostly follow four steps: 1) matching cost computation, 2) cost aggregation, 3) disparity computation/optimization and, 4) disparity refinement. Moreover, [2] classifies stereo matching algorithms into two broad categories: *global* and *local*. Global algorithms are usually formulated as an energy minimization problem, which is solved with techniques such as Dynamic Programming, Graph Cuts and Belief Propagation. Such methods produce very accurate results at the expense of high computational complexity and memory needs. The Semi-Global Matching (SGM) [5], [6] methods renounce part of the accuracy by approximating a global 2D function using a sum of 1D optimizations from all directions through the image. SGM methods are therefore more affordable for dedicated hardware implementation, but they still consume excessive memory to store the temporary cost of different aggregation paths.

In contrast, local algorithms determine the disparity associated with a minimum cost function (see [2] for a review) at each pixel by performing block matching and winner-takes-all optimization. Hence, they have lower computational complexity and memory requirements compared to global and SGM methods. Early local algorithms relied on simple aggregation strategies that perform block matching by using either a fixed (typically square) window, or multiple windows with different sizes. However, these approaches are prone to matching errors at depth discontinuity regions; they blindly aggregate pixels belonging to different disparities due to the use of a fixed window (shape and/or size) [4]. Among local algorithms, the most recent adaptive support weight (ADSW) methods are currently the most accurate. They work by assigning different weights to the pixels in the support window based on their color/ proximity distances to the central pixel. In this way, they aggregate only those pixels that lie at the same disparity, leading to very good quality at depth borders [4]. Despite their good quality results, ADSW algorithms cannot take advantage of the "integral image" or "sliding window" techniques, as the adaptive weights have to be recomputed at every pixel. This makes the cost aggregation's hardware complexity directly dependent on the support window size.

B. Stereo Matching using Guided Image Filtering

The recently proposed Guided Image Filter (GIF) [10] has been employed in [11] to reduce the complexity of the cost aggregation step in ADSW methods, leading to a high-quality fast and simple algorithm (Fig. 1), with the following steps:

1) ***Cost Volume Construction.*** This step calculates a matching cost for each pixel p at all possible disparities. The output is a three-dimensional structure consisting of D cost images (*Stereo Cost Volume - SCV*). Each cost is computed as the truncated absolute difference of colors and gradients, a metric that exhibits good robustness to illumination changes

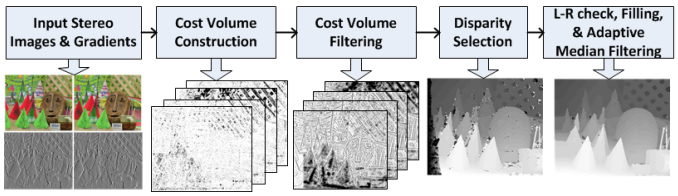

Fig. 1. Major Steps of the GIF-based Stereo Matching Algorithm.

[11]. The overall cost function $C(p,d)$ is computed with (1)-(3), where a is used to balance the influence of the color and gradients terms, and T_c and T_g are truncation thresholds.

$$M(p,d) = \sum_{i=1}^{3} \left| I^i_{left}(p) - I^i_{right}(p-d) \right| \quad (1)$$

$$G(p,d) = \left| \nabla_x\left(I_{left}(p)\right) - \nabla_x\left(I_{right}(p-d)\right) \right| \quad (2)$$

$$C(p,d) = a \cdot min\left(T_c, M(p,d)\right) + (1-a) \cdot min\left(T_g, G(p,d)\right) \quad (3)$$

2) ***Cost Volume Filtering.*** This step utilizes the GIF to smooth each slice of the SCV. Due to its edge-preserving proterty, the GIF leads to good accuracy at depth discontinuities. Typically, the filtered cost value at p and disparity d is a weighted average of the pixels in the same slice of the SCV, and is expressed as in (4).

$$q(p,d) = \sum W_{i,j}(I) C(p,d) \quad (4)$$

The GIF generally uses a *guidance* image I to filter a *guided* image f. In our case, the guidance image is the grayscale reference image, while the guided image is a slice (x, y) of the SCV. The filter weights are defined as in (5), where μ_k and σ_k are the mean and the variance of I in a squared window ω_k with dimensions $r \times r$, centered at pixel k. $|\omega|$ is the number of pixels in the window and ε is a smoothness parameter. An inherent advantage of the GIF is that the weights can be computed with some linear operations (see [10]), which can be decomposed into a series of mean filters with windows radius r. This leads to an efficient algorithm (pseudocode given in Fig. 2).

