Real-Time High-Quality Stereo Vision System in FPGA

Wenqiang Wang, Jing Yan, Ningyi Xu, Yu Wang, *Senior Member, IEEE*, and Feng-Hsiung Hsu

***Abstract*— Stereo vision is a well-known technique for acquiring depth information. In this paper, we propose a real-time high-quality stereo vision system in field-programmable gate array (FPGA). Using absolute difference-census cost initialization, cross-based cost aggregation, and semiglobal optimization, the system provides high-quality depth results for high-definition images. This is the first complete real-time hardware system that supports both cost aggregation on variable support regions and semiglobal optimization in FPGAs. Furthermore, the system is designed to be scaled with image resolution, disparity range, and parallelism degree for maximum parallel efficiency. We present the depth map quality on the Middlebury benchmark and some real-world scenarios with different image resolutions. The results show that our system performs the best among FPGA-based stereo vision systems and its accuracy is comparable with those of current top-performing software implementations. The first version of the system was demonstrated on an Altera Stratix-IV FPGA board, processing 1024 × 768 pixel images with 96 disparity levels at 67 frames/s. The system is then scaled up on a new Altera Stratix-V FPGA and the processing ability is enhanced to 1600 × 1200 pixel images with 128 disparity levels at 42 frames/s.**

***Index Terms*— Field-programmable gate array (FPGA), hardware accelerator, high-quality depth map, stereo vision.**

I. Introduction

STEREO vision is an active research area in computer vision, as it is widely used in many applications. Usually, a stereo vision system has two cameras to capture two different images. Stereo matching is a key function of a stereo vision system. The purpose of stereo matching is to search for disparities between corresponding pixels in stereo images to make the cost function achieve minimum results among all disparities. Then the depth could be calculated from the inverse of this disparity. Stereo matching is a complicated and time-consuming procedure, which makes it hard to process in real time on a CPU. Current research efforts on stereo vision focus on the depth accuracy and processing speed of stereo matching.

Stereo matching algorithms could be divided into two types: 1) local methods and 2) global methods [1]. Local methods compute depth at a local region, usually on a fixed support window; and global methods compute depth based on a global cost optimization [2]. Because local methods use only local information to minimize the cost function, the accuracy is lacking in low-texture regions and occluded regions. While global methods show better results on these regions [2], they are not easily implemented using dedicated hardware because they cause huge pressure on hardware resources due to huge intermediate computing results and large volumes of irregular data access. This is why the majority of existing hardware implementations use local methods [3].

Most of the current acceleration works for stereo matching just evaluated their accuracy on the Middlebury low-resolution benchmark [4]. However, the implementation cannot always maintain a good depth quality when the scenario or the image resolution changes. In this paper, we focus on both depth quality in different scenarios and processing speed for high-definition images. We are motivated by the absolute difference (AD)-Census algorithm [5], which has ranked first on the Middlebury benchmark [4] since 2011. AD-Census is a combination of several existing state-of-the-art technologies. We examine the major hypotheses of these key technologies and further optimize them for hardware implementation on field-programmable gate array (FPGA), which are discussed in Sections III and IV. We discuss processing speed and depth quality in different scenarios in Section V.

The major contributions of this paper are as follows.

1) We propose a hardware-friendly stereo matching algorithm, optimizing it for high depth quality and real-time processing at high resolutions. This is the first complete real-time hardware system that supports both cost aggregation on variable support regions and semiglobal optimization on FPGA.
2) We propose a sophisticated hardware architecture with optimized parallelism scheme to accelerate the proposed algorithm. The hardware implementation is fully parameterized and could easily be scaled up. A novel semiglobal optimization structure that could support hybrid parallelism is proposed.

Manuscript received April 11, 2014; revised August 11, 2014 and November 12, 2014; accepted January 21, 2015. Date of publication January 30, 2015; date of current version September 30, 2015. This work was supported in part by Microsoft, in part by the 973 Project under Grant 2013CB329000, in part by the National Natural Science Foundation of China under Grant 61373026, and in part by the Tsinghua University Initiative Scientific Research Program. This paper was recommended by Associate Editor S. Shirani.

W. Wang was with Microsoft Research Asia, Beijing 100080, China. He is now with the Tsinghua National Laboratory for Information Science and Technology, Center for Brain-Inspired Computing Research, Department of Electronic Engineering, Tsinghua University, Beijing 100084, China (e-mail: wangwqee83@163.com).

J. Yan, N. Xu, and F.-H. Hsu are with the Hardware Computing Group, Microsoft Research Asia, Beijing 100080, China (e-mail: jingyan@microsoft.com; xu.ningyi@microsoft.com; fhh@microsoft.com).

Y. Wang is with Tsinghua National Laboratory for Information Science and Technology, Center for Brain-Inspired Computing Research, Department of Electronic Engineering, Tsinghua University, Beijing 100084, China (e-mail: yu-wang@mail.tsinghua.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2015.2397196

3) We develop a prototype on Altera EP4SGX230 FPGA and upgrade to Altera 5SGSMD5H2 FPGA. The previous version achieves 67 frames/s for 1024 × 768 pixel images and the upgraded version achieves 42 frames/s while processing 1600 × 1200 pixel images. The average error rate of the implementation on the Middlebury benchmark is 5.61%, which is the best of all hardware implementation works to our knowledge.

II. Background and Related Work

As introduced in Section I, to achieve real-time processing, dedicated hardware platforms have been used to accelerate stereo vision systems. Most of the existing works use graphic processing units (GPUs) and FPGAs as their acceleration platforms. Developed in 2002, [6] was the first work to use GPUs to accelerate depth estimation. Then there are some papers on GPU accelerations for stereo matching algorithms using local methods. Among these works, the implementation of [7] has the best speed performance, achieving 12 frames/s at a 450 × 375 resolution with 64 disparity levels, and obtaining a relatively good matching accuracy. Some researchers have also tried to implement global methods on GPUs for better quality. Yang *et al.* [8] and Wang *et al.* [9] use a hierarchical belief propagation and an adaptive aggregation-based dynamic programming separately to achieve high-quality real-time processing. Rosenberg *et al.* [10] implements semiglobal matching on GPUs at 8 frames/s for 320 × 240 pixel images with 64 disparities. The AD-Census algorithm with scanline optimization was proposed and implemented on GPUs in 2011 at a frame rate of about 10 frames/s with a 512 × 384 resolution and 60 disparity levels. It presents the best depth quality in terms of the Middlebury benchmark. Although GPU has proved to be an attractive speedup platform for stereo matching, high power consumption restricts its performance.

Compared with GPUs, FPGA has two advantages: 1) reconfigurable processing units and customized memory hierarchies and 2) low power. Jin *et al.* [11] and Gudis *et al.* [12] have developed real-time stereo vision systems for 640 × 480 resolution images. Zhang *et al.* [13] combine the mini-census transform and cross-based cost aggregation in their structure, which achieves 60 frames/s at 1024 × 768 pixel stereo images. Shan *et al.* [14] combine disparity-level parallelism and row-level parallelism, achieving a 400 frames/s at 640 × 480 resolution with 128 disparity levels. Jin and Maruyama [15] apply cost aggregation and fast locally consistent dense stereo functions. They use Virtex-6 FPGA as the platfrom and achieve 507.9 frames/s for 640 × 480 resolution images. These works have achieved real-time processing; however, to fit the algorithm on FPGAs, they used local methods to compute disparities and have some limitations in large no-texture regions. Thus, they cannot attain good quality for high-definition images. Recently, some researchers have started to implement stereo matching algorithms using global methods on FPGAs for better performance. Park and Jeong [16] and Sabihuddin *et al.* [17] implement a trellis-based dynamic programming and a maximum-likelihood dynamic programming methods on FPGAs in real-time. Jin and Maruyama [18] implement a tree-structured dynamic programming on FPGA and achieves better performance. Gehrig *et al.* [19] develop an FPGA prototype using the semiglobal matching for automotive applications. The most recent work for the semiglobal matching is [3], which proposes a systolic-array-based architecture to obtain a 30 frames/s performance for the 640 × 480 pixel images with a 128-disparity range. Both focus on the semiglobal matching implementations and pay little attention to the other functions of the system. Besides, the influence of the image resolution is not fully discussed.

We have explored state-of-the-art stereo matching technologies extensively and tried to develop an efficient system to achieve better quality and real-time processing speed for the high-definition images based on FPGA.

TABLE I
Explanation of the Symbols in This Paper

Symbol	Explanation
H, W	Height and width of the image
L_{max}	Maximum arm length
N_D	Number of the disparity levels
d_{max}	Maximum disparity value $(N_D - 1)$
(x, y, d)	(Row address, column address, disparity)
$I(\boldsymbol{p})$	Intensity value of pixel $\boldsymbol{p}$
C_I	Initial cost
C_{AD}	AD cost
C_{census}	Census cost
C_{agg}	Aggregated cost
$L_{\boldsymbol{r}}$	Optimized path cost in direction $\boldsymbol{r}$
C_{final}	Final cost after semi-global optimization
P_D	Disparity-level parallelism
P_R	Row-level parallelism

III. Hardware-Friendly Stereo Matching Algorithm

Stereo matching is the key component of the stereo vision system. It aims to find the correspondence between the input stereo images. In this section, we discuss the algorithm selection and tuning considerations for a high-quality hardware-friendly implementation. The symbols commonly used in the paper are listed in Table I.

A. Key Algorithm Selection

Stereo matching aims to find the correspondence between the input images captured in different views. In this paper, we aim at a real-time stereo vision system with high depth quality and high image resolution. Global methods usually provide better depth map, but the computing complexity is also huge. It is very difficult to achieve real-time processing even with hardware acceleration, and thus local methods are selected as the basic algorithm in this paper.

