Hardware Acceleration for Accurate Stereo Vision System using Mini-Census Adaptive Support Region

YI SHAN, YUCHEN HAO, WENQIANG WANG,
YU WANG, XU CHEN AND HUAZHONG YANG, Tsinghua University
WAYNE LUK, Imperial College London

Domain of stereo vision is highly important in the fields of autonomous cars, video tolling, robotics, and aerial surveys. The specific feature of this domain is that we should handle not only the pixel-by-pixel 2D processing in one image but also the 3D processing for depth estimation by comparing information about a scene from several images with different perspectives. This feature brings challenges to memory resource utilization, because extra dimension of data has to be buffered. Due to the memory limitation, few of previous stereo vision implementations provide both accurate and high-speed processing for high resolution images at the same time.

To achieve domain-specific acceleration for stereo vision, the memory limitation has to be addressed. This paper uses a Mini-Census Adaptive Support Region (MCADSR) stereo matching algorithm as a case study due to its high accuracy and representative operations in this domain. To relieve the memory limitation and achieve high-speed processing, the paper proposes several efficient optimization methods including vertical-first cost aggregation, hybrid parallel processing, and hardware-friendly integral image. The paper also presents a customizable system, which provides both accurate and high-speed stereo matching for high resolution images. The benefits of applying the optimization methods to the system are highlighted.

With the above optimization and specific customization implemented on FPGA, the demonstrated system can process 47.6 fps (frames per second) and 129 fps for video size of 1920 × 1080 with a large disparity range of 256 and 1024 × 768 with a disparity range of 128 respectively. Our results are up to 1.64 times better than previous work in terms of million disparity estimation per second (MDE/s). For accuracy, the 7.65% overall average error rate outperforms current work which can provide real-time processing with this high resolution and large disparity range.

Categories and Subject Descriptors: B.5.1 [**Register-Transfer-Level Implementation**]: Design; I.4.8 [**Image Processing and Computer Vision**]: Scene Analysis-Stereo

General Terms: Design, Performance

Additional Key Words and Phrases: Stereo vision, hardware acceleration, parallelism, integral image, FPGA

ACM Reference Format:
Y. Shan, Y. Hao, W. Wang, Y. Wang, W. Luk, X. Chen and H. Yang. 2013. Hardware Acceleration for Accurate Stereo Vision System using Mini-Census Adaptive Support Region. *ACM Trans. Embedd. Comput. Syst.* V, N, Article A (January YYYY), 24 pages.
DOI:http://dx.doi.org/10.1145/0000000.0000000

Submitted to DSMC12 Special Issue.
This work was supported by 973 project 2013CB329000, National Science and Technology Major Project (2013ZX03003013-003), National Natural Science Foundation of China (No. 61373026, No. 61261160501, No. 61028006), and Tsinghua University Initiative Scientific Research Program.
Author's addresses: Y. Wang is the corresponding author. Y. Shan, Y. Hao, W. Wang, Y. Wang, X. Chen and H. Yang, Department of Electronic Engineering, Tsinghua National Laboratory for Information Science and Technology, Tsinghua University, Beijing 100084, China; e-mail: yu-wang@mail.tsinghua.edu.cn. W. Luk, Department of Computing, Imperial College London, London SW7, U.K.; e-mail: wl@doc.ic.ac.uk.

1. INTRODUCTION

STEREO VISION extracts 3D information from digital images. By comparing two differing views on a scene obtained from two horizontally dispatched cameras, the relative depth information can be obtained in the form of disparities, which are inversely proportional to the distance to the objects. Stereo vision is widely used in embedded scenarios, such as autonomous cars, video tolling, robotics, and aerial surveys. However, stereo vision poses a significant implementation problem because of an extra dimension of computation for 3D information compared with conventional 2D image processing. Due to the design complexity and resource limitation in embedded scenarios, implementations on general-purpose processors usually struggle to meet both accurate and high-speed requirements under high resolution and large disparity range constraints.

Domain-specific acceleration is tailored for one application domain, and thus, can enhance performance of the system under resource limitation. For stereo vision acceleration, the biggest challenge is the memory limitation caused by the extra dimension of data storage. To address this challenge, some optimization methods and a high throughput customizable platform should be used to satisfy the accurate and high-speed processing requirements. At the same time, the deployment and updating cost should be carefully considered.

Realizing stereo vision system on GPUs is easier than on other customizable platforms [De-Maeztu et al. 2012][Yang et al. 2006][Wang et al. 2006]. However, it is still hard to achieve real-time processing because of internal memory bandwidth limitation. Besides, GPUs are not feasible for embedded applications due to its high power consumption.

FPGAs are promising platforms with strong customizable computation power and relatively low power consumption. Existing efforts of accelerating stereo vision system on FPGAs mostly choose the regular algorithms with fixed data access and operation pattern to utilize the advantage of FPGAs. [Shan et al. 2012] [Chang et al. 2007] [Ambrosch et al. 2009] [Jin et al. 2010] are all based on the fixed rectangle support region and the accuracy is very low. [Zhang et al. 2011] proposes a structure for adaptive support region and achieves high-speed processing. However, the 2-D adaptive support region is simplified to 1-D adaptive in their work, which also causes a decrease in accuracy. [Jin and Maruyama 2012b] proposes a tree-structure implementation based on the global optimization and gets a good accuracy. However, real-time processing will become difficult for these efforts when dealing with larger image and larger disparity range because of memory resource limitation. Thus, ”how to optimize the system according to the domain-specific features so as to reduce the memory resource consumption to make the system realizable” is an emerging problem.

This paper presents an FPGA based stereo vision system, which provides both accurate and high-speed processing for high resolution images. The implementation uses a Mini-Census Adaptive Support Region (MCADSR) algorithm which provides accurate matching results. To solve the memory limitation problem caused by 3D information computation in the algorithm, we propose three domain-specific optimization methods including vertical-first aggregation, hybrid parallel processing and hardware-friendly integral image. Our main contributions are:

(1) vertical-first cost aggregation which significantly reduces the memory resource consumption;
(2) hybrid parallel processing which efficiently reuses the intermediate results and provides a trade-off between different on-chip resources;
(3) hardware-friendly integral image which uses hierarchical adder tree for the vertical aggregation and shifter for the horizontal aggregation separately;

(4) an accurate and high-speed stereo vision system on FPGA using MCADSR algorithm and above optimizations, which can achieve on average 7.65% error rate. The processing speed is 47.6 fps and 129 fps for image size of 1920×1080 with a disparity range of 256 and 1024×768 with a disparity range of 128, respectively.

The remainder of this paper is organized as follows. Section 2 introduces background and previous work about stereo vision. Section 3 gives the detail of the MCADSR stereo matching algorithm. Section 4 illustrates our proposed optimization methods. Section 5 introduces the hardware implementation on FPGA. The experimental results are presented in Section 6. Section 7 concludes this work.

2. BACKGROUND & RELATED WORK

2.1. Stereo Matching background

Stereo matching algorithms or disparity estimation 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. In remedy of this, most stereo matching algorithms employ camera calibration and image rectification, aligning each epipolar lines to a common axis and projecting each image to a common image plane. This ensures the stereo matching is reduced to a 1D searching problem along the same horizontal scanline of the image pair.

Given two calibrated and rectified images, the problem of stereo matching can be addressed by finding the corresponding pixel in the right image for each and every pixel in the left. To avoid the problem caused by image noise, a support region is built for each pixel and the matching is carried out over these regions instead of pixel by pixel. Given a pixel $P(x, y)$ in the left image, its corresponding pixel $Q(x+d, y)$ is found 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 a disparity. Then the matching cost is computed over the support region of each pixel pair. The smaller the cost is, the more similar these support regions are. Hence, the corresponding pixel is defined as the anchor pixel of the support region with the minimal matching cost among all candidates.

Implementing such stereo matching algorithms on dedicated hardware systems has shown its potential over the years. By concurrently computing the matching cost of a range of disparity levels, these systems achieve high processing speed. However, the processing of each disparity level requires large amount of buffer for input images and intermediate results. As a result, the stereo matching module as a whole consumes significant amount of memory resource. Hence, relieving memory consumption is the major challenge for an accurate and efficient stereo matching system.