$$W_{i,j} = \frac{1}{|\omega|^2} \sum_{k:(i,j)\in\omega_k} \left(1 + \frac{(I_i - \mu_k)(I_j - \mu_k)}{\sigma_k^2 + \varepsilon}\right) \quad (5)$$

3) ***Disparity Selection.*** Once the SCV slices are filtered, the best disparity for pixel p is chosen through a simple Winner-Takes-All (WTA) minimization approach using (6).

$$d_p = \underset{d \in D}{\operatorname{argmin}}\, q(p,d) \quad (6)$$

4) ***Disparity Refinement.*** A left/right consistency check (L-R check) is performed. Hence, the disparity map, D_R, using the right image as reference is also computed. The L-R check marks disparities as invalid if the disparity $D_L(p)$ and its corresponding disparity of $D_R(p)$ differ by more than 1 pixel. Invalid pixels are then filled with the minimum disparity between their closest consistent pixels in the left and right direction. Weighted and typical median filtering are applied next to smooth the filled regions and remove spikes.

Algorithm: Guided Filter. Input: I, C(p,d). Parameters: r, ε.

1: $mean_I = f_{mean}(I)$; $mean_p = f_{mean}(p)$;
   $corr_I = f_{mean}(I .* I)$; $corr_{Ip} = f_{mean}(I .* p)$;
2: $var_I = corr_I - mean_I .* mean_I$; $cov_{Ip} = corr_{Ip} - mean_I .* mean_p$;
3: $a = cov_{Ip} / (var_I + \varepsilon)$; $b = mean_p - a .* mean_I$;
4: $mean_a = f_{mean}(a)$; $mean_b = f_{mean}(b)$;
5: $q = mean_a .* I + mean_b$;

Fig. 2. Pseudocode of the Guided Image Filter [10].

III. Related Work

In recent years, a fair amount of work has been carried out on real-time hardware implementations of local stereo matching algorithms (e.g. [*13*]); a thorough review is presented in [*3*]. The majority of these implementations have adopted simple fixed support and multiple window methods, therefore trading accuracy for speed. High matching accuracy though is of foremost importance in many of today's embedded vision applications. As such, a few attempts have been made recently directed towards improving the matching accuracy, either by combining different stereo algorithms together, or by implementing modified versions of SGM and ADSW algorithms. The hardware implementation in [*14*] performs a modified version of the Census transform in both the intensity and gradient images, in combination with the SAD correlation metric. An FPGA implementation of a stereo algorithm based on the neural network and Disparity Space Image (DSI) data structure is introduced in [*15*]. The real-time FPGA-based stereo matching design presented in [*16*] combines the mini-Census transform and the Cross-based cost aggregation. SGM-based stereo matching systems have been introduced in [*5*], [*6*] and implemented on FPGAs and a hybrid FPGA/RISC architecture, respectively. The technical details/parameters of the different implementations are summarized in Table I.

The works that are closely related to ours in terms of the matching algorithm are the works in [*7*], [*8*], [*9*], which implement ADSW-based algorithms. [*7*] proposed the VLSI design of a hardware-friendly ADSW algorithm that adopted the mini-Census transform to improve the accuracy and robustness to radiometric distortion. [*8*] proposed the implementation of a complete stereo vision system, which incorporates an ADSW algorithm and also integrates pre- and post- processing units. Finally, a hardware-oriented stereo matching system based on the adaptive Census transform is presented in [*9*]. The aforementioned high-quality ADSW-based systems follow a similar algorithm-to-hardware mapping methodology. That is, a complex, but accurate, algorithm is adapted for dedicated hardware implementation through a series of algorithmic modifications/approximations. In most cases, however, these implementations scarify part of the accuracy; quality reduction compared to the original implementation of the ADSW approach in [*4*] is ~ 4-5%.

