A Novel Hardware-Oriented Stereo Matching Algorithm and Its Architecture Design in FPGA

Yanzhe Li[1](✉), Kai Huang[1], and Luc Claesen[2]

[1] Institute of VLSI Design, Zhejiang University, Hangzhou, China
{liyz,huangk}@vlsi.zju.edu.cn
[2] Engineering Technology - Electronics-ICT Department,
Hasselt University, 3590 Diepenbeek, Belgium
luc.claesen@uhasselt.be

Abstract. Stereo matching is a crucial step to extract depth information from stereo images. However, it is still challenging to achieve good performance in both speed and accuracy for various stereo vision applications. In this contribution, a hardware-compatible stereo matching algorithm is proposed and its associated hardware implementation is also presented. The proposed algorithm can produce high-quality disparity maps with the combined use of the mini-census transform, segmentation-based adaptive support weight and effective refinement. Moreover, the proposed architecture is optimized as a fully pipelined and scalable hardware system. Implemented on an Altera Stratix-IV FPGA board, it can achieve 65 frames per second (fps) for 1024×768 stereo images and a 64 pixel disparity range. The proposed system is evaluated on the Middlebury benchmark and the average error rate is 6.56%. The experimental results indicate that the accuracy is competitive with some state-of-the-art software implementations.

Keywords: Stereo matching · Hardware implementation · Real-time · High-quality

1 Introduction

Stereo vision is one of the most active research topics in computer vision and it is widely used in many applications. Just recently, three dimensional television (3DTV) and virtual reality gaming have become popular. They provide the audience with a greater sense of presence in a computer-generated environment. However, the requirement to wear additional eyeglasses is usually perceived uncomfortable. In order to overcome the problem, autostereoscopic displays are utilized to support glasses-free 3D depth perception. In this way, multiple views are shown simultaneously so that the audience always sees a stereo pair from predefined viewpoints regardless of his position. These multiple views need to be generated by using depth-image based rendering (DIBR) from the original views and their corresponding depth maps. Each depth map gives information about

T. Hollstein et al. (Eds.): VLSI-SoC 2016, IFIP AICT 508, pp. 213–232, 2017.
DOI: 10.1007/978-3-319-67104-8_11

the distance between the camera and the objects in the 3D scene. Here the depth maps can be extracted in stereo vision systems. Real-time depth image generation is also important in future advanced driver assistance systems (ADAS) as well as self driving cars. Two or more cameras can assist in the distance calculation of other traffic, vehicles and objects while driving. In comparison to "Time of Flight" cameras, stereo cameras can be used over much larger distances, and can be used under various intensities of the environment (e.g. sun light).

Stereo matching, which is treated as the key operation in a stereo vision system, takes a pair of rectified images, estimates the movement of each pixel between two images and displays the associated movement in a disparity map. The depth of a pixel is inversely proportional to the disparity of this pixel. As a result, stereo matching is a complicated and time-consuming procedure. Considering that many applications often require high performance and real-time processing speed, it is difficult for software implementations of stereo matching algorithms on a CPU to meet these constraints. In this condition, hardware acceleration of stereo matching algorithms is inevitable and it has been done extensively using DSPs, GPUs and dedicated hardware. However, DSPs are limited by the computational ability and fail to support real-time processing; while GPUs always result in excessive power consumption for embedded applications. In contrary, the dedicated hardware approaches using FPGAs and ASICs can provide a balance between computational power and energy efficiency.

In the presented research work, the mini-census and segmentation-based ADSW algorithms are combined to achieve a high matching accuracy in depth discontinuity regions. Different from many other hardware designs that lack refinement, a disparity refinement with segmentation information is presented. This refinement step can significantly improve the quality of initial disparity maps in textureless and occluded regions. Moreover, a fully pipelined and scalable architecture is implemented based on the proposed algorithm. In order to make a tradeoff between accuracy and speed, some techniques such as a simplified weight function and an adaptive window size are applied. A prototype of the proposed hardware system is built on an Altera FPGA board, which achieves 65 fps for 1024×768 stereo images and a 64 pixel disparity range. The system is evaluated on the Middlebury benchmark and the visual satisfactory results are derived. The experimental results indicate that the proposed system has the top-performing processing ability and its accuracy is competitive with state-of-the-art software implementations.

In the rest of this contribution, Sect. 2 reviews the background of stereo matching algorithms and some related work. Section 3 presents the proposed algorithm. In Sect. 4, the hardware implementation based on the proposed algorithm is described. Section 5 presents experimental results and compares them with previous methods. Finally, Sect. 6 concludes the contribution.

2 Background and Related Work

2.1 Stereo Matching Background

Stereo matching algorithms aim to establish correspondence between a pair of images. This requires a pixel-by-pixel search through the whole image, consuming a large amount of computation power. To solve the problem, camera calibration and image rectification are used as preprocessing steps for most stereo matching algorithms. The preprocessing steps project each image to a common image plane and align each epipolar line to a common axis. In this way, stereo matching is reduced to a 1D search problem along the same horizontal scanline of the image pair.

Given two calibrated and rectified images, stereo matching can be addressed by searching for the corresponding pixel in the right image for each pixel in the left. To make the results more reliable, a support region is built for each pixel and the matching process is carried out over these regions instead of pixel by pixel. For a pixel $P(x, y)$ in the left image, its corresponding pixel $P'(x + d, y)$ is searched on the same horizontal line in the right image, where $0 \leq d < D_{max}$, D_{max} is the largest search distance and d is called the disparity. The matching costs are calculated for each pixel pair in the support regions and then aggregated. The smaller the aggregated matching cost is, the more similar the support regions are. Thus, the corresponding pixel is defined as the anchor pixel in the support region with the minimal aggregated matching cost.

2.2 Related Work

Nowadays, stereo matching algorithms can be divided into two groups: local approaches and global approaches [1]. Since local approaches only utilize local information, the accuracy is usually not sufficient in textureless and occluded regions. On the other hand, while global approaches can show better results, they are not yet suitable for real-time implementations due to their high computation complexity [2].