The basic idea of local methods is to determine the correspondence by pixel-to-pixel comparison. Each pixel $\boldsymbol{p}(x; y)$ in the left image is compared with multiple pixels $\boldsymbol{q}(x-d, y), d = 0, 1, 2, \ldots, d_{\max}$ in the right image. The most similar one is selected as its corresponding pixel. However, single pixel comparison is easily affected by noises or ambiguities. Thus, two support regions $SP(\boldsymbol{p})$ and $SP(\boldsymbol{q})$ surrounding

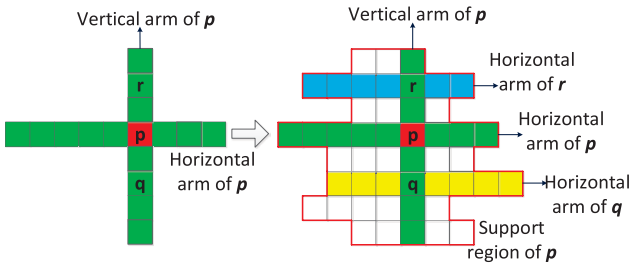


Fig. 1. Cross-based support region. The support region of pixel $\boldsymbol{p}$ is composed of multiple horizontal arms. The maximum arm length is $L_{\max}$.

the processing pixels $\boldsymbol{p}$ and $\boldsymbol{q}$ are compared instead. The algorithm is usually divided into several basic steps: 1) cost initialization; 2) cost aggregation; 3) disparity selection; and 4) postprocessing, as shown in (1)–(3). The cost initialization step computes the initial cost $C_I(\boldsymbol{p}, d)$, which means the difference between pixel $\boldsymbol{p}$ and $\boldsymbol{q}$. Then the aggregated cost $C_{\text{agg}}(\boldsymbol{p}, d)$, which represents the difference between support regions $SP(\boldsymbol{p})$ and $SP(\boldsymbol{q})$, could be computed by the aggregation of the initial costs in the support region. After aggregation, the raw disparity $d_{\text{res}}(\boldsymbol{p})$ is selected with a winner-takes-all (WTA) method. Finally, a postprocessing step is applied to refine the raw disparity map

$$C_I(\boldsymbol{p}, d) = \text{diff}(\boldsymbol{p}, \boldsymbol{q}) \tag{1}$$

$$C_{\text{agg}}(\boldsymbol{p}, d) = \text{diff}(SP(\boldsymbol{p}), SP(\boldsymbol{q})) = \sum_{\boldsymbol{q} \in SP(\boldsymbol{p})} C_I(\boldsymbol{q}, d) \tag{2}$$

$$d_{\text{res}}(\boldsymbol{p}) = \underset{0 \le d \le d_{\max}}{\text{argmin}}\ C_{\text{agg}}(\boldsymbol{p}, d). \tag{3}$$

Cost aggregation is the most important function in local methods because it influences the computing complexity and depth accuracy. A fixed rectangle support region is easy to implement, but the depth map is often blurred in the discontinuous regions. To maintain clear object edges, the pixels with different depths should be excluded when constructing the support regions. Thus, the support region should be variable in shape. However, the 2-D variable support region makes cost aggregation more complex. To solve this problem, a cross-based support region is adopted in this paper. The cross-based support region was first proposed in [20], as shown in Fig. 1. First, a cross is constructed for each pixel. The cross consists of four adaptive support arms composed of pixels similar in color to the central pixel. The support region of pixel $\boldsymbol{p}$ is composed of all the horizontal arms of those pixels in the vertical arm of pixel $\boldsymbol{p}$. As an advantage of this special structure, the aggregation could be divided into two steps: 1) horizontal aggregation and 2) vertical aggregation. This technology greatly reduces the computing complexity.

Fig. 2 shows the disparity maps of a 3-D video with different configurations. The original video [21] is captured by a Sony HDR-TD10 Video Camera in a Natural History Museum with a resolution of 1280×720. With a variable support region, we could see that the depth becomes more accurate in the discontinuous regions.

However, there are still problems after introducing variable support region. First, there are some noisy pixels in the disparity map. When the shape of support region is related

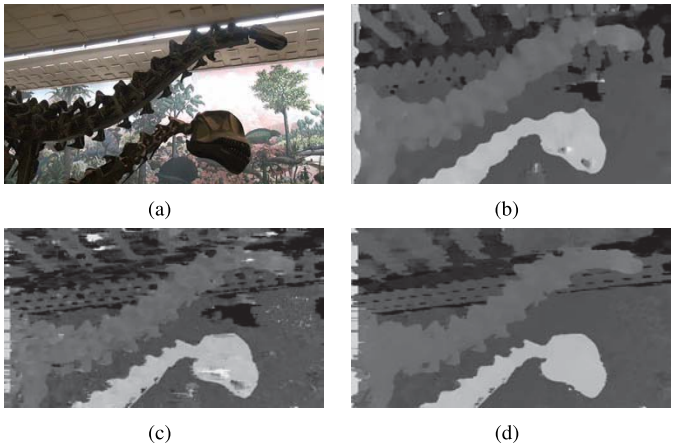


Fig. 2. Depth results of a 3-D video. (a) Original image. (b) Result of fixed support region. (c) Result of variable support region. (d) Result after semiglobal optimization.

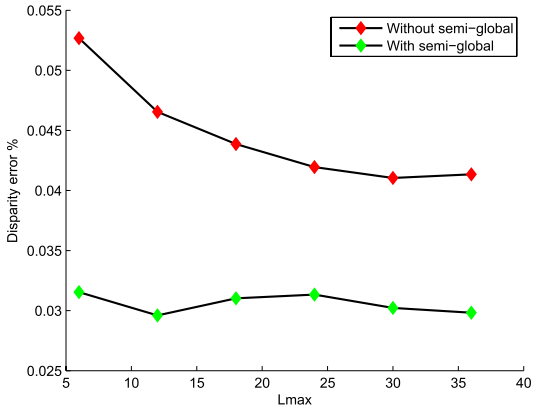


Fig. 3. Disparity error rate with different $L_{\max}$s. The original figure is the baby data set in Middlebury benchmark [4].

to the pixel values, we find the noise could affect the support region size and bring mismatched pixels. Second, there are some mismatched holes in the disparity map. In the large no-texture regions, the pixels in one support region have the same value, and thus, the disparity values are hard to determine due to the ambiguity. This problem could deteriorate with the increasing of the image resolution. For example, there are large mismatched holes in Fig. 2(c). A larger support region size could solve part of this problem. As shown in Fig. 3, a large $L_{\max}$ has a certain improvement on the error rate, but it is still worse than the algorithm with some global methods. Besides, the resource utilization also increases with larger support region. We find that the support region should be larger than 100×100 to remove the mismatched holes, as shown in Fig. 2(c). It will cause unacceptable resource utilization in the cost aggregation step. Third, the disparity map computed by the local method is not smooth even in the same surface. These discontinuous regions usually mean mismatched pixels.

Global optimization is a better choice to further improve the depth quality. As shown in Fig. 3, we can get high depth quality without changing the arm length. However, the computing complexity of global methods is unacceptable for

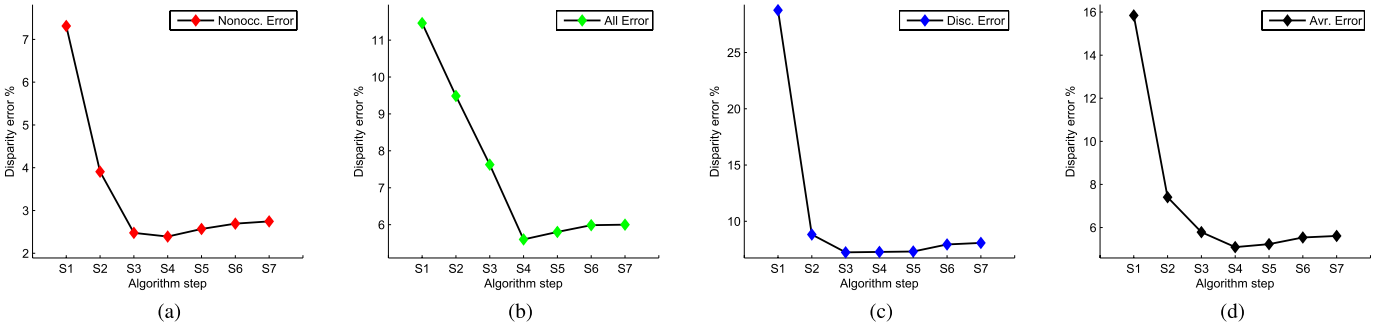


Fig. 4. Average disparity error percentages after performing each algorithm step and tuning. (a)–(c) Disparity error rates in different regions (nonoccluded, all, and discontinuous, respectively). (d) Average error rate of (a)–(c). S1: initial algorithm. S2: with cross-based variable support region. S3: with semiglobal optimization. S4: with postprocessing. S5: after inverting aggregation sequence. S6: after simplifying semiglobal optimization directions. S7: after fixed-point conversion.

real-time processing even with hardware accelerations. To reduce the computing complexity, a semiglobal optimization method is proposed in [7]. Instead of solving a 2-D global optimization problem, the semiglobal method simplifies the smoothness costs so that the costs could be optimized along different directions separately, significantly reducing the computing complexity. Semiglobal optimization also requires many resources, but it is much smaller than the resources that traditional global methods require, which makes real-time processing possible. This has been proved in [3]. With bigger FPGA and optimized hardware structure, real-time semiglobal optimization for high-definition images is practical. Thus, it is adopted in this paper for further optimization of smoothness.

The semiglobal method improves the depth accuracy in both subjective and objective evaluations. The depth map of the 3-D video after semiglobal optimization is shown in Fig. 2(d). There is no ground truth for the scenario, but we could see that the depth map becomes more smooth and the mismatched holes are removed. The objective evaluation based on Middlebury benchmark [4] is shown in Fig. 4. We could see that the disparity error rate is reduced by around 1.5% with semiglobal optimization. Furthermore, semiglobal optimization makes the system scalable to high-definition scenarios. The high-definition images usually have more no-texture regions, which may be larger than the maximum support region. These no-texture regions are more likely to be mismatched using the local stereo matching methods. Applying semiglobal optimization, this problem could be solved without tuning the parameters. The related evaluations are performed in Section V-D1.