In contrast, the proposed stereo matching architecture implements the ADSW aggregation step in a different way; by smoothing the SCV with an edge-preserving filter, the GIF. Due to this type of filter, and its optimized hardware design presented in this work, the proposed architecture pushes further the accuracy limits of hardware-based stereo matching systems, while it also achieves real-time frame-rates for HD images.

IV. Proposed Hardware Implementation

In this section, we first describe how the GIF, the core element of the proposed stereo matching architecture, is implemented in hardware efficiently in a way that its logic resources are independent of the kernel radius *r*. We then describe the architecture of the proposed GIF-based Stereo Matcher (GIF-SM) and its individual components.

A. Hardware Implementation of the Guided Image Filter

Besides the high quality obtained by smoothing the SCV with the GIF, this type of filter can have an efficient dedicated hardware design, as the basic operation involved is the mean filter with windows of radius *r*. The mean intensity of pixels over rectangular windows in the image can be implemented in a fast way using the integral image technique. However, this technique requires huge amount of memory, especially for high-resolution images. Therefore, we instead followed a variant of the approach in [*17*], to implement a custom mean filter design that consumes compact hardware resources. The main idea is to maintain a sum for each column in the image to be filtered. Each column sum accumulates 2r+1 pixels, while the window sum is computed by adding 2r+1 adjacent column sums. While filtering the image, the window sum is updated using the two-step approach illustrated in Fig. 3 (a). When the window is moved to the right from one pixel to the next, the column sum to the right of the window is yet to be computed for the current row, so it is centered one row above. Therefore, the first step consists of updating the column sum to the right of the window, by subtracting its topmost *old pixel* and adding one *new pixel* below it. The second step moves the window to the right and updates the window sum by subtracting its leftmost column sum (*old column sum*), and adding the updated column sum computed in step 1 (*new column sum*).

The mean filtering process is implemented in hardware with simple arithmetic operations (addition, subtraction and fixed-point multiplication) and a series of read/write operations to a memory buffer (stores the column sums) using the architecture shown in Fig. 3 (b). The mean filter architecture receives the new pixel and the old pixel, and outputs the mean corresponding to the window being filtered. Initially, the column sum yet to be updated is read from the column sum memory (its size depends on the image width), and once updated (after adding and subtracting the new and old pixels, respectively), it is written to the memory at the same address (read operation performed one clock earlier to maintain pipeline consistency). The window sum is computed by adding and subtracting the updated and old column sums, respectively, from the content of window sum register. However, we avoid fetching the old column sum from the memory, as we aimed to make the architecture flexible for both ASICs and FPGAs

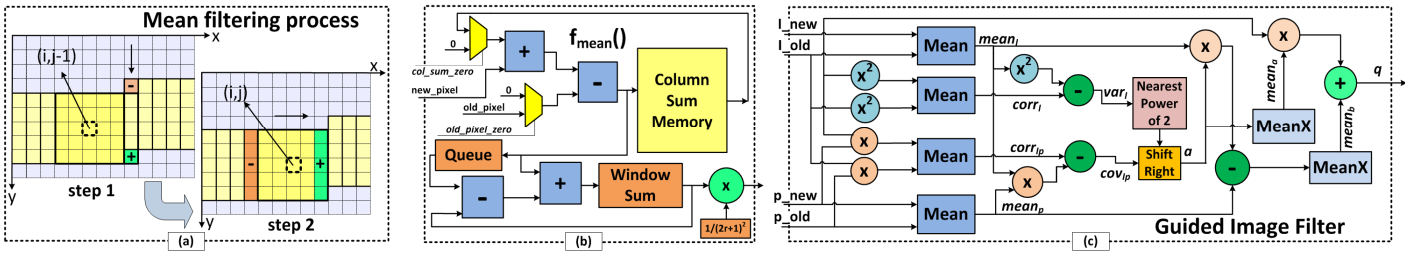

Fig. 3. (a) Mean Filtering Process. (b) Mean Filter Hardware Architecture. (c) Guided Image Filter Hardware Architecture.