2.2. Related Work

Stereo matching algorithms typically fall into two different categories: global and local approaches. In global approaches, disparities are determined based on the optimization of a global energy equation, providing optimal results of all pixels. Local approaches determine disparities based on the consideration of pixels only within a support region, leading to reduced computation intensity and greater potential in real-time implementations.

Prior work has attempted to increase the stereo system throughput using dedicated hardware such as FPGAs and improve the disparity accuracy using more sophisticated algorithms. We discuss some of the related work in this section.

Local Stereo Algorithms Ambrosch et al. [2009] implements an SAD-based fixed-support algorithm using state-of-the-art FPGA device, achieving 599 FPS at an image size of 450×375 and a disparity range of 100. Shan et al. [2012] proposes a Hybrid-D box-filtering algorithm that combines the 1D and 2D algorithms to reduce the computation and memory cost, leading to lower memory consumption of $0.95Mbits$ and the performance of 46 FPS at 1280×1024 with a large disparity range of 256, and the estimation speed reaches up to $15437MDE/s$. Jin et al. [2010] adopts Census transform and computes the matching cost through the Hamming distance, which obtains significant increase in accuracy compared to SAD-based methods.

The majority of the above implementations adopt fixed-support regions to achieve real-time performance, as these algorithms can be greatly benefited by the use of parallel and straightforward structures, which are key factors available in dedicated hardware implementations such as FPGAs. However, these proposals have relatively high disparity error rates (usually larger than 15%).

The adaptive support weight (ADSW) algorithm are capable to deliver better matching accuracy. However, this algorithm involve complex and hardware-unfriendly operations and are computationally more expensive compared to fixed-support algorithms. As a result, ADSW algorithms have been rarely implemented in dedicated hardware. Chang et al. [2010] explores the potential of realizing an ADSW algorithm on VLSI architecture and successfully proposes an accurate, hardware-friendly disparity estimation algorithm called Mini-Census Adaptive Support Weight (MCADSW), leading to improved matching accuracy. However, their work achieves real-time performance for relatively small-sized images as $42FPS@352 \times 288$, which is insufficient to meet the high resolution requirement.

Another featured algorithm is adaptive support region (ADSR) algorithm, which is able to achieve an error rate of 7.60% [Zhang et al. 2009a]. Zhang et al. [2011] implements a Mini-Census adaptive support region algorithm on FPGA and obtains a high accuracy and fast estimation speed on high resolution images, which consumes $3.77Mbits$ memory. As mentioned above, they trades off the full adaptiveness of the algorithm in order to alleviate the complexity and achieve pipelining, resulting in increased disparity error rates as 8.20%. This ADSR algorithm can be viewed as a lite version of ADSW where the weight of neighboring pixels is either one or zero. Thus, these algorithms require only some additional optimization methods to be implemented compared to fixed-support ones while preserving the adaptiveness of ADSW algorithms.

Global Stereo Algorithms Dynamic Programming (DP) [Veksler 2005] [Wang et al. 2006] [Jin and Maruyama 2012b] is a technique that can offer optimized solution for independent scanlines in an efficient manner. As the DP algorithms optimize the disparity map on a scanline by scanline basis, the inter-scanline consistency is not enforced, leading to the well-known “streaking” artifacts. Jin and Maruyama [2012b] proposes a tree-structured DP algorithm aiming to address this issue (see Fig. 15). However, it can hardly be eliminated.

Belief Propagation (BP) [Klaus et al. 2006] [Yang et al. 2006] is a new global algorithm that has attracted much attention recently. It gathers information from neighboring pixels and incorporates the information to update a smoothness term between the pixel of interest and its neighborhood, and iteratively optimizes the smoothness term to achieve global energy minimization. Unlike scanline methods, these algorithms enforce optimization in two dimensions. Though some of the most impressive results are obtained [Klaus et al. 2006], real-time implementations using BP algorithm have never been achieved.

Jin and Maruyama [2012a] proposes a fast stereo matching system on FPGA. They introduces fast locally consistent (FLC) algorithm together with cost aggregation to improve accuracy. In considering the balance of processing speed and the circuit size, they limit the vertical span of support region to 11 and the disparity range to 60. However, the memory consumption is still as high as 198 block RAMs at 1024 wide

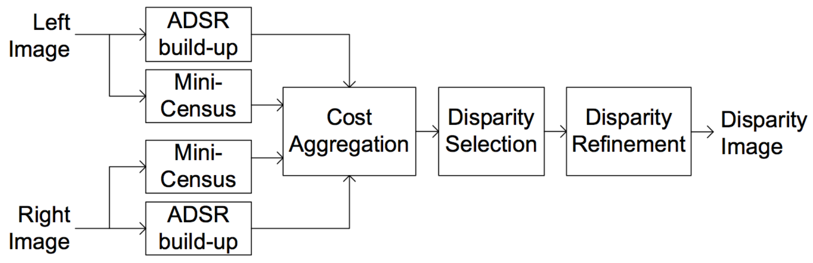


Fig. 1. Overall flow of proposed MCADSR algorithm

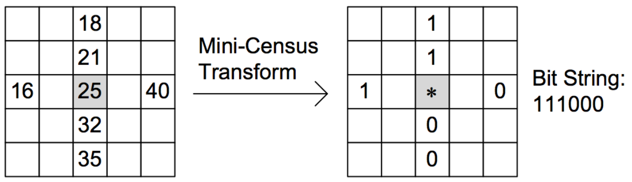


Fig. 2. The Mini-Census Transformation

images and the reduced disparity range is not sufficient for a image size of, if not larger than, 1024×768.

In all of the above global algorithms, high accuracy is achieved at the cost of low processing speed, which struggles to meet real-time processing. In contrast, our design goal is to provide real-time high-definition performance without compromising the accuracy. Hence, we propose a fully pipelined hardware implementation of a high-accuracy local stereo algorithm. The proposed algorithm and hardware design is illustrated in the following sections.

3. MINI-CENSUS ADAPTIVE SUPPORT REGION ALGORITHM

3.1. Overview

The selected local algorithm is Mini-Census Adaptive Support Region (MCADSR). This algorithm is effective and simple which can provide relatively accurate results only with the modification of adaptive support region instead of the rectangle one. Meanwhile, the optimization for aggregation of 2-D adaptive region can benefit many other stereo vision algorithms. The overall flow is shown in Fig. 1. The system takes two calibrated and rectified images as input and performs four main steps: Mini-Census transformation, adaptive support region build-up, cost aggregation, and disparity selection and refinement. First, Mini-Census transformation step locally transforms each pixel into a Mini-Census string. Second, adaptive support region is configured as a cross skeleton for each pixel based on the luminance similarity. In cost aggregation step, matching cost which is defined as the Hamming distance between two Mini-Census strings is aggregated on the support region for each pair of pixels under a certain range of disparities. In the end, the disparity with the minimum matching cost is selected and refined for each pixel so as to produce the final disparity map. The detail will be explained in the following subsections.

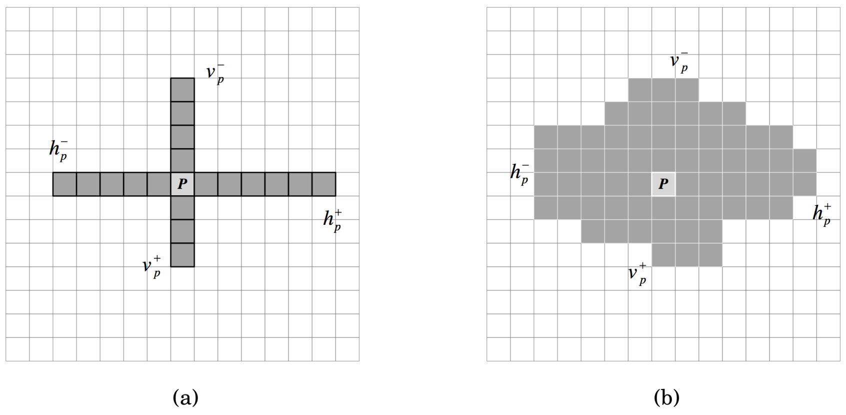


Fig. 3. (a) Configuration of an appropriate cross skeleton for pixel of interest P based on the local image structure. The quadruple $\{h_p^-, h_p^+, v_p^-, v_p^+\}$ denotes the left, right, up and bottom arm length, respectively. (b) Matching costs along each direction are aggregated on the entire support region in cost aggregation step.