Global approaches usually compute disparities based on a global cost optimization. Dynamic programming (DP) [3] is a technique that optimizes disparity maps on a scanline in an efficient manner. Belief propagation (BP) [4] is a global approach that has attracted much attention. It gathers information from neighboring pixels and incorporates the information to update a smoothness term, then iteratively optimizes the smoothness term to achieve global energy minimization. Another popular technique explored by global approaches is graph cut [5]. Its energy function presents three terms: the data term that represents the difference between two corresponding pixels, the smoothness term that makes neighboring pixels tend to have similar disparities and the occlusion term, which imposes a penalty for making a pixel occluded. Although global approaches provide impressive accuracy results, the real-time implementations for high resolution images are challenging due to their computational complexity.

Another type of stereo matching algorithms is the class of local approaches, which compute disparities at a given point within a finite window. Early works on local approaches evaluate the impact of different similarity measures [2]. Common window-based matching costs include the sum of absolute or squared differences (SAD/SSD), normalized cross correlation (NCC), census transforms and mutual information [6]. Another important research topic that has been studied is that of the support regions. The early conventional approach is to use fixed-size square windows, which is easy to implement but suffers from severe artifacts. To remedy this, variable window size [7] is developed and it can improve the disparity quality in textureless regions. A recent development with promising results is to adapt the support weights in fixed-size windows. An adaptive support weight (ADSW) algorithm [8] is proposed, which assigns different weights to the pixels in a support window based on the proximity and color distances to the center pixel. A segment support algorithm [9] assigns fixed weights to the pixels in the same segment as the center pixel is in, and assigns weights to the pixels outside the center pixels segment according to the color similarity between the outside pixel and the center pixel. The disparity quality of [8,9] is comparable to some of the complex global algorithms.

Compared to global approaches, local approaches are more suitable for dedicated hardware implementation such as FPGAs and ASICs because of their low computation complexity and storage requirement. A segmentation-based design with adaptive support weight (ADSW) has been implemented on FPGAs [10]. Their proposed design can achieve 30 fps for 640 $\times$ 480 images using a disparity range of 64 pixels. This design is inspired by the algorithm in [9], which used to be the best local method on the Middlebury benchmark [11]. However, the performance of the design is restricted by the small fixed window size. In [12], a hardware solution provides high-quality disparity results in ASICs based on the mini-census adaptive support weight (MCADSW) method. But this solution only targets low resolution images. Its performance drops to 6 fps for 1024 $\times$ 768 images and a 64 pixel disparity range. In [13], an algorithm is proposed to achieve high accuracy based on mini-census and variable-cross methods and a fully pipelined architecture is presented for real-time processing. The design can process 1024 $\times$ 768 images with a disparity range of 64 pixels in 60 fps. A. Akin proposes a hardware-oriented adaptive window size disparity estimation (AWDE) algorithm and its real-time hardware implementation [14]. It can handle 60 fps at a 1024 $\times$ 768 resolution for a 128 pixel disparity range. Although the results in [13,14] are outstanding among hardware implementations, the accuracy is not comparable to state-of-the-art software implementations.

3 Stereo Matching Algorithm

3.1 Algorithm Overview

In local stereo matching algorithms, cost calculation, cost aggregation, disparity selection and disparity refinement are four well-defined steps [1]. Since the proposed stereo matching algorithm belongs to local approaches, the mini-census transform is used in the cost calculation step; the segmentation-based ADSW algorithm is used in the cost aggregation step; and a tree-structure winner-takes-all (WTA) method is used in the disparity selection step. The last step, disparity refinement, consists of three stages: consistency check, disparity voting and invalid disparity inpainting. The flow of the proposed algorithm is illustrated in Fig. 1. Two images are operated and the corresponding disparity maps are generated simultaneously.

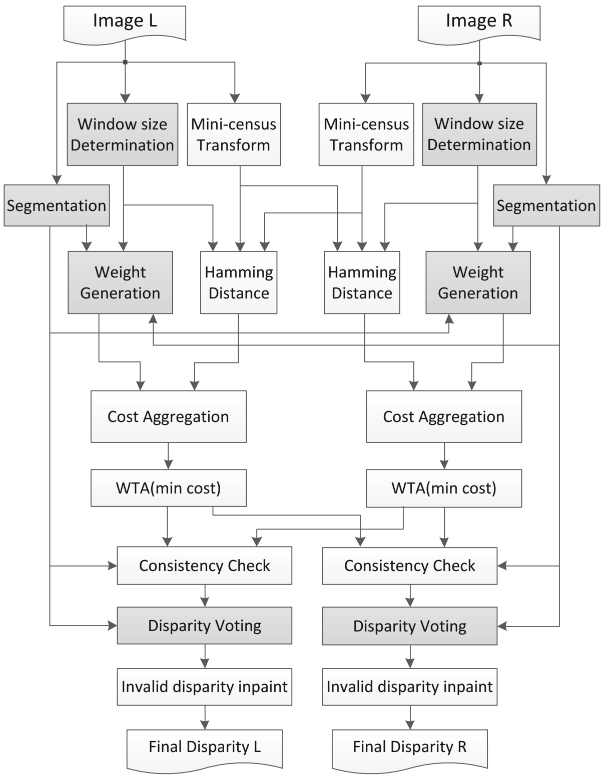

Fig. 1. Overview of the proposed algorithm.

In the cost calculation step, the mini-census transform is a hardware-friendly census transform, which makes the matching cost robust to brightness bias and exposure gain. It extracts a 6-pixel neighborhood information of the center pixel within a support window and encodes the information into a vector. If a pixels luminance is larger than the center pixels, a label 0 will be assigned to the pixel. Otherwise, a label 1 will be assigned [12]. In this way, each pixel can be represented by only 6 bits, which results in a reduction of the memory utilization due to fewer storage bits. Then the matching cost is defined as the Hamming distance between output vectors.