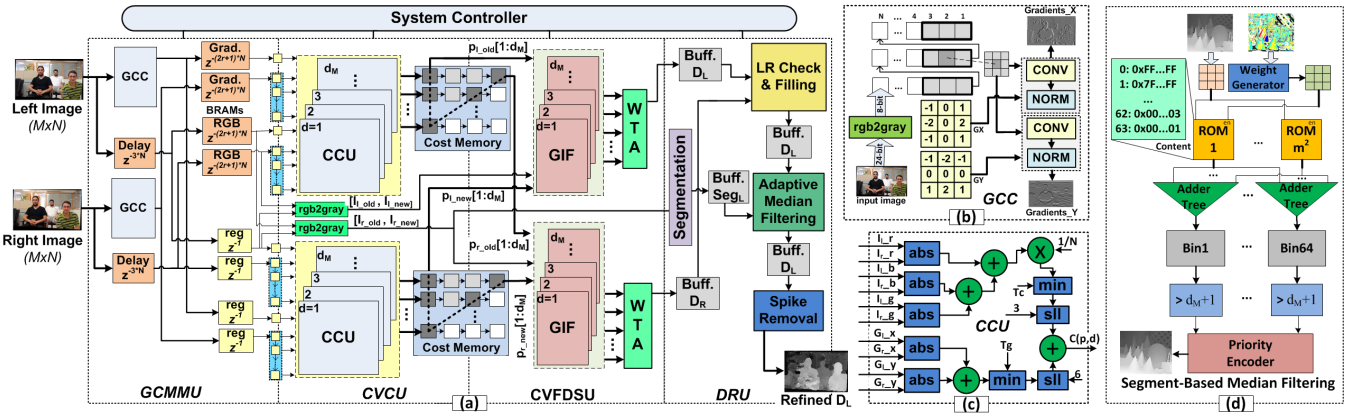

Fig. 4. (a) Architecture of the proposed GIF-based Stereo Matcher. (b) Gradients Computation Core. (c) Cost Computation Unit. (d) Segment-based Adaptive Median Filtering Hardware Architecture.

supporting dual-ported BRAMs (2 ports already used to update the new column sum). Access to the old column sums is instead obtained through a shift register (queue) with size 2r+1 (an old column sum at cycle t is a delayed version of the new column sum at cycle t-2r-1). The final mean value is computed by multiplying the window sum with $1/(2r+1)^2$.

The architecture of the GIF is depicted in Fig. 3 (c). It receives two pixels from the reference image (used as guidance image) and two from the slice of the SCV to be smoothed. The architecture consists of four mean filters that compute the values of $mean_I$, $mean_p$, $corr_I$ and $corr_{Ip}$. The remaining values of the algorithm shown in Fig. 2 are computed using a set of arithmetic units (fixed-point multipliers, adders/subtractors). The complex division operation in step 3 of the algorithm is avoided by approximating the denominator by the nearest power of 2, thus replacing the division with a right shifter. We also avoid finding the mean values of *a* and *b* using the mean filter architecture of Fig. 3(b), as this would require high memory resources in the stereo matcher (for each disparity level and for each column: a column sum and 2r+1 pixel values need to be stored). It was found that by only computing the mean over the x direction (using an accumulator and a FIFO queue) rather than on a rectangular window, it reduces the quality by less than 0.5 %, but it also eliminates the need to store the aforementioned values in BRAMs or shift registers.

B. Proposed GIF-based Stereo Matcher (GIF-SM)

The simplicity of the GIF architecture makes the stereo matching process independent of the match window size. Therefore, stereo matching can now rely on pixel-based operations. In particular, only two pixels (the *new pixel* and *old pixel*) need to be processed every clock cycle. Hence, the hardware complexity can be reduced or a higher level of parallelism can be exploited to enable higher frame rates. Moreover, the memory buffers that store the RGB and gradient values can be implemented not only using shift registers, but also using FIFO/BRAM memories. Thus, the buffer types can be selected as per the design effort and application demands.

The proposed GIF-based Stereo Matching architecture shown in Fig. 4 (a) is decomposed into four major hardware units: the *Gradients Computation & Memory Management Unit (GCMMU)*, the *Cost Volume Construction Unit (CVCU)*, the *Cost Volume Filtering & Disparity Selection Unit (CVFDSU)* and the *Disparity Refinement Unit (DRU)*. These hardware units are fully pipelined in order to obtain best processing throughput. They are also synchronized with the input pixel rate and process pixels in scanline order, so that to comply with existing pixel feeding standards. Fig. 4 (a) shows a block diagram of the architecture and the data flow between units. All processing units are controlled by the system controller.