3.2. Mini-Census Transformation

Census transformation converts the support region into a string of bits by intensity comparison between the center pixel (pixel of interest) and other pixels nearby. Census stereo matching [Zabih and Woodfill 1994] utilizes the bit string, achieved by census transformation, to represent the center pixel instead of its intensity. This method has been shown to provide very robust and accurate disparity results comparing with other local methods and is very well-suited to be implemented on FPGAs due to many bitwise operations. [Fife and Archibald 2013] analyzed a range of sparse census transforms and concluded that the correlation accuracy for the sparse transform is often better than or nearly as good as the full Census transform on standard image datasets.

The selected stereo matching method begins by applying Mini-Census transform which is also a sparse version of Census transform to both left and right images. It transforms 8-bit intensity of every pixel into a bit string where each bit corresponds to a certain pixel in the local neighborhood around a pixel of interest. As is shown in Fig. 2, a bit is set if the corresponding pixel has a lower 8-bit intensity than the pixel of interest. Thus, Census transform not only encodes the relative intensity of pixels, but also the spatial structure of the local neighborhood. This transformation reduces the on-chip storage and makes the stereo matching robust to radiometric changes [Hirschmuller and Scharstein 2009].

Once the input images have been transformed, stereo matching is typically performed by computing the sum of the Hamming distances over a support region, which will be defined in the following subsection.

3.3. Adaptive Support Region Build-up

Ideally a local support region should contain only the neighboring pixels from the same depth with the pixel of interest. Though disparity information is unavailable beforehand, the support region builder still should be aware of local image structures. We utilize the common assumption that pixels with similar intensity within a constrained area are likely from the same image structure and adopt the cross-based local support region proposed by [Zhang et al. 2009a] for accurate local stereo matching.

The key idea of this mechanism is to configure an appropriate cross skeleton for each pixel based on the local image structure (see Fig. 3(a)). The cross of each pixel of interest is defined by a quadruple $\{h_p^-, h_p^+, v_p^-, v_p^+\}$ that denotes the left, right, up and bottom arm length, respectively. Unlike fixed support method, this mechanism is able to exclude pixels from image structures which are different from the pixel of interest, avoiding *pollution* caused by those pixels and thus providing both sufficient and proper support pixels.

The arm lengths are decided upon an intensity testing for a consecutive set of pixels to search for the largest span on each direction. The computation of support arm length can be formulated as follows:

$$L^* = \max_{L\in[1,L\_MAX]} \left(L \prod_{i\in[1,L]} \delta(p, p_i) \right) \tag{1}$$

where L_MAX is the preset maximum arm length and $\delta(p_1, p_2)$ is an indicator function evaluating the intensity similarity between pixel p_1 and p_2

$$\delta(p_1, p_2) = \begin{cases} 1 & \text{for } I(p_1) - I(p_2) \le \tau \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $I(p)$ is the intensity of pixel p and τ is the similarity threshold.

Based on the cross skeleton, an arbitrarily shaped support region $U(p)$ can be constructed for pixel p, where $U(p)$ is defined as an integral of multiple horizontal segments $H(q)$ along the vertical segment $V(p)$, or multiple vertical segments $V(q)$ along the horizontal segment $H(p)$ of the pixel p. Aggregating matching costs over such regions involves irregular memory references, causing difficulty in meeting real-time performance for the entire stereo matching system. Our solution regarding the choice between the two aggregation sequences will be discussed in Section 4.

$$U(p) = \bigcup_{q\in V(p)} H(q) \qquad or \qquad U(p) = \bigcup_{q\in H(p)} V(q) \tag{3}$$

3.4. Cost Aggregation

In the cost aggregation step, the Hamming distance, $Ham(p, d)$, between the pixel p in the left transformed image and corresponding pixel in the right transformed image at the disparity of d, is calculated firstly. And then the matching cost of pixel p is calculated by aggregating the Hamming distance of the adaptive support region. Here, we adopt the orthogonal two-pass technique proposed in the Variable-Cross algorithm [Zhang et al. 2009a], which decomposes aggregation on a 2-D support region into a *vertical aggregation* and a *horizontal aggregation*. The technique firstly aggregates vertically (horizontally) to give a vertical (horizontal) aggregated cost, $VA(p, d)$ ($HA(p, d)$), of each column (row) firstly, and then the horizontal aggregated costs are aggregated horizontally (vertically) together to give the final matching cost $AggCost(p, d)$ at the disparity of d.

$$VA(p, d) = \sum_{q\in V(p,d)} Ham(q, d) \qquad or \qquad HA(p, d) = \sum_{q\in H(p,d)} Ham(q, d) \tag{4}$$

$$AggCost(p, d) = \sum_{r\in H(p,d)} VA(r, d) \qquad or \qquad AggCost(p, d) = \sum_{r\in V(p,d)} HA(r, d) \tag{5}$$

The main ideas are twofold. First, through aggregating matching cost along support arm, independent cross skeletons are also merged to form a complete support region for each pixel (see Fig. 3(b)). This arbitrary region which is the union of vertical arms of horizontal neighbors or horizontal arms of vertical neighbors, is able to provide sufficient area for aggregation and take account of local image structures as well. As a result, performance is improved on regions near depth discontinuities.

Second, this two-pass approach involves two separate 1-D aggregation instead of one 2-D aggregation on arbitrarily shaped regions. In the conventional 2-D aggregation approach, the computation complexity is $O(L\_MAX^2 \times D\_MAX \times IMG\_size)$, where L_MAX is the maximum support arm length, D_MAX is the maximum disparity range, and IMG_size is the pixel number of the image. While, in the two-pass approach, the complexity is $O(L\_MAX \times D\_MAX \times IMG\_size)$ for both *vertical* and *horizontal aggregation*. So, the computation complexity of the two-pass approach increases linearly instead of squarely with the length of the support arm. However, due to increased bandwidth and large amount of buffer cost by the aggregation step, further optimization methods are required to accelerate this process and reduce the hardware overhead (see Section 4).

Since the support region varies from pixel to pixel, we should normalize the aggregated matching costs through dividing them by the area of corresponding support region. As a result, this cost aggregation step should be capable of counting the number of pixels on support region as well.

3.5. Disparity Selection

Once matching costs and pixels counts are available, the disparity with the minimum normalized matching cost is attainable through a tree-structure *Winner-Takes-All (WTA)* module. The implementation details will be discussed in Section 5. After that, post-process including outlier detection and sub-pixel interpolation is performed to refine the final disparity map.

The proposed MCADSR algorithm lays the foundation for a high accuracy local stereo system by incorporating Mini-Census transform and adaptive support region, which improves the performance on images with radiometric changes and regions near depth discontinuities. However, this algorithm brings in computation complexity as well due to the adaptive support region. It is hard to reuse the intermediate results and the computation amount will increase especially for high-resolution videos. Considering the limited logic and memory resources, to achieve a high-speed processing on customizable platforms need lots of efforts. To this end, we propose three optimization methods for efficient hardware implementation.

4. PROPOSED OPTIMIZATION METHODS

Using MCADSR algorithm, the accuracy can be guaranteed by the adaptive support region and Mini-Census transform. However, it also brings in large computation complexity not only due to the large number of operations but the irregular pattern of operations. When assuming L_MAX for maximum support arm length and D_MAX for maximum disparity range, up to $(L\_MAX \times 2+1)^2 \times D\_MAX$ times of Hamming distance measurement and summation are needed to compute matching cost of one pixel. Considering the ever increasing image size and disparity range, these operations can be very time-consuming. On the other hand, the adaptive support region makes it even harder to reduce the complexity by utilizing data reuse, because the overlapping support regions of each pixel become arbitrary. All these factors will cause difficulty in meeting high-speed requirement for the entire stereo vision system.