In the cost aggregation step, the segmentation-based ADSW algorithm employs the segmentation information within the weight cost function to increase the robustness of the matching process. Rather than only relying on colour and proximity, the use of segmentation takes the relationship between pixels and the shape of the segments into account. It assumes that each pixel on the same segment of the center pixel has a similar disparity value, and its weight is equal to the maximum value of the range [9]. The weight coefficients w_r and w_l are defined as

$$w_{r,l} = \begin{cases} 1.0 & p_i \in S_c \\ \exp(-\frac{d_c(I_{r,l}(p_i), I_{r,l}(p_c))}{\gamma_c}) & \text{otherwise} \end{cases} \tag{1}$$

where S_c is the segment of the central point, d_c is the Euclidean distance between two triplets in the CIELAB color space, and γ_c is a fixed parameter in the algorithm. The final aggregated cost is calculated by summing up all the weighted matching costs in the support windows W_r and W_l, and then normalized with the sum of weight coefficients

$$C(p_c, q_c) = \frac{\sum\limits_{p_i \in W_r, q_i \in W_l} w_r(p_i, p_c) w_l(q_i, q_c) MC(p_i, q_i)}{\sum\limits_{p_i \in W_r, q_i \in W_l} w_r(p_i, p_c) w_l(q_i, q_c)} \tag{2}$$

where p_c and q_c are the central points of W_r and W_l, respectively. The cost aggregation step is executed for all disparity levels and a number of aggregated costs are produced.

In the disparity selection step, a tree-structure WTA method is used to pick the disparity with the minimum aggregated cost, as depicted in Fig. 2. The aggregated costs for the whole disparity range are arranged into groups. Here, the disparity range and the size of a group are defined as 64 and 4, respectively. For each group, the smallest value and the corresponding position are selected and stored. Then after several times iteration, the minimum value among groups is finally detected; its position is selected as the disparity result.

The advantage of using a tree-structure method is not only to reduce the complexity of the search operation. It also fits the dataflow within FPGAs very well. Thus, it can be highly pipelined and the throughput increased up to one disparity range per clock cycle.

In the disparity refinement step, the initial disparity maps are operated. The consistency check is used to check whether disparity maps are valid or not. With the help of segmentation information, a left-to-right consistency check is expressed as

$$V_p = (d_p(x, y) == d'_p(x - d_p(x, y), y)) \; \& \; (S_p == S'_p). \tag{3}$$

In the process of left-to-right consistency check, for each pixel p in the left image, the corresponding pixel p' in the right image is determined by the disparity d_p. Then the right image's disparity d'_p and segmentation S'_p will be compared to d_p and S_p of the left image, respectively. If the expression is calculated as false, the

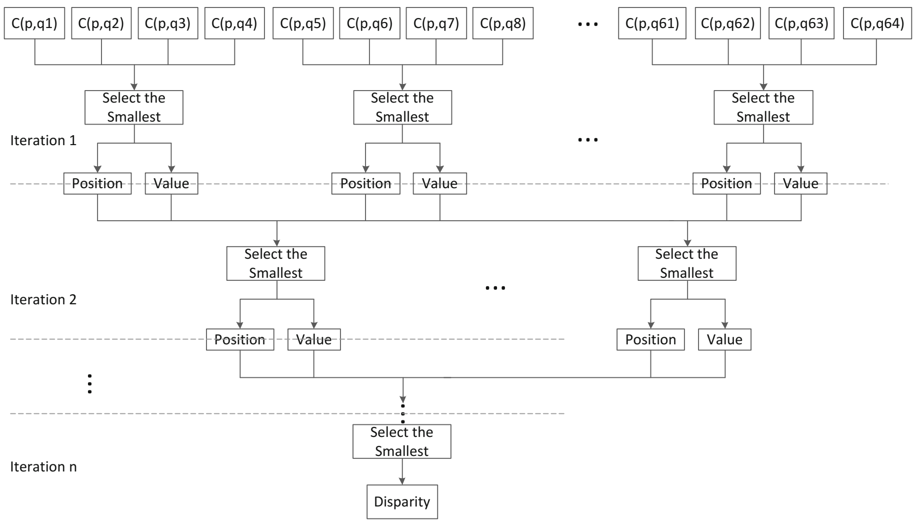

Fig. 2. Tree-structure WTA method.

left-to-right consistency check will fail and the disparity will be marked as invalid. It is noted that this process can also be utilized for a right-to-left consistency check.

After the consistency check, the disparity voting will update the center disparity based on the most frequent valid disparity in its local support window. This is because adjacent pixels that belong to the same object in an image should share the same disparity. Although the disparity voting helps to remove many invalid disparities, it will fail if the window does not contain any valid disparities. In order to address this problem, the disparity inpainting will replace the invalid disparity with the closest valid disparity on its scanline so as to get the final disparity maps. It is worthy to note that the median filtering is not used in the design because of its complicated hardware implementation but limited quality improvement.

3.2 Hardware-Oriented Optimization

To reduce the computation complexity and improve the hardware compatibility of the algorithm, some optimizations are proposed in this subsection and will be applied to the shaded blocks in Fig. 1. These hardware-oriented optimizations affect the accuracy of the final disparity maps slightly.

In the window size determination block, a method called AWDE [14] is introduced to make a tradeoff between accuracy and speed. It uses three different window sizes for different textures on the image; the window size is determined by the mean absolute deviation (MAD) of the pixel in the center of a 7×7 block, which is expressed as

$$MAD(c) = \frac{\sum\limits_{q \in N_c} |I_t(q) - I_t(c)|}{48}. \tag{4}$$

A high MAD value indicates a high texture content, while a low MAD value is a sign of a low texture content. As expressed in (5), a 7 × 7 window is used if the MAD of the center pixel is high, and a 25 × 25 window is used if the MAD is very low.

$$\text{window size} = \begin{cases} 7 \times 7 & MAD(c) > th_7 \\ 13 \times 13 & th_{13} < MAD(c) \leq th_7 \\ 25 \times 25 & MAD(c) \leq th_{13} \end{cases} \tag{5}$$

As a general rule, increasing the window size increases the hardware complexity. In order to provide constant hardware complexity over the three different window sizes, a total of 49 pixels are constantly sampled with different intervals for different window sizes. In this way, a low computation cost is required for large support window sizes.

In the segmentation block, the segmentation-based ADSW algorithm [9] uses mean shift segmentation. However, the computational complexity and memory requirements make it unsuitable for real-time applications. In our algorithm, the image is divided into segments using thresholding [10]; this method is simple and can be implemented in hardware efficiently.

In the weight generation block, in order to replace signed floating-point numbers with unsigned integers, the YUV color representation is adopted instead of the CIELAB color representation. In addition, only the luminance channel (Y) is used in the design to reduce the potential bandwidth and storage requirements. Rather than Euclidean distance, Manhattan distance is used to avoid square and square root computations. Furthermore, for the pixel whose luminance is similar to the center pixel in the support window, it should be allowed to have more influence on the final matching cost. Therefore, a scale-and-truncate approximation of the weight function is proposed, and the curve is shown in Fig. 3. As a result, the multiplication of the weight coefficients is reduced to a left shift operation.