The GCMMU utilizes two Gradient Computation Cores (GCC) that calculate the gradients of the input stereo images. While the original algorithm in [*11*] uses only the gradients in x direction in the final SCV function, this work utilizes the gradients in y direction as well, as it was found to yield better quality results. Therefore, the GCCs implement the Sobel operator to calculate the gradient values in both directions using the architecture shown in Fig. 4(b). The architecture performs convolution of 3x3 windows (fetched from a scanline buffer) with the Sobel kernels using two convolution units (CONV), and normalizes the result in the range [0 255] (NORM units). The GCMMU is also responsible for buffering the computed gradients and input color values, and providing them in a synchronized way to the CVCU. It stores 2r + 1 lines of both RGB and gradient data in BRAMs working in read-first mode; this allows access on both the new and old pixels. The new pixels (BRAM inputs) are synchronized with the old pixels (BRAM outputs) by introducing one cycle delay through register buffers. Finally, the RGB and gradient values of the $2d_M$ most recently fetched old and new pixels in the target image (right) are stored in serial-in parallel-load shift registers. This allows the CVCU to exploit disparity level parallelism.

The CVCU employs a cascade of Cost Computation Units (CCUs) that calculate the pixel-wise costs between the new and old pixels in the left image and their $2d_M$ corresponding pixels in the right image. The architecture of a CCU is shown in Fig.4(c). It consists of absolute difference units, adders and comparators that calculate the truncated color and gradient costs, which are then summed up to compute the overall cost. However, prior to the summation, the truncated color and gradient costs are multiplied by constant values, in order to balance their influence in the overall cost. The constants are selected to be powers of 2 to replace multiplication with shifting. The final pixel costs calculated by using the left image as reference are stored in cost memory buffers, and are reused for the computation of the right disparity map by the CVFDSU.

The CVFDSU employs two series of GIFs; one working on the vertical and one on the diagonal direction on the cost

memories, to smooth the SCVs corresponding to the left and right disparity maps, respectively. Smoothing is performed using the architecture of the GIF shown in Fig. 3 (c). The CVFDSU also incorporates 2 WTA units (composed of comparators organized in tree structures) that select the disparities with the minimum costs.

The DRU implements the L-R check and filling approach (Section II) through a set of comparators and priority encoders that locate the nearest valid disparities in the left and right direction of the invalid pixels. The adaptive median filter is implemented by extending the approach in [*18*], which presents a median filter architecture based on cumulative histograms, to the case of adaptive weights. The histogram-based median filter architecture shown in Fig. 4(d) has been designed in a parallel manner, allowing the computation of the median value for a window of size mxm in a single clock cycle. Its major hardware units include ROM memories, adder-trees, comparators and a priority encoder that locates the median value among the totality of the bins. We used binary weights generated by image segmentation ("one" if a disparity value lies in the same segment with the central pixel of the window, "zero" otherwise), instead of using the adaptive weights from [*11*]. This approach is not only hardware friendly, but also preserves object borders since pixels lying in the same segment are more likely to lie at the same disparity. A simple method that partitions the disparity image into segments based on thresholding was utilized. The binary weights are used as the enable signals on the ROMs. The same structure of the architecture in Fig. 4(d) is also used for implementing a typical median filter (spike removal) by setting all enable signals to 1.

V. FPGA Implementation Results

A. Experimental Platform and Methodology

The proposed system architecture has been implemented on the Inrevium's Kintex-7 FPGA Display Kit [*19*], which is equipped with a Xilinx Kintex-7 FPGA (XC7K325T-FFG900) [*20*]. We used a custom-built stereo camera system, which was directly interfaced to the FPGA board through capturing FMC daughter cards [*19*]. The entire setup was built on 720p@60Hz HD video support. The captured stereo video sequences were rectified (through a custom hardware stereo image rectification unit that follows the architecture presented in [*12*]), and used as input data to the system architecture shown in Fig. 4(a). We configured the system architecture to receive stereo video sequences synchronized with the HDMI pixel sampling clock. The refined disparity maps were also synchronized with the pixel clock, and directed to an HDMI-compatible monitor. The system diagram of the prototype FPGA implementation is given in Fig. 5 (a). The different system parameters were set to constant values: $\{r, \varepsilon, T_c, T_g\}=\{3, 0, 7, 2\}$. These values were found empirically and were used throughout our experiments.