B. Overall Algorithm Flow

To achieve high accuracy and fast processing speed, we select cross-based cost aggregation and semiglobal optimization as the key components. The final stereo matching algorithm is composed of AD-Census cost initialization, cross-based cost aggregation, semiglobal optimization, disparity selection, and postprocessing.

1) AD-Census Cost Initialization: As discussed in Section III-A, the cost initialization step computes the difference between pixels in the left image and the right image. A basic measurement of the initial cost is AD, which represents the mean value of absolute color differences, as shown in (4). AD is simple to implement, but it is easily affected by the radiometric differences between stereo cameras. To solve this problem, the census transform proposed in [22] is adopted. It encodes the pattern of intensity change into a vector and the initial cost is defined as the hamming distance between census vectors. The assumption that the color information is consistent between stereo images is not needed in census-based cost initialization, and thus, it is more robust to radiometric differences. In [5], the information of AD and census transform are combined and it is proved to be an efficient measurement of the initial cost. A robust function is further applied and the final initial cost is the sum of AD cost and census cost, as shown in

$$C_{\mathrm{AD}}(\boldsymbol{p},d)=\frac{\sum_{i=R,G,B}\left|I_i^{\mathrm{left}}(\boldsymbol{p})-I_i^{\mathrm{right}}(\boldsymbol{p}-(d,0))\right|}{3} \tag{4}$$

$$C_I(\boldsymbol{p},d)=1-\exp\left(-\frac{C_{\mathrm{AD}}(\boldsymbol{p},d)}{\lambda_{\mathrm{AD}}}\right)+1-\exp\left(-\frac{C_{\mathrm{census}}(\boldsymbol{p},d)}{\lambda_{\mathrm{Census}}}\right). \tag{5}$$

2) Cross-Based Cost Aggregation: The initial costs are aggregated in the cross-based variable support regions. As we introduced in Section III-A, cross-based cost aggregation could be performed by horizontal aggregation first and vertical aggregation then. The computing procedure is

$$C_{\mathrm{aggh}}(\boldsymbol{p},d)=\sum_{\boldsymbol{q}\in\mathrm{HARM}(\boldsymbol{p})}C_I(\boldsymbol{q},d) \tag{6}$$

$$C_{\mathrm{agg}}(\boldsymbol{p},d)=\sum_{\boldsymbol{q}\in\mathrm{VARM}(\boldsymbol{p})}C_{\mathrm{aggh}}(\boldsymbol{q},d). \tag{7}$$

$\mathrm{HARM}(\boldsymbol{p})$ and $\mathrm{VARM}(\boldsymbol{p})$ represent the horizontal arm and vertical arm of pixel $\boldsymbol{p}$, respectively. For each pixel $\boldsymbol{p}$, the initial costs are first aggregated in the horizontal direction as the horizontal aggregated cost $C_{\mathrm{aggh}}(\boldsymbol{p},d)$. Then the horizontal aggregated costs are aggregated in the vertical direction as the final aggregated cost $C_{\mathrm{agg}}(\boldsymbol{p},d)$. The results are the same as a straightforward 2-D aggregation, but the computing complexity is greatly reduced.

3) Semiglobal Optimization: Semiglobal optimization optimizes the aggregated costs along different directions separately. For direction $\boldsymbol{r}$, the optimized cost is computed as

$$\begin{aligned} L_{\boldsymbol{r}}(\boldsymbol{p}, d) = {} & C_{\text{agg}}(\boldsymbol{p}, d) \\ & + \min\{L_{\boldsymbol{r}}(\boldsymbol{p}-\boldsymbol{r}, d), L_{\boldsymbol{r}}(\boldsymbol{p}-\boldsymbol{r}, d \pm 1) \\ & \quad + P_1, \min_k L_{\boldsymbol{r}}(\boldsymbol{p}-\boldsymbol{r}, k) + P_2\} \\ & - \min_k L_{\boldsymbol{r}}(\boldsymbol{p}-\boldsymbol{r}, k). \end{aligned} \tag{8}$$

The left term $L_{\boldsymbol{r}}(\boldsymbol{p}, d)$ represents the path costs in direction $\boldsymbol{r}$. P_1 and P_2 are penalties for disparity discontinuities. The final optimized costs are the sum of all the path costs

$$C_{\text{final}}(\boldsymbol{p}, d) = \sum_{\boldsymbol{r}} L_{\boldsymbol{r}}(\boldsymbol{p}, d). \tag{9}$$

4) Disparity Selection and Postprocessing: After semiglobal optimization, we use the WTA method to find the best disparity. The best disparity corresponds to the minimum cost

$$d_{\text{res}}(\boldsymbol{p}) = \underset{0 \le d \le d_{\max}}{\operatorname{argmin}} C_{\text{final}}(\boldsymbol{p}, d). \tag{10}$$

A postprocessing step is applied to refine the disparity map. The postprocessing is composed of outlier detection, outlier handling, and sub-pixel interpolation. We begin by selecting the *L–R consistency check* and *uniqueness check* to detect invalid disparity results. A pixel is marked as an outlier if its disparities in the stereo images are different or the minimum cost and the second minimum cost of this pixel are too close. For each outlier pixel, we find the nearest two reliable pixels in the *left* and *right* directions. The smaller value of these two pixels is selected as the disparity of the outlier. Finally, a sub-pixel interpolation method is used to increase the accuracy of the disparity map.

The average disparity error rate after performing each algorithm step is shown in Fig. 4. We could see that the cross-based support region and semiglobal optimization greatly improve the depth accuracy. The final software algorithm provides a disparity error rate of 5.09%.

C. Hardware-Oriented Algorithm Tuning

Most of the existing stereo matching algorithms are developed in a CPU. Although we have considered the implementation platform, there are still some difficulties in putting the algorithm in FPGA. In this section, we discuss the algorithm tunings to make the selected algorithm suitable for hardware implementation. The accuracy loss caused by the tunings is shown in Fig. 4. The final algorithm achieves a disparity error rate of 5.61%. The algorithm tunings reduce the depth accuracy of about 0.5%, but they make it possible to implement the algorithm on FPGA.

1) Inverted Aggregation Sequence: A variable support region is usually difficult to implement on FPGA. The selected cross-based support region in Fig. 1 simplifies the structure, but the resource utilization is still huge. The straightforward implementation requires storing $2 \times L_{\max} + 1$ rows of the horizontal aggregated costs to do vertical aggregation. The total size of these costs is $(2 \times L_{\max} + 1) \times W \times N_D \times \text{DW}_{\text{cost}}$,

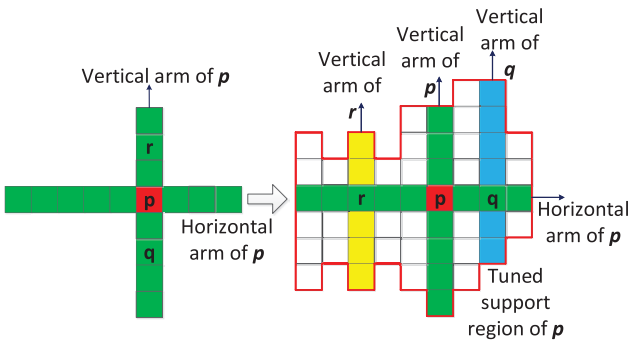


Fig. 5. Tuned aggregation sequence. The support region of pixel $\boldsymbol{p}$ is composed of multiple vertical arms.

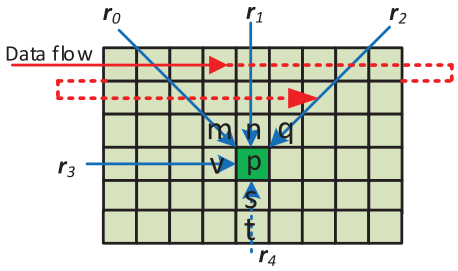


Fig. 6. Selected semiglobal optimization directions—*right bottom*, *bottom*, *left bottom*, and *right*.

where DW_{cost} represents the data width of the costs. This is too large for the on-chip memory. Zhang *et al.* [13] have proposed a fixed vertical arm method to solve this problem but it leads to significant accuracy degradation. We adopt the inverted aggregation sequence method proposed in [23], which does vertical aggregation before horizontal aggregation. The benefit of this sequence is that we could buffer the pixel values instead of the horizontal aggregated costs, reducing the total memory size to $(2 \times L_{\max} + 1) \times W \times \text{DW}_{\text{pxvalue}}$, where $\text{DW}_{\text{pxvalue}}$ represents the data width of the pixel value. The inverted aggregation sequence means that the support region is composed of multiple vertical arms, as shown in Fig. 5. This support region is still 2-D variable and the accuracy loss is acceptable (about 0.14% as shown in Fig. 4).

2) Semiglobal Optimization Simplification: The original semiglobal method optimizes costs along 16 directions [7]. However, those directions opposite to the scan line dataflow are hard to implement in FPGA. As shown in Fig. 6, when optimizing pixel $\boldsymbol{p}$ along direction $\boldsymbol{r}_4$, the results of pixel $\boldsymbol{s}$ must be ready. Furthermore, while optimizing pixel $\boldsymbol{s}$, the results of pixel $\boldsymbol{t}$ must be ready. We could find that the results of the pixel in the bottom of the column must be ready while optimizing pixel $\boldsymbol{p}$. This is not suitable for a pipelined hardware structure. In our system, only four directions are chosen for the semiglobal optimization. The four directions are *right bottom*($\boldsymbol{r}_0$), *bottom*($\boldsymbol{r}_1$), *left bottom*($\boldsymbol{r}_2$), and *right*($\boldsymbol{r}_3$), as shown in Fig. 6. The previous pixel values $\boldsymbol{m}$, $\boldsymbol{n}$, $\boldsymbol{q}$, and $\boldsymbol{v}$ in these four directions could be utilized while optimizing pixel $\boldsymbol{p}$. The corresponding accuracy loss is about 0.3%, as shown in Fig. 4.