We propose several optimization methods to reduce the number of operations and increase data reuse. These methods are general and hardware-friendly which can benefits many computing platforms under resource constrains. First, we propose vertical-

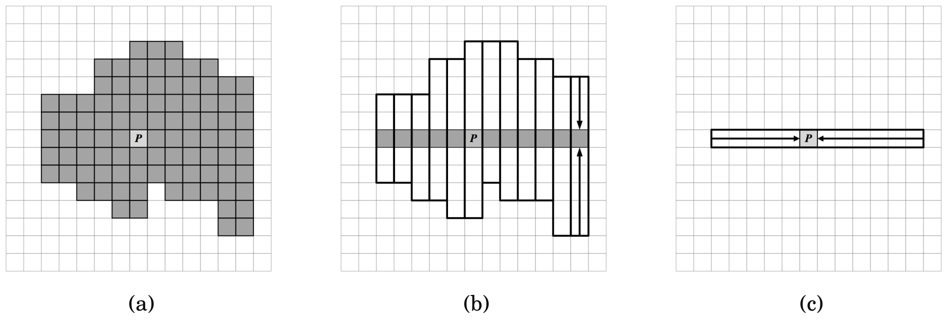


Fig. 4. The proposed aggregation sequence. (a) The support region of the pixel of interest P, which is a combination of vertical arms of horizontal neighbors. A matching cost with pixel P is measured by aggregating on the entire support region. The whole cost aggregation step involves a (b) vertical aggregation followed by (c) a horizontal aggregation on vertically aggregated costs.

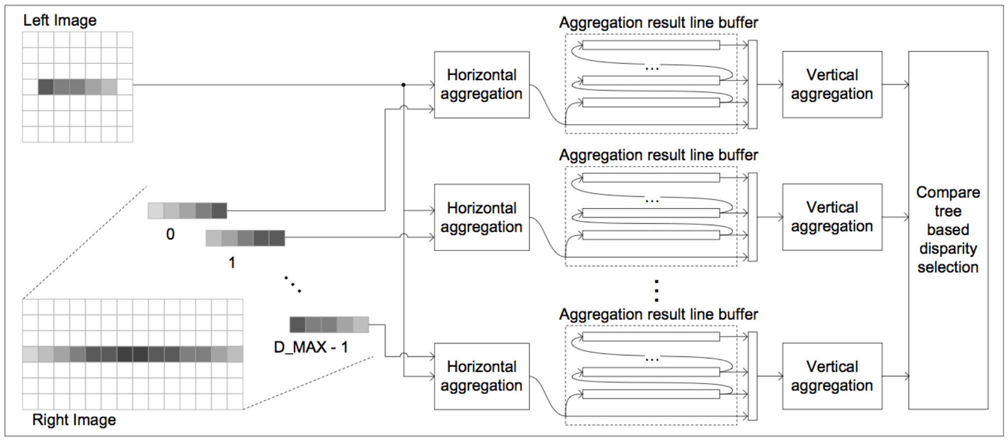
first aggregation to save memory resources. Second, we propose hybrid parallel processing to utilize data reuse. Third, we propose two hardware-friendly integral image methods to reduce the operation amount of vertical and horizontal cost aggregation respectively. Using these optimization methods, accurate and high-speed stereo vision system become a reality on resource-limited platforms. A detailed introduction of each method is given below.

4.1. Vertical-first Aggregation

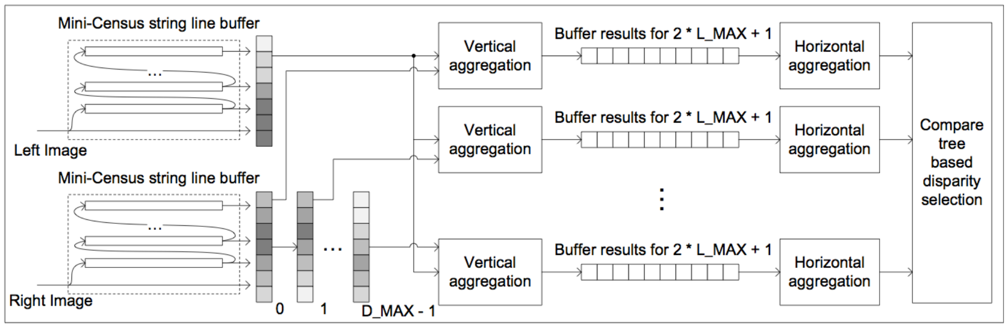
In the stereo matching step, for each pixel of interest in the left image, right image should provide as many as the number of disparities pixels to pair with it. To accelerate this matching process, the cost aggregation of each disparity level is performed concurrently using same hardware modules. Within each module, the matching cost over the support region is initiated and then aggregated in a two-pass manner.

There are basically two kinds of orthogonal two-pass aggregation method according to the 1-D aggregation sequence in adaptive support region scenario. The horizontal-first one aggregates horizontally to give a horizontal aggregated cost of each row firstly, and then the horizontal aggregated costs are aggregated vertically together to give the final matching cost. On the contrary, the vertical-first one firstly computes the vertical aggregated cost as shown in Fig. 4. These two methods have nearly the same accuracy and previous work usually adopts the former one without detailed analysis [Zhang et al. 2011] [Jin and Maruyama 2012a]. However, we will see from below that the horizontal-first scheme will cost unaffordable memory resource. In this work, we propose vertical-first aggregation in adaptive support region algorithm to largely reduce the memory consumption.

As shown in Fig. 5 (a), using horizontal-first method, the differences of the corresponding horizontal arms are measured by the Hamming distance and then aggregated into horizontal aggregated cost. The bitwidth of the Hamming distance in the horizontal aggregation should be $\lceil \log_2 (census\_width) \rceil$, where $census\_width$ is the bitwidth of the Mini-Census bit string. And the bitwidth of horizontal aggregated cost should be $(\lceil \log_2 (census\_width) \rceil + \lceil \log_2 (L\_MAX \times 2 + 1) \rceil)$ after an aggregation of $L\_MAX \times 2 + 1$ Hamming distance. Several lines of these results which could reach up to $(L\_MAX \times 2 + 1) \times IMG\_width$ of elements must be buffered in order to calculate the final matching cost vertically. Meanwhile, since the same aggregation process should be completed for each and every disparity, the number of this group of line buffers is proportional to the *disparity-level parallelism*. So, the total size of the line buffers be-



(a)


Mini-Census string line buffer
Left Image
Vertical aggregation
Buffer results for 2 * L_MAX + 1
Horizontal aggregation
Right Image
0
1
D_MAX - 1
Compare tree based disparity selection

(b)

Fig. 5. The overview of hardware implementation of two different stereo matching modules using (a) Horizontal-first aggregation and (b) Vertical-first aggregation.

tween the horizontal and vertical aggregation will reach up to $(\lceil\log_2{(census\_width)}\rceil + \lceil\log_2{(L\_MAX \times 2+1)}\rceil) \times IMG\_width \times (L\_MAX \times 2{+}1) \times disparity$. Supposing a stereo matching system dealing with a high definition of 1920×1080, disparity range of 256, L_MAX of 15 and 6-bits Mini-Census transform, the memory consumption should be $121.90Mbits$. This number is unaffordable for resource-limited platforms. For example, the latest Altera Stratix V FPGAs have only $40.73Mbits$ on-chip memory. If the data are stored in off-chip memory, the bandwidth will be the bottleneck due to the large amount of concurrent access. At the same time, the resource consumption will keep increasing with higher and higher requirement of applications.