B. Disparity Map Quality Evaluation

We used the Middlebury benchmark dataset [*2*] to evaluate the quality of the resulting disparity maps. We measured the percentage of bad matching pixels relative to the ground truth disparity maps of four pairs of test images in the dataset. The benchmark disparity maps and those produced by the proposed system architecture are shown in Fig. 5 (b). Quantitative evaluation results are listed in Table I (columns 2-5), which

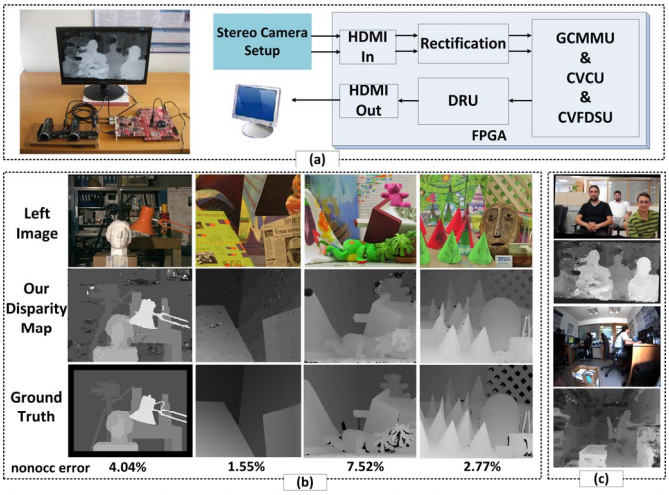

Fig. 5. (a) Experimental Testbed. (b) Benchmark Stereo Images. (c) Real-world Stereo Pairs Captured in the Lab.

also provides a comparison with related work. As it can be observed, the proposed architecture generates high quality results, even at regions with low texture and close to object boundaries; this is evidenced by obtaining the lowest error rates among related hardware implementations at *disc* regions. Most importantly, among the implementations for which an overall percentage of bad matching pixels is provided, the proposed system obtains the lowest error rate. Besides the Middlebury benchmark images, the proposed architecture is able to deal with real-world scenes as well, producing detailed and accurate disparity maps with clean object boundaries (see Fig. 5 (c)).

We have also investigated how the resulting disparity maps are compared with those generated by the original GIF-based algorithm presented in [*11*]. As it can be observed from Table I, the distance in quality between the work in [*11*] and the proposed architecture is only 1.14% on average. Moreover, the algorithm in [*11*] implements a color version of the GIF, which in general yields better accuracy. Therefore, the distance in quality can be further reduced if the proposed GIF-SM is extended to support color guidance images as well. Finally, it is worth noting that the matching accuracy of the proposed GIF-based stereo matcher approaches the quality of the original ADSW algorithm in [*4*] (6.69% vs. 6.67%), while the ADSW dedicated hardware implementations in [*8*], [*9*] exhibit a quality reduction compared to [*4*] of ~4.84% and ~4.68%, respectively. This evidences the superiority of the proposed hardware-based GIF-SM in maintaining the matching accuracy of the original ADSW algorithm, thanks to the integration of the GIF.

C. Processing Speed

We measured the processing speed of the proposed GIF-SM in frames processed per second (fps) and in Million Disparity Estimations per second (MDE/s), a metric that also takes into account the number of pixels and the disparity range ($MDE/s = M \cdot N \cdot D \cdot fps$). The FPGA prototype of the architecture is able to process 720p (1280x720) HD video at 60 fps, with a pixel clock rate of ~74.25MHz. When considering the post place-and-route frequency (103 MHz) provided by the Xilinx ISE Design tool, the maximum throughput of the architecture is expected to reach ~83 fps. In general, the architecture presents good scalability with respect to the frame rate and image size, as it is intensively pipelined and