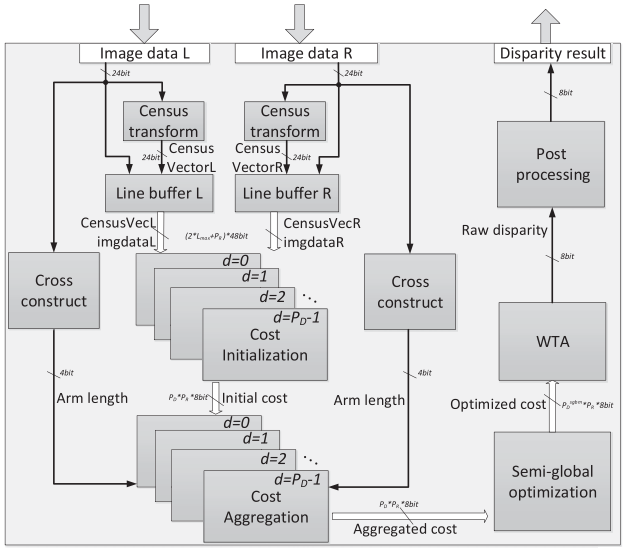


Fig. 7. Overall structure of the proposed stereo matching module.

3) Fixed-Point Conversion: The original software implementation is based on floating-point operations. However, floating-point units are very expensive in FPGA. Thus, we use fixed-point operations in the hardware. The initial costs, the aggregation costs, and the final optimized costs are all represented with 8-bit fixed-point data in FPGA. This greatly reduces the resource utilization, while the accuracy loss is acceptable (about 0.08% as shown in Fig. 4).

IV. Hardware Implementation of Stereo Matching

A. Overall Design

In this section, we discuss the FPGA implementation of the proposed stereo matching algorithm. Considering the hardware implementation, there are two major challenges. The first challenge is to develop an efficient parallelism scheme to combine the cross-based cost aggregation and semiglobal optimization. We adopt a hybrid parallelism scheme proposed in [14]. This parallelism scheme is efficient for the cost aggregation module, but it could not be used in semiglobal optimization due to the data dependency. It requires a huge buffer between the two modules. To solve this problem, we further optimize the parallelism scheme to reduce the buffer size. The second challenge is to develop a semiglobal optimization structure to support the proposed parallelism scheme. The disparity-level parallelism is a big problem due to data dependency in adjacent pixels. For example, the previous structure proposed in [3] supports pixel-level parallelism only. We propose a novel semiglobal optimization module to support both disparity-level and row-level parallelisms.

The overall architecture of the stereo matching core is shown in Fig. 7. The rectified image RGB (Red, Green, Blue) data and the corresponding census vectors are buffered and sent to the cost initialization module. The initial costs of these pixels at a certain number of disparities are then computed. Meanwhile, the arm length, which represents the size of the cross, is computed. Then the initial costs and arm length are sent to the aggregation module to compute aggregated costs. After aggregation, semiglobal optimization is applied to improve the accuracy of the costs. Finally, the disparity result is selected through the WTA module and refined through postprocessing. To build a scalable architecture, the hardware implementation is fully parameterized using the *generate* statement in Verilog. Thus, the parameter settings could be tuned to achieve the smallest resource utilization for different workloads and different platforms.

The details of the hardware implementation are discussed in Sections IV-B–IV-E. The optimized parallelism scheme is discussed first as an overall introduction of the system design. Then the implementation details of each module are discussed.

B. Parallelism Scheme

Most current hardware stereo vision systems process all the disparities in parallel and process pixel by pixel using progressive scan [11]–[13], as shown in Fig. 8(a). This parallelism scheme is simple to implement, but has been proved inefficient in [14]. They introduce row-level parallelism, which means that multiple pixels in adjacent rows are processed in parallel. The new row-level parallelism leads to little resource increase using data reuse technology. Then the disparity-level parallelism degree could be greatly reduced when targeting the same processing speed. For example, if P_R neighboring pixels along the column direction are processed in parallel and P_D disparities are processed in parallel for each pixel, we only need to make sure the total parallelism degree $P_R \times P_D$ is enough for real-time processing. Thus, P_D does not have to be equal to N_D. To get the costs for all disparity levels, each group of P_R rows is processed for $K = N_D/P_D$ passes in the cost aggregation module. In each pass, we could process P_D disparities. Thus, we completely go through $P_D \times K = N_D$ disparities in K passes. The corresponding dataflow is shown in Fig. 8(b).

The hybrid parallelism scheme in [14] reduces resource utilization of the cost aggregation module. However, it is not suitable for semiglobal optimization. As shown in (8), the input aggregated costs $C_{\text{agg}}(\boldsymbol{p}, d)$ for all the disparity levels must be ready when computing $L(\boldsymbol{p}, d)$. The required dataflow in semiglobal optimization module is shown in Fig. 8(c). Thus, the multipass dataflow of the aggregated costs in Fig. 8(b) cannot be sent to the semiglobal module directly. A simple solution is to add a cost buffer to rearrange the output aggregated costs, which is adopted in [24]. The size of the buffer is $P_R \times W \times N_D \times \text{DW}_{\text{cost}} \times 2$, where DW_{cost} represents the width of the aggregated costs. The size is multiplied by 2 because a ping-pong buffer is needed to support fully pipelined processing. This buffer is quite huge. For example, the total size of the buffer is about 6.3 Mbits if we set $W = 1024$, $N_D = 96$, $P_R = 4$, $W_{\text{cost}} = 8$. Thus, it becomes the main bottleneck for high-definition image processing.

To solve the problem, we further optimize the dataflow using the hybrid parallelism scheme, as shown in Fig. 8(d). In our design, we partition each row into multiple segments. These segments are taken as an independent row of the hybrid parallel processing scheme, which is computed in multiple passes.

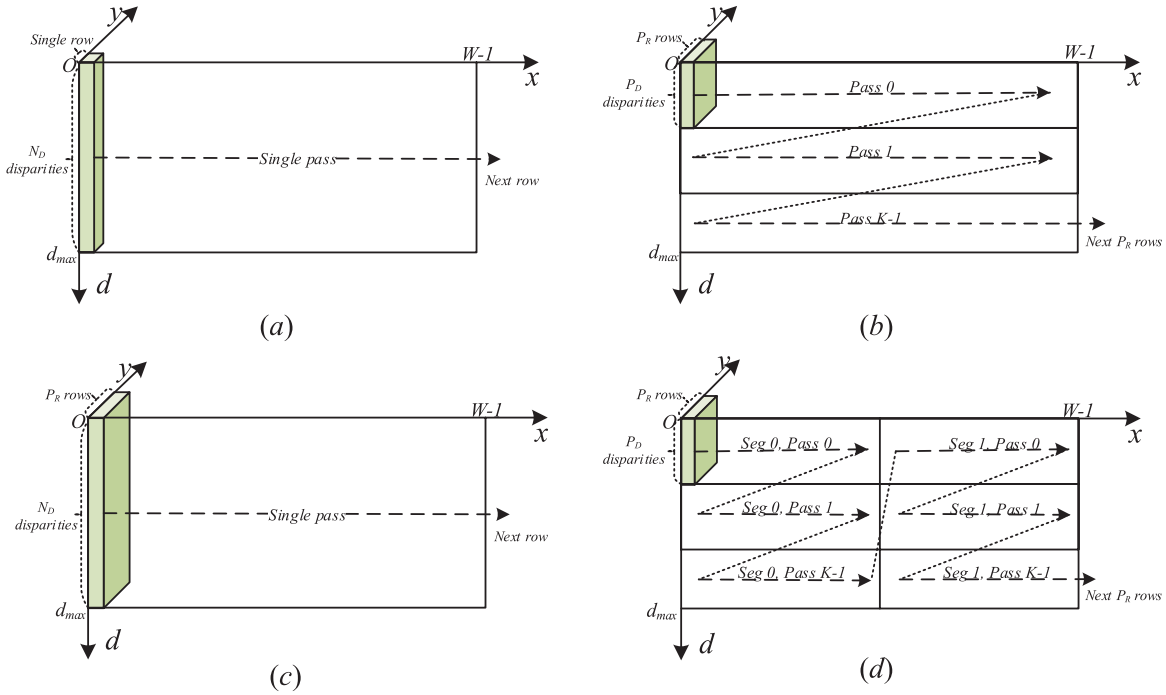


Fig. 8. Dataflow of matching costs. (a) Straightforward dataflow. (b) Optimized dataflow in [14]. (c) Required dataflow in semiglobal optimization module. (d) Proposed dataflow of the aggregated cost in this paper.

Thus, the size of the buffer to rearrange the aggregated costs depends on the width of the segment, instead of the image row. The new size is $P_R \times W_{\text{seg}} \times N_D \times \text{DW}_{\text{cost}} \times 2$, where W_{seg} represents the width of each segment. W_{seg} could be set much smaller than the width of the image, significantly reducing the buffer size. The special data access pattern needs a special design while fetching left and right image data from line buffers; we discuss this later in Section IV-C.

One problem caused by the proposed parallelism scheme is the aggregation in the edge of the segment. Because of the disparity and the support window, aggregation for pixel $\boldsymbol{p}$ involves pixel $\boldsymbol{p} - (P_D + L_{\max}, 0)$ and pixel $\boldsymbol{p} + (L_{\max}, 0)$. This means that $P_D + 2 \times L_{\max}$ extra cycles are needed to load these extra pixels so as to preheat the processing logic. For a whole row of the image, the preheating procedure is always ignored because the pixels out of the image border are unknown at all. However, after the partition, the preheating procedure cannot be ignored. Thus, each pass for one segment costs $W_{\text{seg}} + P_D + 2 \times L_{\max}$ cycles, which means that the computing efficiency caused by preheating is $W_{\text{seg}}/(W_{\text{seg}} + P_D + 2 \times L_{\max})$.