In the disparity voting block, a local support window is applied to achieve a reliable result. A vertical-horizontal approach is used to efficiently determine the most frequent valid disparity in the window, as shown in Fig. 4. Here the numbered shaded squares indicate valid disparities, which are used to update the center disparity. First, the approach searches for the majority disparity vertically in each column. Then it searches for the majority disparity horizontally and finally selects it as the center disparity. To further reduce the computation complexity, the approach also reduces the internal bandwidth.

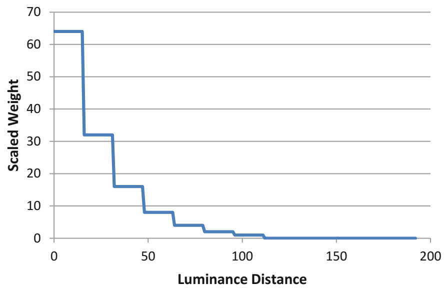

Fig. 3. Weight function.

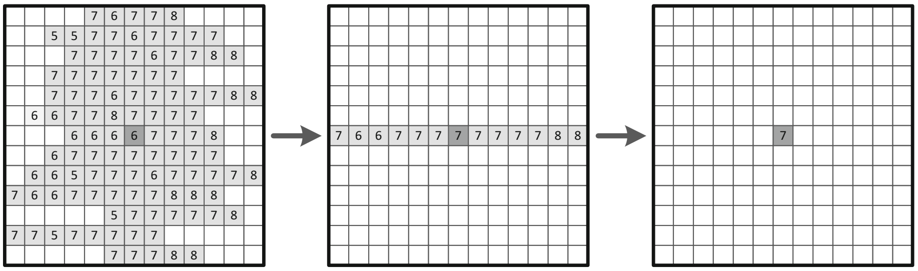

Fig. 4. The vertical-horizontal approach for disparity voting.

4 Hardware Implementation

4.1 Architecture Overview

In this section, a hardware architecture is designed based on the proposed algorithm. The whole system consists of three stages: pre-processing, stereo-matching and post-processing. First, in the pre-processing stage, pixel-based operations are performed on each pixel and the temporary results are stored into the line buffers. Then these temporary results are operated in the stereo-matching stage in order to calculate initial disparity maps. When the initial disparity maps are available, they will be refined in the post-processing stage. The top-level block diagram of the proposed hardware architecture is shown in Fig. 5; the implementation details of the three stages will be discussed in the following subsections.

One key feature of the proposed system is high processing ability. To achieve this goal, the architecture is designed to be fully pipelined without external memory limitation. All of the three stages are fully pipelined, i.e. source image pixels are fetched and operated in scanline order; initial disparity maps are generated in pipeline using a parallelism scheme; finally the disparity maps are refined in scanline order after a certain pipeline latency. External memory bandwidth is also an important limitation to the processing ability. In order to solve the problem in our design, each pixel is read only once from the external memory during

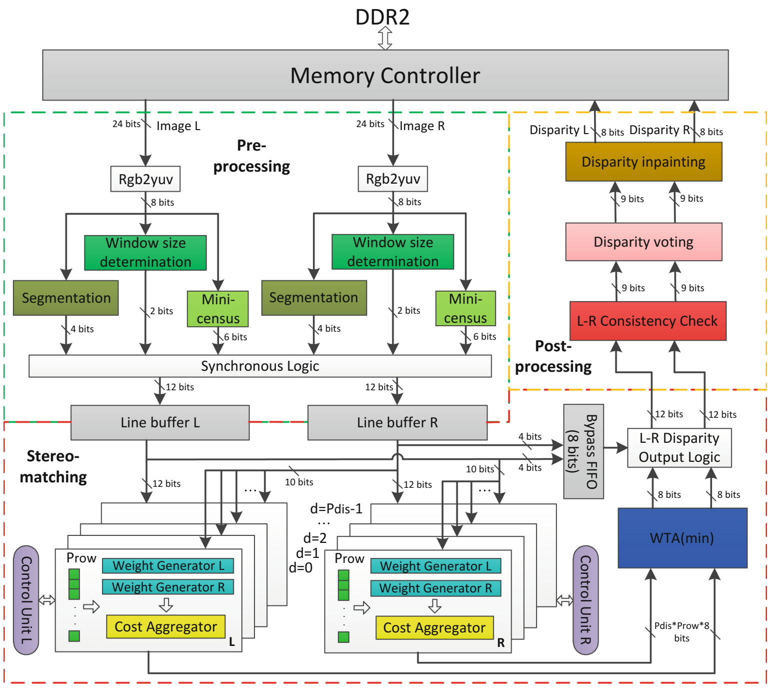

Fig. 5. Block diagram of the proposed architecture. (Color figure online)

the whole processing flow. In this case, dual-port BRAMs are used as buffers to store original pixel data and temporary intermediate results.

Another key feature of the proposed system is scalability. The design can be scaled with image resolution, disparity range and parallelism degree to achieve maximum flexibility. The image resolution is related to the user demand, while the disparity range is configured depending on the expected distance to the objects. Configuring the hardware for low image resolution and disparity range increases the processing speed. In contrast, high image resolution and disparity range lead to a high accuracy. The parallelism degree is used to indicate how many disparities are calculated in parallel, which can make a tradeoff between resource utilization and processing speed.

4.2 Pre-processing Stage

In the preprocessing stage, 24-bit source pixels of RGB will be fetched from the DDR2 memory once the system is started. The RGB pixels are treated as the input to the color space converters (rgb2yuv). Hereafter 8-bit Y values are generated from the converters and will be utilized to produce 12-bit temporary results in the three submodules.