TABLE I. QUALITY AND SPEED COMPARISON WITH RELATED WORK

Work	Average Error Rates				Image size	D	Speed (fps)	MDE/s (10^6)	Platform
	nonocc[1]	all[2]	disc[3]	overall[4]					
Baha [15]	7.16	11.5	n.a.	n.a.	450x375	50	46	388	FPGA
Zhang [16]	4.41	7.41	12.8	8.20	1024x768	64	60	3019	FPGA
Jin [13]	8.31	13.9	27.2	16.5	640x480	64	230	4522	FPGA
Ambrosch [14]	5.82	9.37	22.2	12.5	750x400	60	60	1080	FPGA
Chang [7]	n.a.	6.81	n.a.	n.a.	352x288	64	42.5	276	ASIC
Ding [8]	7.41	11.9	15.6	11.5	640x480	60	51	940	FPGA
Perri [9]	4.39	10.09	19.57	11.35	640x480	60	45	829	FPGA
Gehrig [5]	n.a.	8.13	n.a.	n.a.	340x200	64	27	118	FPGA
Banz [6]	8.43	n.a.	n.a.	n.a.	640x480	128	30	1179	FPGA
ADSW [4]	3.48	6.53	9.98	6.67	320x240	30	0.01	0.0263	n.a.
Hosni [11]	2.65	5.57	8.43	5.55	640x480	26	25	200	CPU/GPU
Proposed	3.97	6.80	9.30	6.69	1280x720	64	60	3538	FPGA

[1]all points except for occluded areas, [2] all points including half-occluded regions, [3] only points along depth discontinuities. [4] average error rate at *nonocc*, *all* and *disc* regions.

synchronized with the pixel clock rate. Therefore, it can easily be configured to process Full HD 1080p video (pixel clock = 145 MHz) by simply increasing the number of pipeline stages.

Table I (columns 6-10) presents a comparison between the developed FPGA prototype and related work in terms of processing speed. Obviously, the proposed system is among the fastest implementations when considering the MDE/s metric, outperforming all dedicated hardware implementations targeting high-quality disparity map estimation. Only the work in [*13*] obtains higher MDE/s. However, this implementation is not competitive in terms of quality, mainly due to the use of a fixed support algorithm and its associated problems discussed in Section II. Finally, the proposed system has a speedup improvement of ~17x when compared to the original GIF-based algorithm [*11*] (implemented on a hybrid CPU/GPU system), thus justifying the very small quality reduction. Conclusively, the obtained speed/quality results indicate that the proposed system has high potential for embedded vision applications requiring fast and accurate stereo matching.

D. FPGA Synthesis Results

The FPGA utilization figures are summarized in Table II. The complete FPGA prototype utilizes ~35% of the available Slice Registers and ~65% of the available Slice LUTs. It also utilizes ~87% of the DSP units and ~60% of the block memories. The required DSPs are currently limiting resources for a larger scale implementation. However, their usage can be reduced by mapping some of the DSPs on LUT units. Table II also provides the hardware overheads of the proposed GIF-SM (w/o the DRU) for different values of d_M = {16, 32, 64}, in order to investigate how it scales with respect to d_M. It can be observed that the hardware implementation complexity of the GIF-SM scales linearly with the number of disparity levels.

VI. CONCLUSION AND FUTURE WORK

This paper presented a high-quality real-time hardware stereo matcher based on the GIF. A compact and efficient hardware design of the filter was illustrated, which was then utilized to develop a parallel and scalable ADSW stereo matcher. The GIF design reduces the hardware complexity of the ADSW cost aggregation, enabling real-time frame-rates on HD images. Moreover, the GIF's inherent edge-preserving nature allows for improved matching accuracy when compared to existing state-of-the-art hardware stereo matching systems. As an immediate follow up of this work, we intend to extend the GIF architecture to support color guidance images. We also plan to perform a detailed design space exploration of the stereo matcher hardware resources with respect to the various system parameters (radius kernel, image size, etc).

TABLE II. KINTEX-7 UTILIZATION REPORT

Design Unit	Slice LUTs		Slice Registers		DSP48E		BRAM	
	Total=203800		Total=407600		Total=840		Total=445	
Rectification	3473	1.7%	6146	1.5%	0	0	24	5.4%
GIF-SM 16	24262	11.9%	27247	6.7%	238	28.3%	99	22.2%
GIF-SM 32	42464	20.8%	52217	12.8%	446	53.1%	147	33%
GIF-SM 64	88791	43.6%	117260	28.8%	734	87.4%	243	54.6%
DRU	39554	19.4%	17507	4.3%	0	0	0	0
Prototype[1]	132020	64.8%	140967	34.6%	738	87.9%	267	60%

[1] Prototype system integrating the rectification unit, the GIF-SM 64 and the DRU.