C. Cost Initialization

The cost initialization module provides the initial costs for the aggregation module. For the inverted aggregation module, the initial costs of the up and down arms are needed at the same time. To get these initial costs in parallel, we design line buffers to fetch multiple pixels, as shown in Fig. 9. The line buffer is composed of multiple block RAMs to build a wide output port. RGB and census data are written to line buffers progressively. As shown in Fig. 9, the colored block RAMs are ready for output. A column of the image could

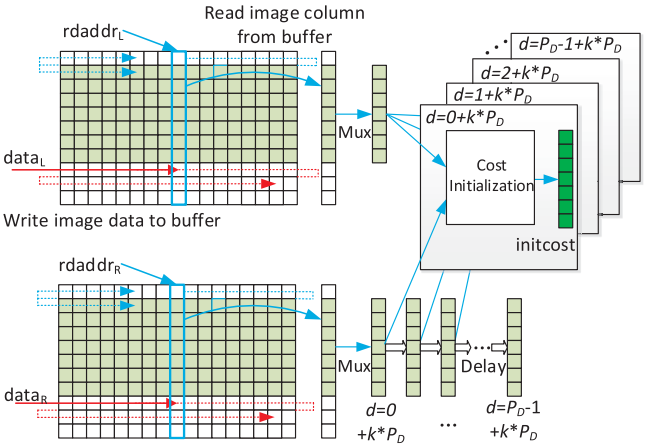


Fig. 9. Structure of the line buffers and the cost-initialization module. The size of the buffer and the disparity-level parallelism degree P_D are parameterized.

be read from the output port of the buffer. Then the needed image and census data are selected out by the multiplexer and sent to the computing units. In this paper, the initial costs are represented with 8-bit fixed-point value. And the exponential function in (5) is implemented by small look-up tables in FPGA.

As we have discussed in Section IV-B, the total N_D disparity levels are processed in $K = N_D/P_D$ passes. To process different disparity ranges in different passes, we add different biases to the read address of the right buffer $rdaddr_R$. In the kth path ($k = 0, 1, \ldots, K-1$), the bias is set as $k \times P_D$, and $rdaddr_R = rdaddr_L - k \times P_D$. Thus, we could compute the cost for $d = k \times P_D + j$ if we delay the output of the right buffer for j cycles. For example, in the second pass, $rdaddr R$ should be equal to $rdaddr_L - 2 \times P_D$, and we could

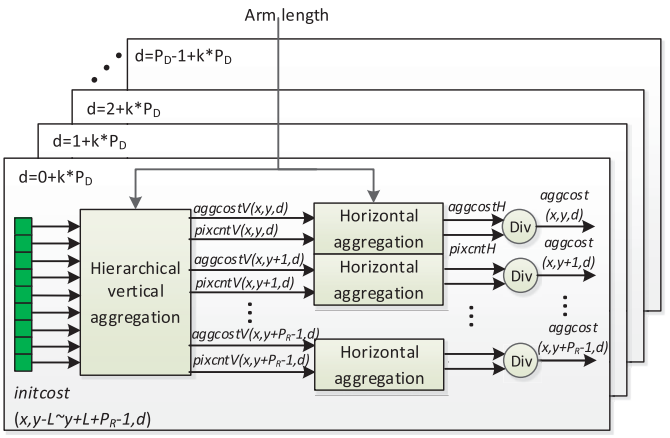


Fig. 10. Structure of the cost aggregation module. P_D and P_R are parameterized.

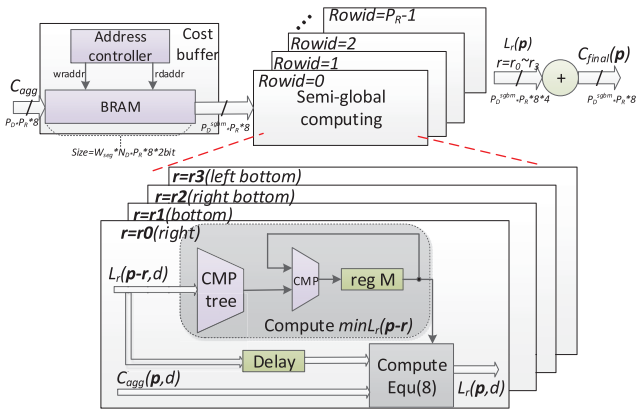


Fig. 11. Structure of the semiglobal optimization module. P_D^{sgbm} and P_R are parameterized.

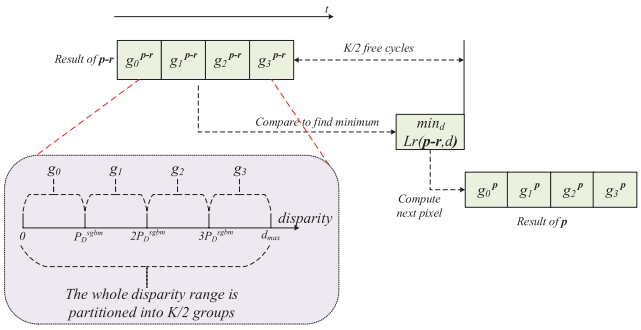


Fig. 12. Dataflow of the semiglobal computing module. Here we take $K = N_D/P_D = 8$ as an example.

compute the initial cost for $d = 2 \times P_D + 2$ with the output of the left buffer and the two cycle-delayed output of the right buffer. In our structure, we delay the output of the right buffer for 0 to $P_D - 1$ cycles with a shift register array, as shown in Fig. 9. Thus, the costs for $d = k \times P_D, k \times P_D + 1, \ldots, k \times P_D + P_D - 1$ are computed in parallel in the kth path. We could go through all the costs for $d = 0, 1, \ldots, N_D - 1$ in K passes.

D. Cost Aggregation

To solve the memory utilization problem caused by the cross-based support region, we adopt the inverted aggregation sequence method, which is proposed in [23]. To accelerate cost aggregation, Zhang *et al.* [20] propose the integral image, which reuses data between neighboring rows. Then, Shan *et al.* [23] implement a hierarchal vertical aggregation structure on FPGA to compute the integral image when doing vertical aggregation. We also use this structure for fast and efficient implementation.

The structure of the cost aggregation is shown in Fig. 10. A total of P_D aggregation modules are generated to deal with P_D disparities in parallel. In each module, P_R pixels are processed in parallel. The initial costs are aggregated first vertically and then horizontally. During the aggregation, the pixel number in the support region is recorded. All aggregated costs need to be divided by this pixel number for normalization.

E. Semiglobal Optimization With Hybrid Parallelism

To match the optimized hybrid parallelism scheme, we propose a new semiglobal optimization structure that supports both disparity-level and row-level parallelism.

When implementing the disparity-level parallelism, the data dependency is a problem for pipelined processing. As shown in (8), the path costs $L_{\boldsymbol{r}}(\boldsymbol{p})$ depends on the minimum of $L_{\boldsymbol{r}}(\boldsymbol{p} - \boldsymbol{r})$. In the direction *right*, pixel $\boldsymbol{p}$ is processed right after $\boldsymbol{p} - \boldsymbol{r}$. The minimum of N_D costs could not be computed in one cycle. So there must be some additional free cycles between the output of these two adjacent pixels.

In the proposed parallelism scheme, the computation of each pixel costs $K = N_D/P_D$ cycles. If the disparity range is still partitioned into K groups and processed in pipeline, there will be no free cycles. Thus, we increase the disparity-level parallelism degree of the semiglobal optimization module to solve this problem. The semiglobal disparity-level parallelism degree is represented as P_D^{sgbm} and is currently set as $2 \times P_D$. The corresponding structure and dataflow are shown in Figs. 11 and 12. The whole disparity range is partitioned into $K/2$ groups. Thus, there are $K/2$ cycles between the output of adjacent pixels. These free cycles are used to compute $\min_d L_{\boldsymbol{r}}(\boldsymbol{p} - \boldsymbol{r}, d)$. This computation is processed by the hardware comparison logic units in Fig. 11. The minimum of the whole disparity range $\min_d L_{\boldsymbol{r}}(\boldsymbol{p} - \boldsymbol{r}, d)$ could be ready at the Kth cycle. The path costs for pixel $\boldsymbol{p}$ could be computed in the following $K/2$ cycles.

The row-level parallelism degree of the semiglobal optimization module is equal to P_R in Section IV-B. To implement the row-level parallelism, we also need to handle the data access problem. The optimized costs $L_{\boldsymbol{r}}(\boldsymbol{p}, d)$ of the upper row are buffered as the input to the current row in the directions $\boldsymbol{r}_0$, $\boldsymbol{r}_1$, and $\boldsymbol{r}_2$ in Fig. 6. For different directions, the path costs of the upper row are delayed for different cycles.

V. Experimental Results

A. Experimental Setup

Fig. 13 shows the architecture of the proposed hardware-accelerated stereo vision system. The system is

TABLE II
MAXIMUM PROCESSING SPEED AND CORRESPONDING RESOURCE UTILIZATION

FPGA Platform	Image resolution	Frame rate	Module	Resource utilization		
				ALUTs [1]	Registers	RAM bits
Altera EP4SGX230	1024*768@96 disparities	67.82fps	Stereo matching core	125,255	81,092	9,282,494
			Whole system	137,425	86,639	9,387,470
Altera 5SGSMD5K2	1600*1200@128 disparities	42.61fps	Stereo matching core	222,034	149,288	16,604,247
			Whole system	236,498	155,992	17,071,191

[1] The Quartus software usually reports resource utilization with ALMs for Stratix V devices. The ALUTs utilization is also reported in the resource section. Here we show the ALUTs utilization to match the Stratix IV devices.