We propose vertical-first aggregation method for our MCADSR algorithm as shown in Fig. 5 (b). At the beginning of the aggregation step, the left and right images are buffered for several lines. For the right image, which is also called the candidate image, several columns of shift registers are initialized at the output ports of the line buffer and these are vertical arms used for the vertical aggregation for different disparity levels. After the vertical aggregation, the vertical-first method only needs to

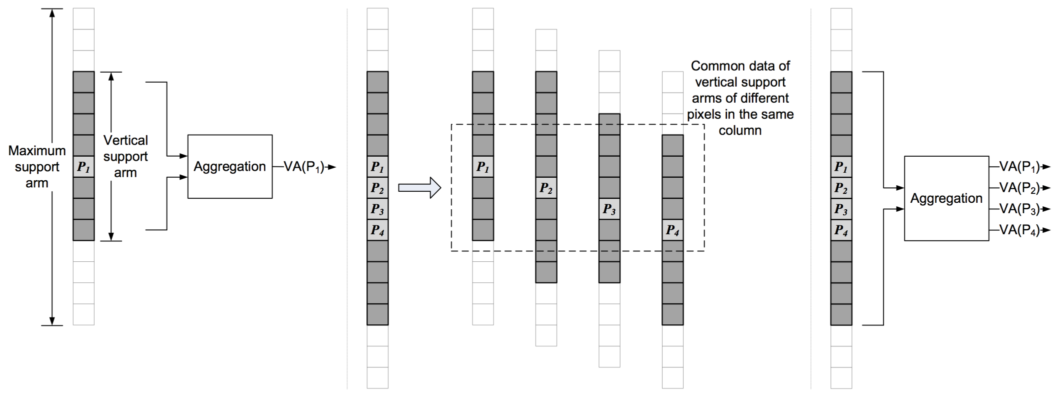


Fig. 6. Row-level parallelism: data from four image rows can be processed concurrently by adding 3 additional pixels to the vertical integral range.

buffer $L\_MAX \times 2 + 1$ of elements for the horizontal aggregation. Comparing with the horizontal-first method, it is IMG_width times reduction of memory consumption.

The key idea of this method is as follows. In these full pipelined architectures, the first aggregation step, no matter vertical or horizontal, takes input from the left and right images directly, while the second aggregation step takes intermediate results from each disparity level. As a result, the buffering of the first step can be done directly to the image pair in a centralized way, while the buffering of the second step has to be distributed into the processing of each disparity level. Meanwhile, the vertical aggregation requires the storage of several lines of data in order to process vertically. On the contrary, the horizontal aggregation only requires the storage of a range of data from the same horizontal line. Therefore, putting the *storage-hungry* vertical aggregation first can significantly reduce the memory consumption.

4.2. Hybrid Parallel Processing

In order to achieve high-speed processing, we propose a *row-level* parallel method and combine it with conventional *disparity-level parallelism*. This hybrid parallel processing method could efficiently utilize data reuse and reduce computation resource consumption.

In stereo matching, the matching cost calculations for different disparities have no data dependency and can be executed in parallel. This is called the *disparity-level parallelism*. Nearly all the previous work utilized this kind of parallelism and tried to calculate the final matching cost of one pixel with all the disparities at the same time. However, this kind of parallelism may need up to disparities copies of computation units and each unit may cost large amount of computation resources when using normal aggregation algorithm. This is unaffordable for resource-limited platforms, such as FPGAs. One way is to reuse the computation units in different time periods but this will increase the computation time to several cycles for one pixel.

Another way is to reuse the intermediate data results to reduce the complexity of the computation unit. In the vertical aggregation, the original idea is to utilize one vertical support arm for one pixel's computation. Obviously, there are common data between the computation of several pixels in neighbor rows within the same column. Here, we propose a *row-level parallelism* that could calculate the vertical aggregated costs of pixels in different rows together. As shown in Fig. 6, we have added 3 more pixels of the neighbor rows to the vertical maximum aggregation arm, and then these 4 pixels' vertical aggregated costs, $VA(P_1)$, $VA(P_2)$, $VA(P_3)$, and $VA(P_4)$, can be calculated at

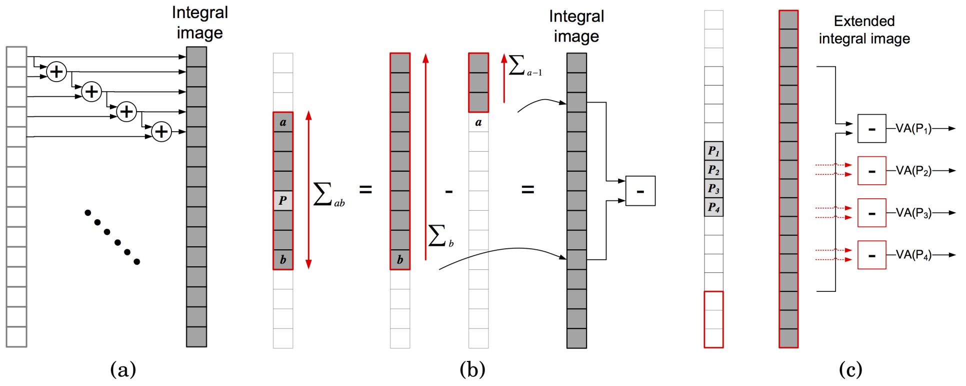


Fig. 7. (a) The summation along the support arm is pre-computed. (b) Then two values are selected according to support arm length to produce the partial sum. (c) Row-level parallelism is supported by summing more pixels and picking multiple pairs of values to yield results for each image row.

the same time with lots of common data. Compared with one pixel's computation, there are minor increases of computation resources in vertical aggregation module if proper method is adopted which will be detailed described in the next subsection. Utilizing this kind of *row-level parallelism*, the number of *disparity-level parallelism* can be reduced accordingly. So, the system can be more area efficient while achieving the same performance. In the horizontal aggregation, the scan sequence is the same as the aggregation sequence. It is hard to achieve parallelism along this sequence. However, there are also data reuse opportunities and this will be utilized to reduce computation amount which will also be described in the next subsection.

4.3. Hardware-friendly Integral Image

As stated previously, the size of support region of different pixels is entirely arbitrary. Fortunately, the 2-D aggregation is replaced by two orthogonal 1-D aggregations using a cross-based variable support region. This makes the support region simple: support arm. Taking advantage of this special structure, it is only needed to do aggregation in horizontal arms and vertical arms. To further reduce the computation complexity, Zhang et al. [2009a] proposed an efficient way using integral images [Veksler 2003]. As is shown in Fig. 7 (a), this method pre-computes the sum of all initial costs to the above. After the integral image is computed, the sum of initial costs over any support arm length can be computed via a minus operation as shown in Fig. 7 (b). The main ideas are twofold. First, by pre-computing an integral image before producing the aggregated results, this method reduces the number of re-referencing times and relieves the inner consumption of bandwidth. Second, it provides constant processing time guarantees on both vertical and horizontal line with varying length. Based on the above mechanisms, we are able to pipeline the entire cost aggregation step on arbitrarily shaped support regions.

In the matching cost computation, we propose two hardware-friendly integral image methods for the vertical and horizontal aggregation respectively. For the horizontal one, a shifter architecture with limited length is used to save storage. For the vertical one, an extended integral image is used for aggregation which could produce several results belong to different rows at the same time with minor increases of resources consumption.

For the horizontal aggregation, the original method has to build the integral image firstly and this will also involve large amount of memory storage. Considering that

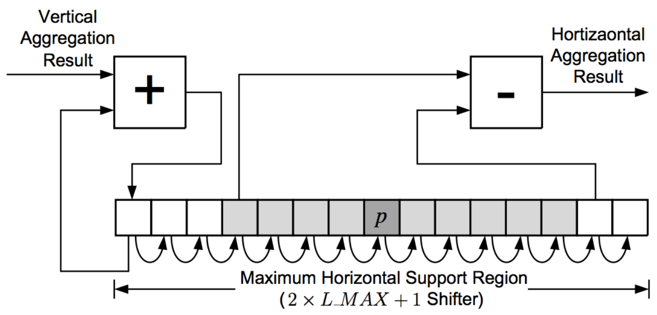

Vertical
Aggregation
Result
+
-
Hortizaontal
Aggregation
Result
p
Maximum Horizontal Support Region
($2 \times L\_MAX + 1$ Shifter)

Fig. 8. The vertical aggregation result is summed and stored in a shifter which is exactly $L\_MAX \times 2 + 1$ long. Those results are selected according to the support arm length and then shifted each cycle, with the oldest value swapped out.

the horizontal aggregation only needs the data of the support arm. The earlier aggregated data will soon become useless and should be thrown. So we propose a shifter based horizontal aggregator as shown in Fig. 8. The aggregated vertical cost will be added with the last data of the shifter and then be stored into the last position of the shifter. The other data in the shifter will move forward one position. The subtraction of the left arm end and the right arm end is the horizontal aggregated cost which is the final matching cost $AggCost(x, y, d)$. For simple consideration, the structures for $PixCount(x, y, d)$ aggregation are not depicted in the Figure, which are essentially the same.