REFERENCES

[1] B. Cyganek and J. P. Siebert, *Introduction to 3D Computer Vision Techniques and Algorithms*.: Wiley, John & Sons, 2009.

[2] D. Szeliski and R. Scharstein, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *Inter. J. of Comput. Vision*, vol. 47, pp. 7-42, 2002.

[3] L. Nalpantidis, G. C. Sirakoulis, and A. Gasteratos, "Review of stereo vision algorithms: from software to hardware," *Inter. J. of Optomechatronics*, vol. 2, no. 4, pp. 435-462, 2008.

[4] K.-J. Yoon and I.-S. Kweon, "Adaptive support-weight approach for correspondence search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 4, pp. 650-656, Apr. 2006.

[5] S. K. Gehrig, F. Eberli, and T. Meyer, "A Real-Time Low-Power Stereo Vision Engine Using Semi-Global Matching," *Computer Vision Systems, Lecture Notes in Computer Science*, vol. 5815, pp. 134-143, 2009.

[6] 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 *ICSAMOS*, 2010, pp. 93-101.

[7] N. Chang, T. Tsai, B. Hsu, Y. Chen, and T. Chang, "Algorithm and Architecture of Disparity Estimation With Mini-Census Adaptive Support Weight," *IEEE Trans. on CSVT*, vol. 20, no. 6, pp. 792-805, 2010.

[8] J. Ding et al., "Real-time stereo vision system using adaptive weight cost aggregation approach," *EURASIP JIVP*, 2011.

[9] S. Perri, P. Corsonello, and G. Cocorullo, "Adaptive Census Transform: A novel hardware-oriented stereovision algorithm," *Computer Vision and Image Understanding*, vol. 117, no. 1, pp. 29-41, January 2013.

[10] Kaiming He, Jian Sun, and Xiaoou Tang, "Guided Image Filtering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 6, pp. 1397-1409, 2013.

[11] A. Hosni, M. Bleyer, C. Rhemann, M. Gelautz, and C. Rother, "Real-time local stereo matching using guided image filtering," in *IEEE Int. Conf. on Multimedia and Expo (ICME)*, 2011, pp. 1-6.

[12] H.-S. Son, K.-r. Bae, S.-H. Ok, Y.-H. Lee, and B. Moon, "A Rectification Hardware Architecture for an Adaptive Multiple-Baseline Stereo Vision System," in *Communication and Networking*.: Springer Berlin Heidelberg, 2012, pp. 147-155.

[13] S. Jin et al., "FPGA design and implementation of a real-time stereo vision system," *IEEE Trans. on CSVT*, vol. 20, no. 1, pp. 15-26, 2010.

[14] K. Ambrosch and W. Kubinger, "Accurate hardware-based stereo vision," *Comput. Vis. Image Und.*, vol. 114, no. 11, pp. 1303-1316, Nov. 2010.

[15] N. Baha and S. Larabi, "Accurate real-time neural disparity MAP estimation with FPGA," *Pattern Recognition*, vol. 45, no. 3, pp. 1195-1204, March 2012.

[16] L. Zhang et al., "Real-Time High-Definition Stereo Matching on FPGA," in *FPGA'11*, Monterey, CA, USA, 2011, pp. 55-64.

[17] S. Perreault and P. Hebert, "Median Filtering in Constant Time," *IEEE Trans. on Image Process.*, vol. 16, no. 9, pp. 2389-2394, 2007.

[18] S. A. Fahmy, P. Y. K. Cheung, and W. Luk, "Novel FPGA-Based Implementation of Median and Weighted Median Filters for Image Processing," in *FPL2005*, Tampere, Finland, August 2005, pp. 142–147.

[19] TOKYO ELECTRON DEVICE LIMITED. (2013, August) Kintex-7 FPGA Display Kit. [Online]. http://solutions.inrevium.com/products/kits/consumer/tb-7k-acdc.html

[20] Xilinx Inc. (2013, August) Kintex-7 FPGA Family. [Online]. http://www.xilinx.com/products/silicon-devices/fpga/kintex-7/