TABLE III
OVERALL COMPARISON OF THE FPGA-BASED STEREO VISION SYSTEMS

	Image Size	Disparity Levels	FPS[1]	MDE/s[2]	MDE/s/KLCs[3]	Error Rate	Algorithm
Shan et al. [14]	1280 × 1024	256	46	15437	622.7	17.3%	Local SAD-based block matching
MCADSR [23]	1024 × 768	128	129	13076	217.7	7.65%	Cross-Based local stereo matching
Proposed	**1600 × 1200**	**128**	**42.61**	**10472**	**47.2**	**5.61%**	**Cost aggregation + Semi-global**
Ambrosch et al. [25]	450 × 375	100	599	10125	168.75	all=31.5%	Local SAD-based block matching
Jin et al. [15]	1024 × 768	60	199.3	9362	76.2	6.05%	Fast locally consistent dense stereo
Jin et al. [11]	640 × 480	64	230	4522	75.4	17.2%	Local census-based block matching
Banz et al. [3]	640 × 480	128	103	4050	42.1	nonocc = 6.7%	Semi-global matching
Zhang et al. [13]	1024 × 768	64	60	3019	89.7	8.2%	Cross-Based local stereo matching
Tree structured DP[4][18]	640 × 480	60	58.7	1082	9.86	8.71%	Dynamic programming

[1] FPS: Frame per second.
[2] MDE/s: Million disparity estimation per second.
[3] KLCs: Kilo LUTs (for Xilinx FPGAs) or Kilo LEs (for Altera FPGAs).
[4] DP: Dynamic programming.

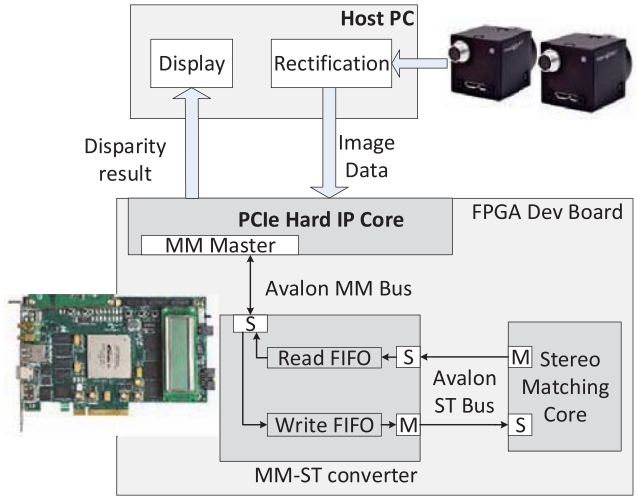


Fig. 13. Architecture of the demo system.

composed of a host PC, an FPGA board, and two Flea3 cameras [25]. The host PC gets the stereo image data streams from the Flea3 cameras and sends them to FPGA through the Peripheral Component Interconnect Express interface. The rectification is also processed in the software because modern PCs are capable of doing real-time rectification. On the FPGA board, we build a hardware-accelerated stereo matching core to compute the disparity map. The final disparity map is read out from the FPGA board and displayed by the host PC.

We evaluate both the processing speed and the accuracy performance to prove the effectiveness of the proposed stereo vision system. The first version of the system is based on an Altera EP4SGX230 FPGA. We further migrate the system to the Altera 5SGSMD5K2 FPGA and get higher processing ability. The system could run up to 180 MHz on both of the two platforms.

B. Overall Performance Evaluation

Table III makes an overall comparison among the state-of-the-art FPGA-based stereo vision systems. When evaluating FPGA-based stereo vision systems, we mainly focus on three aspects: 1) depth quality; 2) processing speed; and 3) resource utilization. The depth quality is measured by error rate, which means the rate of mismatched pixels. The error rate of the proposed system is 5.61%, which ranks first among the FPGA-based stereo vision systems. It will be discussed in detail in Section V-D.

The processing speed is evaluated by million disparity estimations per second (MDEs/s), which means the product of *image size, disparity levels, and frame rate*. Table II shows the maximum processing speed and the corresponding resource utilization of the system. On Altera EP4SGX230 FPGA, we could achieve a processing speed of 67.82 frames/s for 1024 × 768 images with 96 disparities. On the larger Altera 5SGSMD5K2 FPGA, the processing speed improves and we could achieve 42.61 frames/s for 1600 × 1200 images with 128 disparities. The proposed system could achieve 10 472 MDE/s on Altera 5SGSMD5K2 FPGA, which ranks high in the comparison. Some systems [14], [15], [23], [26] achieve a similar or even better processing speed. But their algorithm is simpler and the accuracy is worse than those of our system. The systems in [14] and [26] are all based on a fixed support window, which leads to low accuracy. The systems in [15] and [23] improve the accuracy with variable

TABLE IV
RESOURCE UTILIZATION UNDER DIFFERENT PARAMETER SETTINGS

Image resolution	Settings P_D	Settings P_R	Frame rate	Resource utilization ALUTs	Registers	RAM bits
640*480@	4	1	31.17fps	61,221	23,396	3,813,439
64 disparities,	4	2	62.33fps	64,878	30,796	4,193,550
W_{seg}=160	8	2	122.1fps	92,249	52,537	4,156,903
1024*768@	4	2	17.19fps	67,233	31,818	7,918,498
96 disparities,	8	2	33.91fps	94,464	54,133	7,827,966
W_{seg}=256	8	4	67.82fps	125,255	81,092	9,282,494
1600*1200@	8	2	10.85fps	94,647	54,871	14,051,519
128 disparities,	8	4	21.70fps	126,724	82,684	16,758,556
W_{seg}=400	16	4	42.61fps	222,034	149,288	16,604,247

TABLE V
RELATION BETWEEN RESOURCE UTILIZATION AND PARAMETERS

Modules	Number of ALUTs	Number of RAM bits
Cost initialization	$P_D * P_R * 120 + P_D * 2,900$[1]	$(2 * P_R + 24) * W * 96$
Cost aggregation	$P_D * P_R * 430 + P_D * 1,600$	0
Semi-global optimization	$P_D * P_R * 1,600$	$P_R * N_D * W_{seg} * 45$
Other Modules	$40,000$	$W * 2,200$

[1] These numerical values in the table are based on the compilation results.

support region or fast locally consistent algorithm. But their accuracy is still worse than that of our system because their algorithm is based on a local stereo matching method.

The resource utilization is another important criterion especially when the resource is limited. Thus, a normalized processing speed, which is measured by MDEs/s/KLCs, is also shown in Table III. It is an estimated evaluation because the basic units of different FPGA platforms are not the same. We could see that the performance of the proposed system is worse than those of the systems based on local methods. It is intuitive because we achieve the highest accuracy among these platforms. Besides, our primary goal is to improve the depth quality running in a real-time speed (30 frames/s). We believe higher resource will be provided and the accuracy will become more important.

A unified figure of metric (FOM) which includes all the mentioned aspects is quite helpful for the evaluation. However, it is difficult because the depth quality and the processing speed are in quite different domains. Besides, the importance of these two aspects varies over the requirement of the specific application. Thus, we think it is more meaningful to discuss the FOM in a specific scenario, and it is an open question for domain expertise.

C. Scalability

Another key feature of the proposed system is scalability. As shown in Table II, the stereo matching core occupies most of the hardware resource in the whole system. To make the system more flexible, the stereo matching core is fully parameterized to build a scalable hardware design. Thus, we could make a tradeoff between processing speed and resource utilization by tuning the parameters. The processing speed and resource utilization are mainly determined by the image size, disparity range, disparity-level parallelism P_D, row-level parallelism P_R, and segment width W_{seg}. Table IV shows the resource utilization of the stereo matching core with different parameters.

We further analyze the relationship between the resource utilization and parameter settings, as shown in Table V. Parameters P_D, P_R, N_D, H, W, and W_{seg} are considered here. The other parameters such as the cost width DW_{cost} and the maximum arm length $L_{\max}$ are fixed under different workloads ($\text{DW}_{\text{cost}} = 8$, $L_{\max} = 12$ in current system), and thus, they are not listed for simplification. The values in Table V are based on the compilation results of each module. Due to unpredictable optimization in compilation tools, the real resource utilization of the whole core may be a little different from the estimated value. The total resource utilization could be estimated with (11). When the total parallelism degree $P_D \times P_R$ is fixed, the logic resource utilization mainly depends on P_D, while the memory resource utilization mainly depends on P_R. The processing speed (frame per second) is shown in (12). The parameters could be adjusted according to the resource specification

$$\begin{aligned} \text{ALUTs} &\approx P_D \times P_R \times 2,200 + P_D \times 4,500 + 40,000 \\ \text{RAM bits} &\approx P_R \times (W \times 192 + N_D \times W_{\text{seg}} \times 45) + W \times 4,500 \end{aligned} \tag{11}$$

$$\text{FPS} = \frac{H \times W \times N_D}{P_D \times P_R} \times \frac{W_{\text{seg}}}{W_{\text{seg}} + P_D + 2 \times L_{\max}}. \tag{12}$$

D. Quality Evaluation

The depth quality of the proposed system is discussed in this section. We evaluate the system with both benchmark and real-world scenarios. The experiment shows that our system could provide accurate disparity maps in different scenarios and different resolutions.

1) On the Middlebury Benchmark: The Middlebury benchmark [4] is widely used in evaluating the quality of stereo matching algorithms. The four image pairs *tsukuba*, *venus*, *teddy*, and *cones* are processed using the proposed system and the results are shown in Fig. 14. The average percentage of bad pixels in the disparity map is 5.61%. We compare the proposed system with some state-of-the-art stereo vision systems and list the results in Table VI. The accuracy of the proposed system is the best of the hardware-accelerated stereo vision systems. The first is the original AD-Census implementation on GPU. AD-Census uses many promising technologies to achieve the best accuracy on the Middlebury benchmark. It requires large resources to implement all functions for AD-Census. The system in [15] shows comparable accuracy performance. But they use a local stereo matching method and the accuracy may drop with the increased image resolution.