Noted that this conventional integral image technique also requires a considerable amount of memory resource to cache $L\_MAX \times 2$ lines of integral results so as to serve the vertical aggregation of the following row. Considering the *disparity-level parallelism*, the original method needs to store disparities copies of line buffers, which store the vertical integral image. This is unaffordable and we have to compute the integral image for every support region for every pixel without data reuse. While, if the *row-level parallelism* is considered, the extended vertical integral image can serve several pixels at the same time. As shown in Fig. 7 (c), the support regions of the 4 pixels in the neighboring rows and the same column has a lot of common data. So three more data are added to the support region which is called extended support region in our paper. And then an extended integral image is built based on this region. At last, the 4 vertical aggregated costs can be achieved at the same time. Using this method, the 4 copies of up to $L\_MAX \times 2$ addition can be reduced to 1 copy of $L\_MAX \times 2 + 3$ addition.

We presented how the three optimization methods improve the hardware implementation of MCADSR algorithm in terms of memory consumption and high-speed processing. Moreover, these methods as general optimization methods also benefit the domain of stereo vision. Vertical-first aggregation basically can be incorporated in algorithms requiring a two-pass aggregation, especially over an arbitrarily shaped region. Hybrid parallelism can be obtained based on the disparity level parallelism, leveraging data reuse opportunities in neighboring image lines and providing a knob that allows a tradeoff between the disparity level and row level parallelism. Hardware friendly integral image can be broadly applied to the aggregation process in this domain. And this method provides constant processing time guarantees on the summation over any given regions.

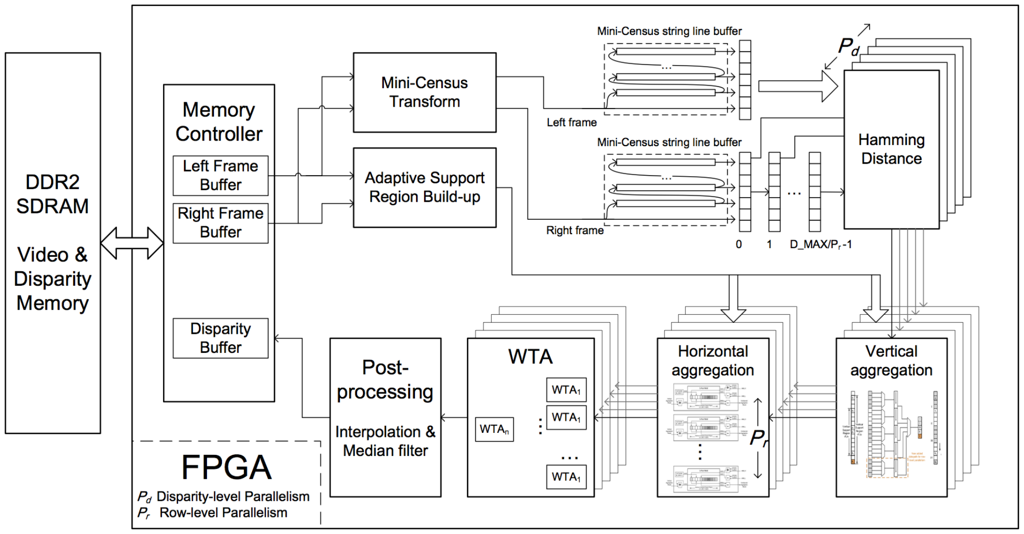


Fig. 9. Overview of the hardware implementation of our proposed algorithm.

5. HARDWARE IMPLEMENTATION

To achieve high-speed MCADSR stereo matching, we propose a full pipelined design on FPGA which can operate at pixel clock. The optimization methods, proposed in Section 4 for customizable platforms, are carefully mapped onto FPGA under the memory resource constraints. First, a customized datapath involving both full pipelined and hybrid parallel processing is proposed. The high-speed processing is guaranteed by the pipeline and parallelism, and the memory consumption is reduced by the hybrid parallel architecture. This architecture is designed in a parameterized method based on which the trade-offs can be easily analyzed. Second, hierarchical adder tree and RAM based shifter are used for vertical and horizontal integral images in cost aggregation. The intermediate results are highly reused to save the computation resource. Besides, some general techniques are used in other modules, such as line buffer storage in Mini-Census transformation and adaptive support region build-up.

Fig. 9 shows our proposed hardware design. The video frames and disparity results are stored in the off-chip DDR2 SDRAM. The processing blocks are all implemented using FPGA in a full pipelined processing method. The architecture on FPGA consists of three high-level modules: pre-processing, stereo matching, and post-processing. The pre-processing module changes the input gray image into the Mini-Census representation and builds up the support region for each pixel. The stereo matching is the main processing module which contains three parts: a vertical cost aggregator, a horizontal aggregator and a WTA calculator. The post-processing module will exclude the error matching and refine the results. The detail will be explained in the following subsections.

5.1. Hybrid Parallel Processing

We have proposed a hybrid parallel method that utilizes both *disparity-level parallelism* and *row-level parallelism* in Section 4.2. With *row-level parallelism* incorporated, the *disparity-level parallelism* can be lessened, leading to reduced resource consumption without changing the processing.

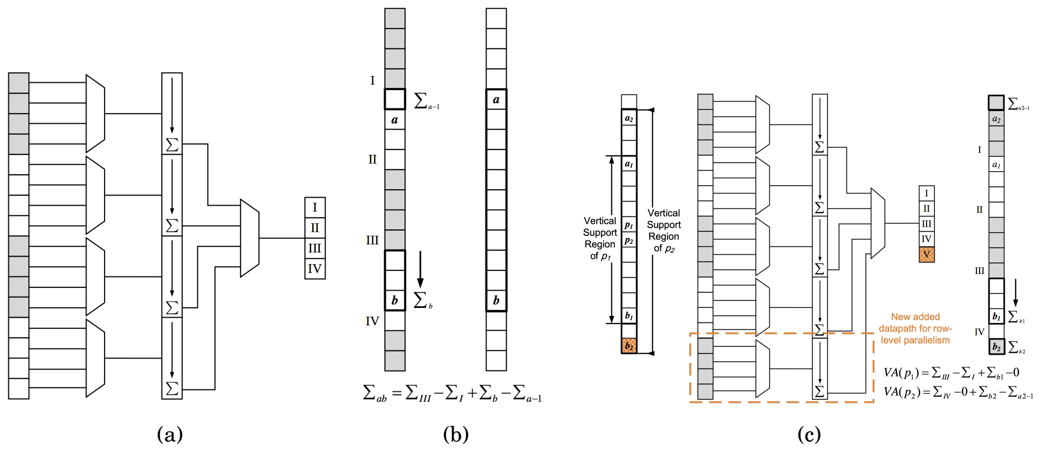

I
II
III
IV
(a)
I
Σ_{a-1}
a
II
III
Σ_b
b
IV
a
b
$\Sigma_{ab} = \Sigma_{III} - \Sigma_I + \Sigma_b - \Sigma_{a-1}$
(b)
a_2
a_1
p_1
p_2
b_1
b_2
Vertical Support Region of p_1
Vertical Support Region of p_2
I
II
III
IV
V
New added datapath for row-level parallelism
Σ_{a2-1}
Σ_{b1}
Σ_{b2}
$VA(p_1) = \Sigma_{III} - \Sigma_I + \Sigma_{b1} - 0$
$VA(p_2) = \Sigma_{IV} - 0 + \Sigma_{b2} - \Sigma_{a2-1}$
(c)

Fig. 10. (a) The proposed hierarchical adder tree for the vertical aggregation; (b) A computation example using hierarchical adder tree: two values from the last level and two values from the first level are selected based on the support arm length to produce the exact result; (c) Row-level parallelism support with additional adding structure and selecting logic.

Fig. 9 shows the utilization of P_d *disparity-level* and P_r *row-level parallelism* for a stereo matching with D_MAX disparities in total, where $D\_MAX = P_d \times P_r$. The system works like this: first, it concurrently scans pixels from P_r lines for the first P_d disparities (i.e. $0 \sim P_d - 1$). And then the system rescans these P_r lines for the second P_d disparities (i.e. $P_d \sim P_d \times 2 - 1$). This will last until all the disparities are scanned, after which the process of P_r new lines will begin.