As displayed in the green dotted box in Fig. 5, 4 bits of the 12-bit temporary result are generated in the segmentation module. In this module, the number of segments k is given as input. A segmentation label is calculated using a simple method that multiplies the Y value by the value of $k/256$. Note that k is always defined as 16 in our system. In this way, the shift operation can be exploited instead of multiplication, and the label can be expressed within 4 bits. Another 6 bits come from the mini-census transform module. Here the center pixel is compared with its surrounding 6 pixels, and a 6-bit mini-census vector is obtained as the comparison result. The last 2 bits are produced in the window size determination module. The MAD of the center pixel in a 7×7 block is calculated, and then the 2-bit window size is assigned based on the MAD value. Since all the operations in the submodules are window-based, a register matrix is employed to provide pipelined processing, as shown in Fig. 6. The whole register matrix of 7×7 is used for window size determination; the 6 green registers are used for the mini-census transform; the red one in the center is used for segmentation. The Y values will be shifted from the left to the right in the register matrix per clock cycle. Meanwhile, a total of 12 bits will be written into the line buffer as the temporary result for each pixel.

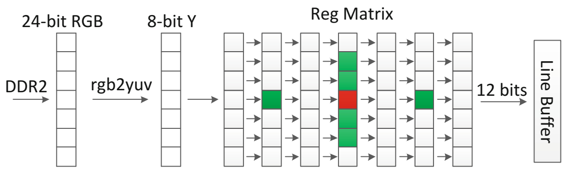

Fig. 6. Pipeline architecture in the pre-processing stage. (Color figure online)

4.3 Stereo-Matching Stage

In the stereo-matching stage, it is challenging to develop an efficient parallelism scheme for cost aggregation due to the requirement for real-time processing speed. Many hardware systems calculate all the disparities in parallel and process pixel by pixel. It is simple to implement but inefficient. In our system, a hybrid parallelism scheme [15] is adopted. It combines the row-level parallelism P_{row} with the disparity-level parallelism P_{dis}. The row-level parallelism means that P_{row} pixels in neighboring rows are processed in parallel, and the disparity-level parallelism means that P_{dis} disparities are processed for each pixel. Thus, the parallelism degree is $P_{row} \times P_{dis}$ in the proposed system. As shown in the red dotted box in Fig. 5, a total of $P_{dis} \times 2$ aggregation modules are generated to deal with different disparities for both images, while P_{row} pixels are processed in each aggregation module. Here the generate statement in Verilog is used for these two parameters to make the hardware architecture scalable.

To satisfy the row-level parallelism, P_{row} pixels along the column direction are processed in parallel. Therefore the line buffer is composed of $(P_{row}+6)$ dual port BRAMs to build a wide throughout, and the size of the register matrix in the pre-processing stage is extended to $(P_{row}+6)\times 7$. Source data and temporary results in each column of the matrix can be reused to reduce the computational requirements, because a column is usually a part of multiple horizontally overlapping windows.

In each aggregation module, the weight generators are utilized to generate weight coefficients using the 4-bit segmentation labels in the temporary results. The circuit of the weight generator is shown in Fig. 7; the look-up table (LUT) is a straightforward solution that consumes a low amount of hardware resources. Meanwhile in the cost aggregator, 49 Hamming distances are generated as matching costs between corresponding pixels in the support windows W_r and W_l. These matching costs are shifted by the corresponding weight coefficients w_r and w_l. The final aggregated cost is calculated by summing the weighted costs using a tree adder, then dividing it by the sum of weight coefficients. The architecture of the cost aggregator is illustrated in Fig. 8. Then, the disparity with the minimum aggregated cost is selected in the WTA module. In addition, the Bypass FIFO is used to store the segmentation label of each pixel for the next stage. The output of the stereo-matching stage is a data stream that consists of the initial disparity maps and their corresponding segmentation labels.

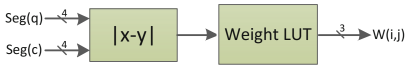

Fig. 7. Circuit of the weight generator.

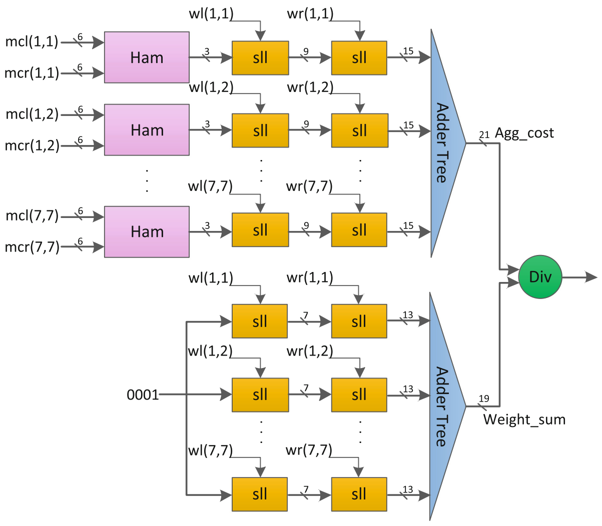

Fig. 8. Architecture of the cost aggregator.

4.4 Post-precessing Stage

In the post-precessing stage, three submodules work in pipeline to generate the final disparity maps, as shown in the yellow dotted box in Fig. 5. First, the initial disparity maps of both images are used to check the consistency of every pixel with the help of the segmentation information in the consistency check module. The module generates one more bit to label whether each disparity is valid or not.

Then in the disparity voting module, the disparities are updated in a 25 × 25 support window using the vertical-horizontal method. To enable a fully pipelined implementation, a bitwise fast voting technique [16] is applied to handle the most frequent valid disparity value. For each column, it drives each bit of the most frequent disparity independently from the other bits. In this way, the hardware cost depends on the number of the disparity bits in binary. It is noted that when counting bit votes, the valid information of each disparity must be taken into account. The architecture of bitwise fast voting for one column is shown in Fig. 9. Since the support window size is 25 × 25, the 25 most frequent disparities for the 25 columns are derived. The same technique is applied to the 25 derived disparities and finally the most frequent valid disparity in the support window is picked as the center disparity.

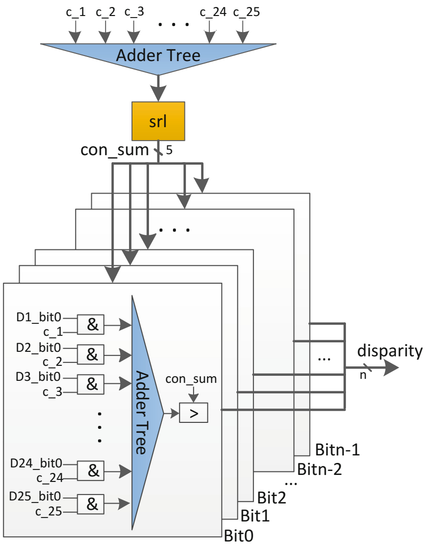

Fig. 9. Architecture of bitwise fast voting.