The resolutions of the data set *tsukuba*, *venus*, *teddy*, and *cones* are all smaller than that of VGA. However, our system could process high-definition images up to 1600×1200@128 disparities. To evaluate the tolerance for resolution changes, we further process some high-definition images in the benchmark, including data set *Dolls*, *Baby1*,

TABLE VI
COMPARISON OF ACCURACY RESULT ON MIDDLEBURY BENCHMARK

	Platform	Tsukuba			Venus			Teddy			Cones			Average Percent
		nonocc[1]	all[2]	disc[3]	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc	Bad Pixels
ADCensus [5]	GPU	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
PatchMatch [27]	CPU	2.09	2.33	9.31	0.21	0.39	2.62	2.99	8.16	9.62	2.47	7.80	7.11	4.59
Proposed	**FPGA**	**2.39**	**3.27**	**8.87**	**0.38**	**0.89**	**1.92**	**6.08**	**12.1**	**15.4**	**2.12**	**7.74**	**6.19**	**5.61**
Jin et al. [15]	FPGA	1.66	2.17	7.64	0.40	0.60	1.95	6.79	12.4	17.1	3.34	8.97	9.62	6.05
SemiGlobal [7]	CPU	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
Banz et al. [3]	FPGA	4.1	-	-	2.7	-	-	11.4	-	-	8.4	-	-	nocc. = 6.7
Variable Cross [20]	CPU	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
Variable Cross GPU [28]	GPU	1.71	2.22	6.74	0.55	0.87	2.88	9.90	15.0	19.5	6.66	12.3	13.4	7.65
MCADSR [23]	FPGA	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]	FPGA	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
Tree structured DP [18]	FPGA	1.43	2.51	6.60	2.37	2.97	13.1	8.11	13.6	15.5	8.12	13.8	16.4	8.71
Jin et al. [11]	FPGA	9.79	11.6	20.3	3.59	5.27	36.8	12.5	21.5	30.6	7.34	17.6	21.0	17.2
Shan et al. [14]	FPGA	-	24.5	-	-	15.7	-	-	15.1	-	-	14.1	-	all = 17.3
Chang et al. [29]	DSP	20.4	20.6	47.9	15.3	16.6	29.5	25.1	32.4	34.1	22.9	31.1	30.6	27.2

[1] nonocc: Average percentage of bad pixels in non-occluded regions.
[2] all: Average percentage of bad pixels in all regions.
[3] disc: Average percentage of bad pixels in discontinuous regions.

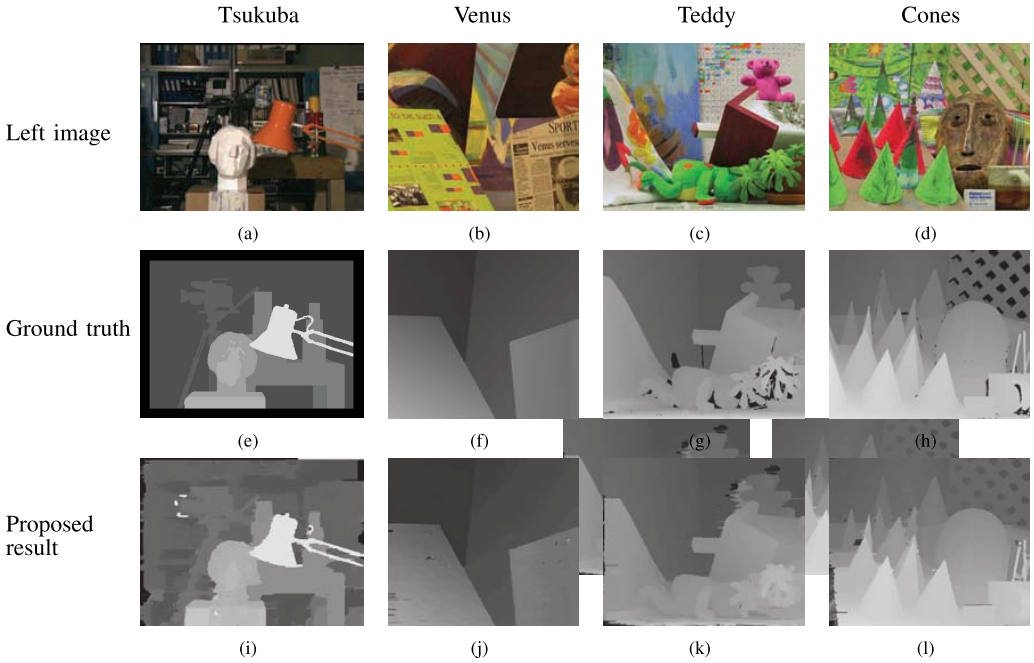


Fig. 14. Evaluation results on Middlebury benchmark. (a)–(d) Left images (data set *tsukuba*, *venus*, *teddy*, and *cones*, respectively). (e)–(h) Corresponding ground truth. (i)–(l) Depth maps provided by the proposed system.

and *Cloth4*. For each data set, the Benchmark provides seven pictures captured at different viewpoints. We select view 1 as the left image and view 3 as the right image to make sure that the disparity range is within 128 levels. The result disparity maps are shown in Fig. 15. Although the parameters are tuned for the low-definition images, the proposed system still provides clear and smooth disparity maps in high-definition scenarios. The disparity error rate in *all* regions are also listed in Fig. 15. The numerical depth accuracy is also comparable with the low-definition results.

2) On Real-World Scenarios: The evaluation of the Middlebury benchmark proves that the results of the proposed system are quite accurate. However, the original images in the benchmark are all well captured and rectified. In real-world scenarios, the stereo images may not be as high-quality as the benchmark. Thus, the depth accuracy may decrease due to some nonideal factors such as luminance differences and rectification error. To prove its robustness in real-world scenarios, the proposed system is further evaluated by the images from Flea3 cameras and online 3-D videos, as shown in Fig. 16. Shown in Fig. 16(a) and (b) are the images captured by the Flea3 cameras in the demo system with a resolution of 1600×1200. Fig. 16(c) is one frame of a 3-D video [21], which is captured by a Sony HDR-TD10 Video Camera in a Natural History Museum with a resolution of 1280×720. Shown in Fig. 16(d)–(f) are the proposed depth results. We could see that our system still provides clear disparity maps in real-world scenarios.

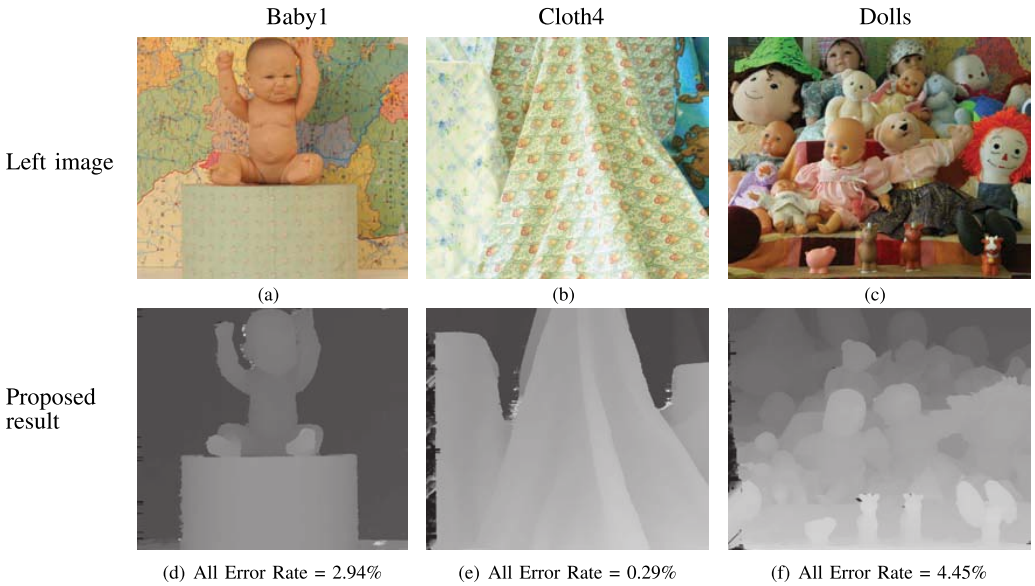


Fig. 15. Evaluation results on high-definition images. (a)–(c) Original images from the Middlebury benchmark. (d)–(f) Disparity maps of the proposed system.

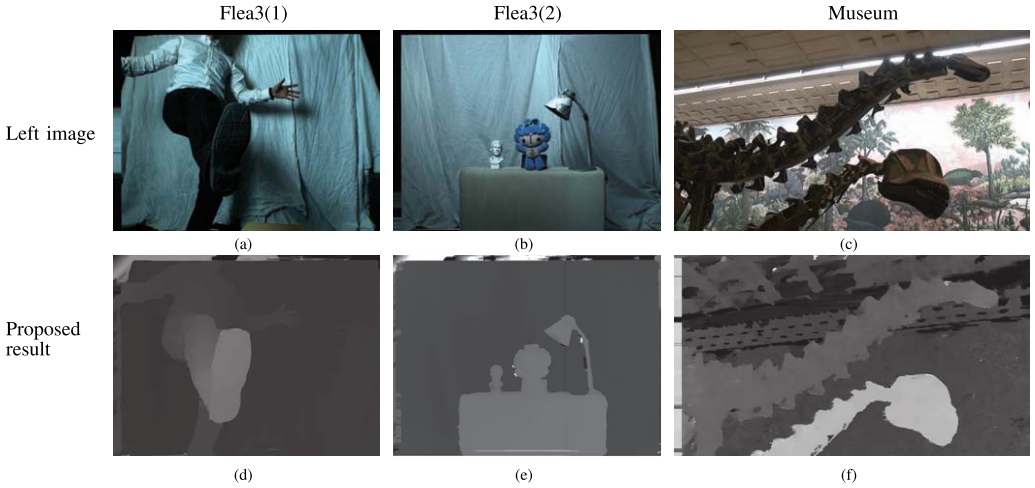


Fig. 16. Evaluation results in real-world scenarios. (a) and (b) Captured by Flea3 cameras. (c) Online 3-D video. (d)–(f) Disparity maps of the proposed system.