As a result, available Mini-Census strings and support arm lengths should be buffered in order to serve the re-reference from cost aggregation module, rather than be outputted directly. To ensure that the aggregation can be performed continuously without any pause, this specific buffer should also provide additional lines to hold incoming data and hence, avoid pollution to the data in use. As cost aggregation module vertically takes $L\_MAX \times 2 + P_r$ data at a time and scans P_r times per image line, the height of the buffer should be set to $L\_MAX \times 2 + P_r + P_r$. Once P_r scans on current line are completed, the buffer should "jump" P_r lines to continue.

With the buffer described above, the streaming Mini-Census strings and support arm lengths are buffered and outputted vertically to perform the proposed vertical-first aggregation under hybrid parallelism. Modifications to other submodules will be discussed in the following subsections.

5.2. Vertical and Horizontal Integral Image for Cost Aggregation

Integral image is usually used in cost aggregation due to its natural feature of data reuse. Due to the resource constraints, two optimizations are proposed in Section 3 for the vertical and horizontal aggregation respectively. When being mapped onto FPGAs, there are still some challenges, such as long latency for the vertical aggregation and too many registers consumption for the horizontal aggregation. We propose hierarchical adder tree and RAM based shifter with reduced bit-width to overcome the challenges.

Using traditional vertical integral image, the latency will be $L\_MAX \times 2$ addition and this is too long. Fig. 10 illustrates the proposed vertical integral image for cost aggregation. The input Mini-Census strings are first buffered to be aggregated vertically (line buffers are not included in the Figure). In order to save aggregation cycles, we

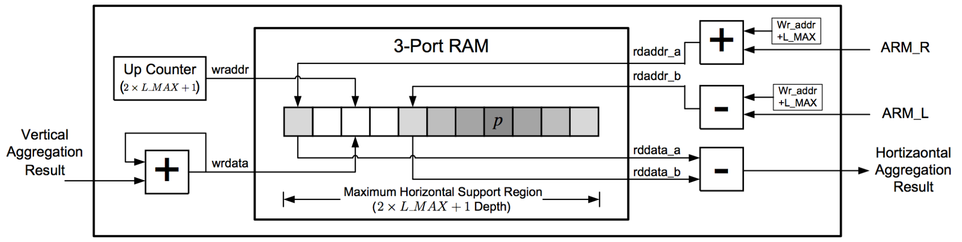


Fig. 11. The detailed hardware architecture of horizontal aggregation. The shifter is implemented using 3-port RAM and offset computing logic.

implement a hierarchical adder tree as in Fig. 10 (a), which computes partial sums to produce the final result based on the idea of integral image. The vertical support arm is divided into several groups firstly. In the first cycle, the integral image is built for each group. In the second cycle, the last data of each group, which is also the summation of each group, is fetched to build the second layer of integral image. In the third cycle, the vertical aggregation can be drawn by two data from the first stage's integral image and two data from the second stage as shown in Fig. 10 (b). With this structure, the vertical aggregated results can be produced within three cycles. And this is suitable for *row-level parallelism* as shown in Fig. 10 (c). Note that the partition granularity has close relation with the maximum processing frequency due to the critical path. If the granularity is too fine in the first level, the critical path in the second level will be very long due to large number of groups. If the granularity is too coarse in the first level, then the critical path in the first level will be very long due to large amount of data in each group. A promising partition is to make the group number similar to the number of data in each group. We designed a parameterized architecture and the performance could be tuned during the experiment.

As stated before, an integral image along the row direction is built for our horizontal aggregation. The proposed shifter architecture is easily built by registers. However, this will cost too many registers which are very limited resources. We propose an architecture in Fig. 11 and the main component is a 3-port RAM based shifter with fixed length of the maximum aggregation region. There is no need to store the whole row because the old integral result will soon become useless. The RAM is organized as a ring structure which is controlled by the write and read addresses. The front end of horizontal aggregator is an accumulator. The vertical aggregated results are accumulated and then stored in RAMs, which will possibly be fetched to produce the final $AggCost(x, y, d)$ in the near future. Because only the difference of the two data, the left and right end of the support arm, in the shifter is needed, not the absolute value of each data, the bit-width could be reduced to which could express the summation of the support arm. In our design, bit-width of 13 is used when L_MAX is 15. For simple consideration, the structures for $PixCount(x, y, d)$ aggregation are not depicted in the Figure, which are essentially the same.

5.3. Mini-Census Transformation

As stated before, the sparse Census transformation which intends to reduce hardware resource requirements is able to deliver better results than full Census transformation [Fife and Archibald 2013]. We implement a 5×5 Mini-Census filter to produce the 6-bit string. The transformation pattern is shown in Fig. 2. In order to make the window slide over the image in scanline order, we adopt line buffer and register matrix to preserve the local neighborhood. Certain pixels are compared with the pixel of in-

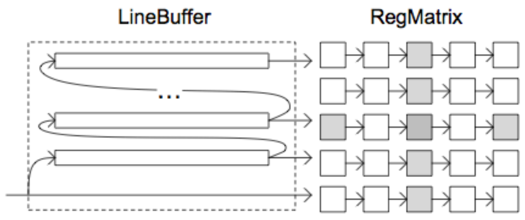


Fig. 12. The proposed Mini-Census transformation architecture. Pixels first arrive through a line buffer to be output vertically, and then transformed on a register matrix of 5×5.

terest as pixels come in continuously. With the structure shown in Fig. 12, the output Mini-Census string rate is synchronized with the input pixel rate.

5.4. Adaptive Support Region Build-up

For storage consideration, we set L_MAX to 15. Inside the algorithm proposed in Section 3, we perform a luminance similarity test for a consecutive set of pixels to search for the largest span in each direction. We present two independent thresholds providing a finer knob for support region boding: a loose (aggressive) threshold in the near neighborhood to tolerate moderate changes and a strict (conservative) threshold in the remote to prevent including unwanted pixels. We empirically set $\tau_1 = 35$ for $dist(p_1, p_2) \leq 8$ and $\tau_2 = 6$ for $8 < dist(p_1, p_2) \leq L\_MAX$ in our design.

The architecture for support region builder is very similar to the Mini-Census transformer. However, the number of line buffers and the side length of register matrix are increased to $2 \times L\_MAX + 1$ to cover the maximum possible region. The input data are first buffered to be examined vertically. Once the data are ready for luminance similarity check, pixels reside on four directions to the pixel of interest are examined to produce four bit strings using Equation 2. Then the quadruple $\{h_p^-, h_p^+, v_p^-, v_p^+\}$ is produced based on the indicator.

5.5. Disparity Selection & Refinement

After matching cost aggregation the $AggCost$ and $PixCount$ values for all disparity levels in the ranges $[0 : D\_MAX - 1]$ are compared with each other in order to compute the minimum normalized value and its disparity. The comparison is carried out by a a tree-structure *Winner-Takes-All (WTA)* module consisting of a collection of compare units and registers. Each compare unit receives two pairs of $AggCost$ and $PixCount$ and their corresponding disparities as input, compares the two normalized values and outputs the minimum pair along with its disparity. In performing the normalization step, we use a multiply-subtract technique to track the minimum normalized matching cost in order to avoid the division computation, which is indicated by (6).

$$\begin{gathered}\frac{AggCost\,(d_0)}{PixCount\,(d_0)} < \frac{AggCost\,(d_1)}{PixCount\,(d_1)} \\ \Longleftrightarrow \\ AggCost\,(d_0) \times PixCount\,(d_1) < AggCost\,(d_1) \times PixCount\,(d_0)\end{gathered} \tag{6}$$

To explore hybrid parallelism that concurrently process more image rows and less disparity levels, the whole tree-structured WTA module is further divided into several small trees with line buffers, where results from only a small disparity range are pro-

Table I. Hardware resource consumption report

	Total	Mini-Census	ADSR Builder	Stereo Matcher	WTA
# of Unites in the System	NA	2	2	32	4
Combinational ALUTs	60160	84	1663	1395	2077
Registers	33291	84	550	894	929
Memory Bits	2869138	29460	178719	5704	21030
DSP Blocks	512	0	0	0	128

cessed at a stage to find the local minimum value. As a result, local results must be buffered to compare with the results of the next stage. Such process is repeated until all local minimum results have been compared. Then the global minimum value and its disparity within the entire disparity range is ready for output.