After the disparity voting, most invalid disparities will be updated to valid ones only by their valid neighbours. In the disparity inpainting module, the remaining invalid disparities are replaced with the closest valid ones. At the end, the refined disparity maps are written out in scanline order.

5 Experimental Results

A prototype of the proposed system has been implemented on an Altera EP4SGX-230 FPGA board. It is evaluated using rectified synthetic stereo images, initially stored in the DDR2 memory. In addition, the system is designed to be scaled with image resolution, disparity range and parallelism degree. Unless stated otherwise, the image resolution is defined as 1024×768, the disparity range as 64, the row-level parallelism P_{row} as 4, and the disparity-level parallelism P_{dis} as 8 in our design. The evaluation results of three important aspects — resource utilization, processing speed and quality evaluation — are elaborated in the following subsections.

5.1 Resource Utilization

Table 1 lists the detailed resource utilization of the FPGA prototype. The proposed system occupies 80% of the ALUTs, 58% of the registers, and 16% of the memory bits on the FPGA board, and can operate at 120 MHz. As shown in Table 1, the majority of the ALUTs and registers are consumed in the stereo-matching stage, which is mainly composed of the weight generators and the cost aggregators. So the resource utilization is mainly determined by the parallelism degree in the proposed system. The two parameters, the disparity-level parallelism P_{dis} and the row-level parallelism P_{row}, can be scaled to make the system more flexible. To make a tradeoff between resource utilization and processing speed, P_{dis} is 8 and P_{row} is 4 in the current system. On the other hand, the memory bits are mostly used as line buffers to store original pixel data and temporary results.

Table 1. Resource utilization report

Altera EP4SGX230	ALUTs total: 228000	Registers total: 228000	Memory bits total: 17133000
Pre-processing	4170	2286	1015808
Stereo-matching	158496	120292	1638400
Post-processing	18852	9684	28672
Whole system	181518	132262	2682880

5.2 Processing Speed

The processing speed of stereo matching is given by million disparity estimations per second (MDE/s), which is calculated by (image resolution $\times$ disparity range $\times$ frame rate). Table 2 presents a comparison between some exiting implementations and the proposed system. It is shown that CPU and GPU based implementations can hardly achieve real-time speed with high resolution images.

Table 2. Processing speed comparison

Design	Platform	Image size	Disparity range	FPS	MDE/s
Shan et al. [15]	FPGA	1280 × 1024	256	46	15437
MCADSR [17]	FPGA	1024 × 768	128	129	13076
AWDE-IR [14]	FPGA	1024 × 768	128	60	6040
Zhang et al. [13]	FPGA	1024 × 768	64	60	3019
Wang et al. [18]	FPGA	1024 × 768	96	31.8	2400
Ttofis et al. [10]	FPGA	640 × 480	64	30	589
MCADSW [12]	ASIC	352 × 288	64	42	272
AD-Census [19]	GPU	450 × 375	60	10.6	107
Yang et al. [20]	GPU	640 × 360	20	10	46
VariableCross [21]	CPU	450 × 375	60	0.63	13
SemiGlobal [22]	CPU	450 × 375	64	0.55	6
Proposed	FPGA	1024 × 768	64	65	6543

For FPGA based implementations, the achievable processing speed is usually limited by the available hardware resources, such as on-chip memories. The proposed system is able to achieve 65 fps for 1024 × 768 images with a disparity range of 64 pixels. Although the design in [15] has the highest processing speed, it is based on a simple SAD matching method that leads to low accuracy. Likewise, the system in [17] improves the accuracy with variable support regions, but its disparity quality is still worse than that of the proposed system.

5.3 Quality Evaluation

To discuss the quality of the proposed system, the disparity maps are evaluated based on the Middlebury benchmark using the percentage of bad pixels on different regions, a commonly accepted metric [1]. Table 3 lists the accuracy comparison with some state-of-the-art implementations. The average error rate of the final disparity maps in the proposed system is 6.56%. The first row shows the results of the AD-Census algorithm implemented on GPUs. However, it is challenging to realize it into an FPGA because of its multi disparity enhancement functions. The design in [18] utilizes cross-based regions and semi-global optimization on an FPGA, but its high accuracy is achieved at the expense of the decreased processing speed. The SegSupport algorithm outperforms the proposed design slightly but fails to reach real-time performance. The algorithms in [21,22], which are also software implementations, have a higher error rate than the proposed algorithm. To summarize, the comparison shows that the accuracy of the disparity maps is not only among the best in hardware accelerated stereo systems, but also competitive with state-of-the-art software implementations.

Table 3. Accuracy comparison on the middlebury benchmark

Data set	Tsukuba			Venus			Teddy			Cones			Average error rate
Evaluation	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc	
AD-Census [19]	1.07	1.48	5.73	0.09	0.25	1.15	4.10	6.22	10.9	2.42	7.25	6.95	3.97
Wang et al. [18]	1.93	2.95	7.90	0.61	1.43	2.87	6.44	13.8	16.0	2.37	11.1	6.70	6.17
SegSupport [9]	1.25	1.62	6.68	0.25	0.64	2.59	8.43	14.2	18.2	3.77	9.87	9.77	6.44
MCADSW [12]	–	2.80	–	–	0.64	–	–	13.7	–	–	10.1	–	all = 6.81
SemiGlobal [22]	3.26	3.96	12.8	1.00	1.57	11.3	6.02	12.2	16.3	3.06	9.75	8.90	7.50
VariableCross [21]	1.99	2.65	6.77	0.62	0.96	3.20	9.75	15.1	18.2	6.28	12.7	12.9	7.60
MCADSR [17]	3.62	4.15	14.0	0.48	0.87	2.79	7.54	14.7	19.4	3.51	11.1	9.64	7.65
Zhang et al. [13]	3.84	4.34	14.2	1.20	1.68	5.62	7.17	12.6	17.4	5.41	11.0	13.9	8.20
Ttofis et al. [10]	4.48	6.04	12.7	6.01	7.47	18.2	21.5	28.1	28.8	17.1	25.9	25.8	16.8
Result1[1]	9.86	11.3	19.3	5.44	7.64	17.9	10.3	19.3	22.4	4.88	15.3	12.3	13.0
Result2[2]	3.50	3.98	11.7	0.44	0.71	5.66	4.60	9.25	16.1	3.34	8.64	10.8	6.56

[1]The error rate of the initial disparity maps before the disparity refinement step.
[2]The error rate of the final disparity maps in the proposed system.