VI. Conclusion

In this paper, we propose a stereo vision system based on an FPGA accelerator. The proposed algorithm is from top-performing stereo matching algorithms on Middlebury benchmarks. We design a scalable architecture for the implementation of key functions and build prototypes using Altera boards. Its quality is better than that of existing hardware stereo vision systems. Moreover, the processing ability of our system is among the best of current hardware implementations. Our system is the first complete work on FPGA that supports aggregation on a cross-based region (it also could be a fixed window or a variable window) and semiglobal optimization. The depth quality is evaluated both on Middlebury benchmarks and real-world scenarios. The results show that our implementation has the best performing stereo matching accuracy on the Middlebury Benchmark and the top-performing processing ability. The system could be tailored to different solutions according to the application requirements. In the future, we intend to start its application-specified integrated circuit design for lower cost and power. This could make more mobile scenarios solvable.

References

[1] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *Int. J. Comput. Vis.*, vol. 47, nos. 1–3, pp. 7–42, 2002.

[2] H. Hirschmuller and D. Scharstein, "Evaluation of cost functions for stereo matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2007, pp. 1–8.

[3] 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 *Proc. Int. Conf. Embedded Comput. Syst. (SAMOS)*, Jul. 2010, pp. 93–101.

[4] *Vision.middlebury.edu/stereo.*. [Online]. Available: http://vision.middlebury.edu/stereo/, accessed May 2015.

[5] X. Mei, X. Sun, M. Zhou, S. Jiao, H. Wang, and X. Zhang, "On building an accurate stereo matching system on graphics hardware," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCV Workshops)*, Nov. 2011, pp. 467–474.

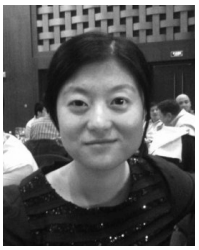
[6] R. Yang, G. Welch, and G. Bishop, "Real-time consensus-based scene reconstruction using commodity graphics hardware," *Comput. Graph. Forum*, vol. 22, no. 2, pp. 207–216, 2003.

[7] H. Hirschmuller, "Stereo processing by semiglobal matching and mutual information," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 2, pp. 328–341, Feb. 2008.

[8] Q. Yang, L. Wang, R. Yang, S. Wang, M. Liao, and D. Nistér, "Real-time global stereo matching using hierarchical belief propagation," in *Proc. BMVC*, vol. 6. 2006, pp. 989–998.
[9] L. Wang, M. Liao, M. Gong, R. Yang, and D. Nister, "High-quality real-time stereo using adaptive cost aggregation and dynamic programming," in *Proc. 3rd Int. Symp. 3D Data Process., Vis., Transmiss.*, Jun. 2006, pp. 798–805.
[10] I. D. Rosenberg, P. L. Davidson, C. M. R. Muller, and J. Y. Han, "Real-time stereo vision using semi-global matching on programmable graphics hardware," in *Proc. ACM SIGGRAPH Sketches*, 2006, Art. ID 89.
[11] S. Jin *et al.*, "FPGA design and implementation of a real-time stereo vision system," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 1, pp. 15–26, Jan. 2010.
[12] E. Gudis, G. van der Wal, S. Kuthirummal, and S. Chai, "Multi-resolution real-time dense stereo vision processing in FPGA," in *Proc. IEEE 20th Annu. Int. Symp. Field-Program. Custom Comput. Mach. (FCCM)*, Apr./May 2012, pp. 29–32.
[13] L. Zhang, K. Zhang, T. S. Chang, G. Lafruit, G. K. Kuzmanov, and D. Verkest, "Real-time high-definition stereo matching on FPGA," in *Proc. 19th ACM/SIGDA Int. Symp. Field Program. Gate Arrays*, 2011, pp. 55–64.
[14] Y. Shan *et al.*, "FPGA based memory efficient high resolution stereo vision system for video tolling," in *Proc. Int. Conf. Field-Program. Technol. (FPT)*, Dec. 2012, pp. 29–32.
[15] M. Jin and T. Maruyama, "Fast and accurate stereo vision system on FPGA," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 7, no. 1, 2014, Art. ID 3.
[16] S. Park and H. Jeong, "Real-time stereo vision FPGA chip with low error rate," in *Proc. Int. Conf. Multimedia Ubiquitous Eng. (MUE)*, Apr. 2007, pp. 751–756.
[17] S. Sabihuddin, J. Islam, and W. J. MacLean, "Dynamic programming approach to high frame-rate stereo correspondence: A pipelined architecture implemented on a field programmable gate array," in *Proc. Can. Conf. Elect. Comput. Eng. (CCECE)*, May 2008, pp. 001461–001466.
[18] M. Jin and T. Maruyama, "A real-time stereo vision system using a tree-structured dynamic programming on FPGA," in *Proc. ACM/SIGDA Int. Symp. Field Program. Gate Arrays*, 2012, pp. 21–24.
[19] S. Gehrig, F. Eberli, and T. Meyer, "A real-time low-power stereo vision engine using semi-global matching," in *Computer Vision Systems* (Lecture Notes in Computer Science), vol. 5815, M. Fritz, B. Schiele, and J. H. Piater, Eds. Berlin, Germany: Springer-Verlag, 2009, pp. 134–143.
[20] K. Zhang, J. Lu, and G. Lafruit, "Cross-based local stereo matching using orthogonal integral images," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 7, pp. 1073–1079, Jul. 2009.
[21] *Sony TD10 3D Video: Natural History Museum in 3D!* [Online]. Available: http://www.youtube.com/watch?v=tQ_8xlxJQCM, accessed May 2015.
[22] R. Zabih and J. Woodfill, "Non-parametric local transforms for computing visual correspondence," in *Computer Vision—ECCV*. Berlin, Germany: Springer-Verlag, 1994, pp. 151–158.
[23] Y. Shan *et al.*, "Hardware acceleration for an accurate stereo vision system using mini-census adaptive support region," *ACM Trans. Embedded Comput. Syst.*, vol. 13, no. 4s, 2014, Art. ID 132.
[24] W. Wang, J. Yan, N. Xu, Y. Wang, and F.-H. Hsu, "Real-time high-quality stereo vision system in FPGA," in *Proc. Int. Conf. Field-Program. Technol. (FPT)*, Dec. 2013, pp. 358–361.
[25] *Flea3 GigE Vision (Gigabit Ethernet) Cameras for Industrial, Life Science, Traffic, and Security Applications*. [Online]. Available: http://www.ptgrey.com/flea3-gige-vision-cameras, accessed May 2015.
[26] K. Ambrosch, M. Humenberger, W. Kubinger, and A. Steininger, "SAD-based stereo matching using FPGAs," in *Embedded Computer Vision*. London, U.K.: Springer-Verlag, 2009, pp. 121–138.

Wenqiang Wang received the B.S. degree in electronics engineering from Tsinghua University, Beijing, China, in 2012, where he is currently working toward the M.S. degree with the Department of Electronic Engineering.

His research interests include application specific hardware computing in image processing field.



Jing Yan received the B.S. and M.S. degrees from Tsinghua University, Beijing, China, in 2007 and 2010, respectively.

She is with the Hardware Computing Group, Microsoft Research Asia, Beijing, as an Assistant Researcher. Her research interests include hardware accelerator system design, reconfigurable computing, and application specific hardware computing.

Ningyi Xu received the B.S. and Ph.D. degrees from Tsinghua University, Beijing, China, in 2001 and 2006, respectively.

He is a Researcher with the Hardware Computing Group, Microsoft Research Asia, Beijing. His research interests include heterogeneous computing for applications and services in data centers.

Yu Wang (S'05–M'07–SM'14) received the B.S. and Ph.D. (Hons.) degrees from Tsinghua University, Beijing, China, in 2002 and 2007, respectively.

He is an Associate Professor with the Department of Electronic Engineering, Tsinghua University. He has authored or co-authored over 120 papers in refereed journals and conferences. His research interests include parallel circuit analysis, application specific hardware computing (in particular, on brain-related problems), and power/reliability aware system design methodology.

Dr. Wang received the IBM X10 Faculty Award in 2010, the Best Paper Award in the IEEE Computer Society Annual Symposium on VLSI (ISVLSI) in 2012, the Best Poster Award in the Hardened Electronics and Radiation Technology Conference in 2012, and six best paper nominations in the Asia and South Pacific Design Automation Conference (ASPDAC)/the International Conference on Hardware/Software Codesign and System Synthesis/the International Symposium on Low Power Electronics and Design (ISLPED). He is an Associate Editor of IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN and *Journal of Circuits, Systems, and Computers*. He was the Technical Program Committee (TPC) Co-Chair of the International Conference on Field Programmable Technology (ICFPT) in 2011. He was the Finance Chair of ISLPED from 2012 to 2015, and serves as a TPC Member in many important conferences, such as the Design Automation Conference, the International Symposium on Field-Programmable Gate Arrays, the Design Automation and Test in Europe, ASPDAC, ISLPED, the International Symposium on Quality Electronic Design, ICFPT, and ISVLSI.

Feng-Hsiung Hsu received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, and the Ph.D. degree in computer science from Carnegie Mellon University, Pittsburgh, PA, USA.

He was with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA; Western Research Laboratory (Compaq), Palo Alto, CA, USA; and HP Research Laboratory, Palo Alto. He currently manages the Hardware Computing Group with Microsoft Research Asia, Beijing, China, where he is involved in both cloud and device related research. His current research interests include computer architecture, very-large-scale integration design, field-programmable gate arrays systems, parallel algorithms, sensors, actuators, displays, and Internet of Things devices.

Dr. Hsu received the ACM's Grace Murray Hopper Award for his work on Deep Thought, the first machine to play chess at Grandmaster level. He received the Fredkin Prize, along with M. S. Campbell and A. J. Hoane, for building Deep Blue, the first machine to defeat the world chess champion in a set match.