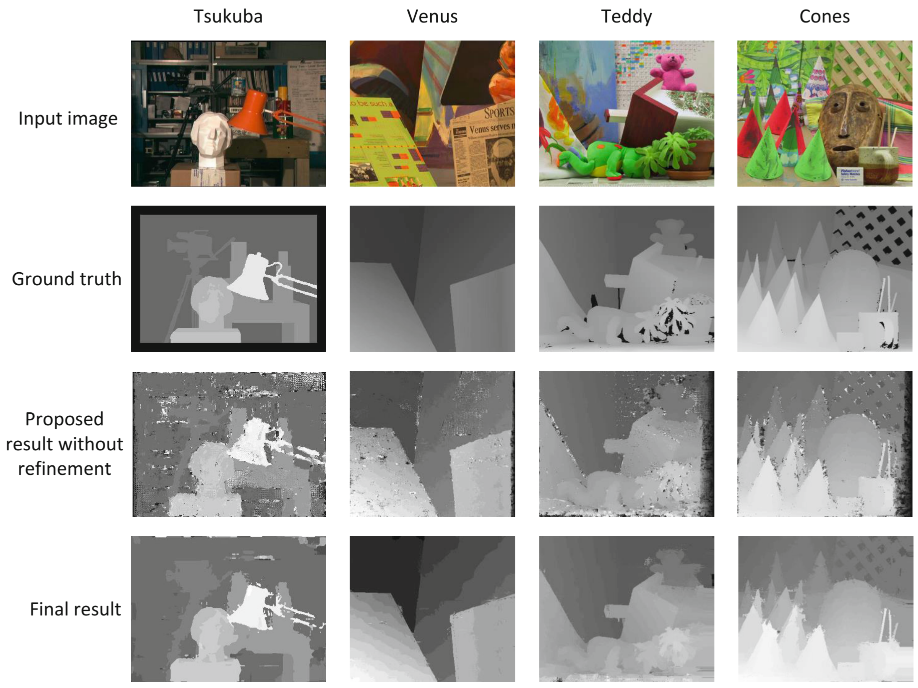

Fig. 10. True disparity maps and experimental results.

The disparity maps of the four data sets *Tsukuba, Venus, Teddy* and *Cones* are displayed in Fig. 10, and the final disparity maps are compared with the initial disparity maps to demonstrate the effect of the disparity refinement. Some of them are generated from the left images, and the others are generated from the

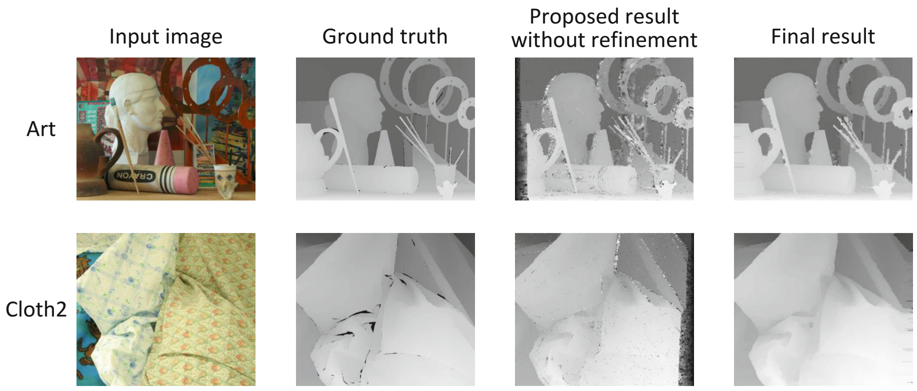

Fig. 11. Evaluation results on high-definition images.

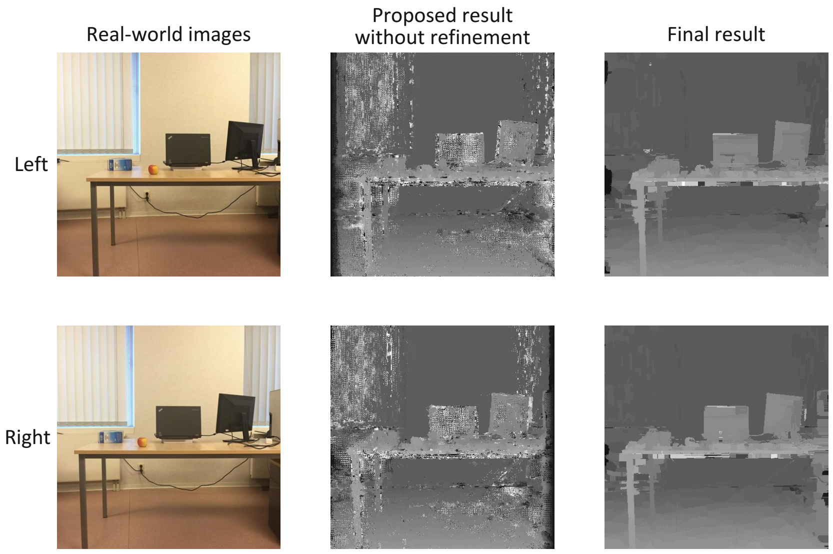

Fig. 12. Evaluation results on real-world images.

right images. In this way, the proposed system is comprehensively evaluated. It is observed that the refinement step contributes significantly to the final disparity maps and many visual improvements are obvious, including the elimination of speckle noise, fewer errors at the image borders and sharply delineated edges. The quantitative results in Table 3 also verify it.

The resolutions of the data sets *Tsukuba, Venus, Teddy* and *Cones* are all smaller than that of VGA. To further evaluate the proposed design, some high-definition images in the benchmark are used with a disparity range of 128 pixels. The results of the data sets *Art* and *Cloth2* captured at different viewpoints are shown in Fig. 11. The overall error rates are 12.85% and 4.67%, respectively. The proposed system provides quite clear and smooth disparity maps and the accuracy is comparable to the low-definition results.

The reference images in the benchmark are all well captured and rectified so that the results are quite accurate. But the quality of the disparity maps for real-world images may decrease due to some undesirable factors, such as luminance differences and rectification errors. The proposed system is further evaluated by real-world images to prove its robustness. The images are captured by two adjacent cameras in a office, and then rectified by the toolbox in Matlab 2016b. Here the image resolution is 460×460, and the disparity range for the real-world images is defined as 45. It is noted that our system still provides high-quality disparity maps for the real-world images, as shown in Fig. 12.

6 Conclusion

This contribution has proposed a stereo matching algorithm based on the mini-census transform and the segmentation-based ADSW. The disparity refinement step with segmentation information has been presented and the quality of disparity maps has been improved significantly. Furthermore, a fully pipelined and scalable hardware architecture is designed with hardware-oriented optimizations. A prototype of the hardware system has been built on an Altera Stratix-IV FPGA board. The design is evaluated on the Middlebury benchmark and the average error rate is 6.56%. The experimental results have shown that our hardware system has the top-performing processing ability and its accuracy is competitive with state-of-the-art software implementations. In the future, we will introduce a part of global matching algorithms to achieve higher accuracy of disparity maps.

Acknowledgment. The research in this contribution was sponsored in part by the Belgian FWO (Flemish Research Council) and the Chinese MOST (Ministry of Science and Technology) bilateral cooperation project number G.0524.13.

References

1. Scharstein, D., Szeliski, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. Int. J. Comput. Vis. **47**, 7–42 (2002)
2. Hirschmuller, H., Scharstein, D.: Evaluation of cost functions for stereo matching. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8. IEEE Press, Minneapolis (2007)
3. Veksler, O.: Stereo correspondence by dynamic programming on a tree. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 384–390. IEEE Press, San Diego (2005)
4. Klaus, A., Sormann, M., Karner, K.: Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. In: 18th International Conference on Pattern Recognition (ICPR 2006), pp. 15–18. IEEE Press, Hang Kong (2006)

5. Kang, S.B., Szeliski, R., Chai, J.: Handling occlusions in dense multi-view stereo. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 103–110. IEEE Press, Kauai (2001)
6. Hirschmuller, H.: Accurate and efficient stereo processing by semiglobal matching and mutual information. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 807–814, San Diego (2005)
7. Kanade, T., Okutomi, M.: A stereo matching algorithm with an adaptive window: theory and experiment. IEEE Trans. Pattern Anal. Mach. Intell. **16**, 920–932 (1994)
8. Yoon, K.-J., Kweon, I.-S.: Adaptive support-weight approach for correspondence search. IEEE Trans. Pattern Anal. Mach. Intell. **28**, 650–656 (2006)
9. Tombari, F., Mattoccia, S., Stefano, L.: Segmentation-based adaptive support for accurate stereo correspondence. In: Mery, D., Rueda, L. (eds.) PSIVT 2007. LNCS, vol. 4872, pp. 427–438. Springer, Heidelberg (2007). doi:10.1007/978-3-540-77129-6_38
10. Ttofis, C., Theocharides, T.: Towards accurate hardware stereo correspondence: a real-time FPGA implementation of a segmentation-based adaptive support weight algorithm. In: Proceedings of the Conference on Design, Automation & Test in Europe, Conference and Exhibition (DATE), pp. 703–708. IEEE Press, Germany (2012)
11. Middlebury benchmark. http://vision.middlebury.edu/stereo/
12. Chang, N.Y.-C., Tsai, T.-H., Hsu, B.-H., Chen, Y.-C., Chang, T.-S.: Algorithm and architecture of disparity estimation with mini-census adaptive support weight. IEEE Trans. Circ. Syst. Video Technol. **20**, 792–805 (2010)
13. Zhang, L., Zhang, K., Chang, T.S., Lafruit, G., Kuzmanov, G.K., Verkest, D.: Real-time high-definition stereo matching on FPGA. In: 19th ACM/SIGDA International Symposium on Field Programmable Gate Arrays, pp. 55–64. ACM Press, Monterey (2011)
14. Akin, A., Baz, I., Schmid, A., Leblebici, Y.: Dynamically adaptive real-time disparity estimation hardware using iterative refinement. Integr. VLSI J. **47**, 365–376 (2014)
15. Shan, Y., Wang, Z., Hao, Y., Wang, Y., Tsoi, K., Luk, W., Yang, H.: FPGA based memory efficient high resolution stereo vision system for video tolling. In: International Conference on Field-Programmable Technology (FPT), pp. 29–32. IEEE Press, Seoul (2012)
16. Zhang, K., Lu, J., Lafruit, G., Lauwereins, R., Gool, L.V.: Real-time accurate stereo with bitwise fast voting on CUDA. In: 12th International Conference on Computer Vision Workshops (ICCV Workshops), pp. 794–800. IEEE Press, Kyoto (2009)
17. Shan, Y., Hao, Y., Wang, W., Wang, Y., Chen, X., Yang, H., Luk, W.: Hardware acceleration for an accurate stereo vision system using mini-census adaptive support region. ACM Trans. Embed. Comput. Syst. **13**, 1–24 (2014)
18. Wang, W., Yan, J., Xu, N., Wang, Y., Hsu, F.H.: Real-time high-quality stereo vision system in FPGA. In: International Conference on Field-Programmable Technology (FPT), pp. 358–361. IEEE Press, Kyoto (2013)
19. Mei, X., Sun, X., Zhou, M., Jiao, S., Wang, H., Zhang, X.: On building an accurate stereo matching system on graphics hardware. In: 14th International Conference on Computer Vision Workshops (ICCV Workshops), pp. 467–474. IEEE Press, Barcelona (2011)

20. Yang, Q., Li, D., Wang, L., Zhang, M.: Fast local stereo matching using two-level adaptive cost filtering. In: International Conference on Acoustics, Speech and Signal Processing, pp. 1986–1990. IEEE Press, Vancouver (2013)
21. Zhang, K., Lu, J., Lafruit, G.: Cross-based local stereo matching using orthogonal integral images. IEEE Trans. Circ. Syst. Video Technol. **19**, 1073–1079 (2009)
22. Hirschmuller, H.: Stereo processing by semiglobal matching and mutual information. IEEE Trans. Pattern Anal. Mach. Intell. **30**, 328–341 (2008)