The disparity refinement modules including subpixel interpolation and outlier detection use similar structures like line buffer as mentioned above. Hence, we are not going to illustrate them in detail.

6. EXPERIMENTAL EVALUATION

6.1. Experiment Setup

The proposed MCADSR architecture is synthesized using Altera Quartus-II 11.0 targeting the Altera Stratix IV EP4SGX230 FPGA device. Unless stated otherwise, we assume an image size of 1024×768, disparity range of $0 \sim 127$, maximum support arm length (L_MAX) of 15, and row-level parallelism of 4 which means the evaluated system simultaneously processes data from 4 image rows. The maximum operation frequency is $102.86MHz$.

Table I lists the detailed hardware resource consumption of each submodule. As we can see from the Table, the majority of resource is consumed by stereo matching module, which will scale linearly with the disparity range. The memory consumption is relatively lower than previous work due to the use of our proposed optimization methods. The advantages of different optimization methods will be illustrated in the following subsections.

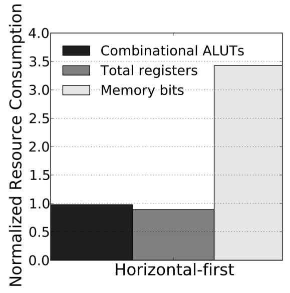
6.2. Effect of Vertical-first Aggregation

The effect of our proposed aggregation scheme is evaluated by comparing two stereo vision systems using two different aggregation sequences: vertical-first and horizontal-first. We use the same configuration as mentioned above and a row-level parallelism of 4 on both systems. The horizontal-first one causes significantly higher amount of resource as $9.83Mbits$ on-chip memory, $58.87k$ combinational ALUTs and $29.7k$ registers, compared to $2.87Mbits$ memory, $60.16k$ combinational ALUTs and $33.29k$ registers of the vertical-first one. Fig. 13(a) depicts the resource consumption of horizontal-first system normalized to the vertical-first one. Though the logic consumption is lightly increased under vertical-first scheme, the proposed aggregation sequence is able to save 70.8% on-chip memory than the horizontal-first scheme.

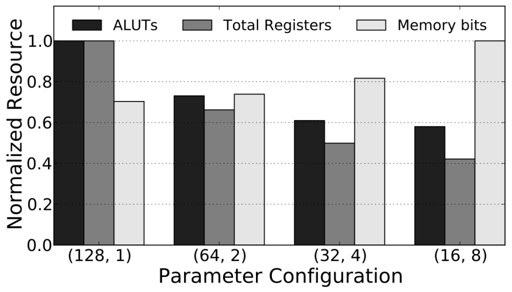
The vertical-first scheme uses one central line buffer before cost aggregation step is performed rather than distributed line buffers in each cost aggregation, which significantly reduce the consumption of on-chip memory. Due to the benefit of this optimization technique, we are able to achieve fast processing on much higher resolution images.

6.3. Effect of Hybrid Parallelism

Fig. 13(b) shows normalized hardware resource consumption of MCADSR as disparity-level parallelism and row-level parallelism of the baseline system are varied while keeping the same $P_d \times P_r$. The ordered pair on x axis denotes (P_d, P_r), which means

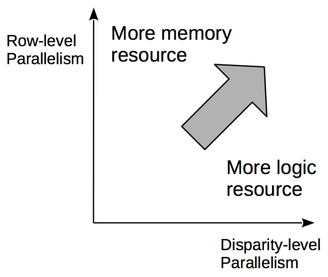


(a)

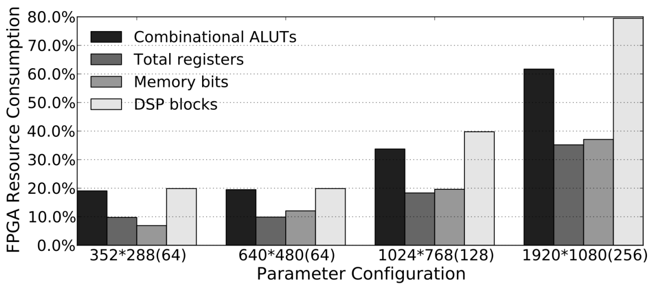


(b)

Fig. 13. (a) The resource consumption of horizontal-first aggregation scheme with respect to vertical-first. (b) Normalized resource consumption under different configurations of parallelism.



(a)



(b)

Fig. 14. (a) The row-level parallelism provides an alternative in scaling parallelism. (b) The proposed MCADSR architecture scalability.

the number of disparities and the number of image rows concurrently being processed, respectively. As is shown in the figure, when we trade off disparity-level parallelism in favor of row-level parallelism, the amount of logic resource as Combinational ALUTs and Registers drops while the consumption of memory rises. Comparing with the implementation with full disparity-level parallelism, the first in the figure, the implementation with row-level parallelism of 4 can save nearly 40.10% ALUTs with the penalty of only 15.26% memory increase.

To illustrate this, Fig. 14(a) depicts the design space for total parallelism, which can be formulated as $P_{total} = P_d \times P_r$. Two conclusions are in order: 1) high row-level parallelism relieves the computation of a large disparity range and meanwhile requires more image rows to be buffered, 2) under the same level of total parallelism, the row-level parallelism provides a *computation / storage trade-off knob* such that we can select a value that favors our desired metric.

6.4. Design Scalability

The design scalability is illustrated in Fig. 14(b). The detailed configurations are given as $ImageSize(DisparityRange)$ along the horizontal axis, and the numbers of row-level parallelism are all 4. The maximum frequencies are $113.64MHz$, $109.67MHz$, $102.86MHz$ and $98.80MHz$ for different configurations shown from left to right in the figure. The maximum frequencies decrease as the $ImageSize$ and $DisparityRange$ in-

Table II. Processing speed comparison for stereo vision system

	Platform	Max Disp.	Image Size	FPS	MDE/s	Algorithm
Proposed	FPGA	256	1920×1080	47.6	25242	Mini-Census adaptive support region
		128	1024×768	129	13076	
		64	640×480	357	7028	
		64	352×288	1121	7279	
Shan et al. [2012]	FPGA	256	1280×1024	46	15437	SAD-based block matching
Ambrosch et al. [2009]	FPGA	100	450×375	599	10125	SAD-based block matching
Jin and Maruyama [2012a]	FPGA	60	1920×1080	76	9456	SAD+Mini-Census-based FLC
Jin et al. [2010]	FPGA	64	640×480	64	4522	Census transformation
Zhang et al. [2011]	FPGA	64	1024×768	60	3019	Mini-Census adaptive support region
Ambrosch and Kubinger [2010]	FPGA	60	750×400	60	1080	Modified Census transform
Ttofis and Theocharides [2012]	FPGA	64	640×480	30	589	Segmentation-based ADSW (SAD)
MCADSW [2010]	ASIC	64	352×288	42	272.5	Mini-Census ADSW
Zhang et al. [2009b]	GPU	64	450×375	12	129.6	SAD-based adaptive support region
Chang et al. [2007]	DSP	16	384×288	50	88.5	SAD-based block matching
RealtimeGPU [2006]	GPU	32	320×240	43	53	Dynamic programming
RealtimeBP [2006]	GPU	16	384×288	16	22.2	Hierarchical belief propagation
FastAggreg [2008]	CPU	16	320×240	5	18.9	SAD-based efficient aggregation
VariableCross [2009a]	CPU	60	450×375	0.63	13	SAD-based adaptive support region

crease. This is because more resource consumption brings more pressure on place and route, which is one of the key factors for maximum processing frequency in FPGAs. The scalability figure shows that the numbers of combinational ALUTs and registers scale almost linearly with the disparity range. The reason is that the cost aggregation and WTA module should be duplicated as many as the number of disparities. On the other hand, the amount of memory consumption increases as the image size becomes larger because the involving line buffers should scale with IMG_W. The consumption of DSP blocks corresponds to the multiply computation in the WTA module. As we fix P_{row} to 4 in all schemes in the figure, it scales linearly with the disparity range.

Our proposed optimization techniques significantly reduce the memory resource consumption and enables implementing a Full HD stereo vision system with a large disparity range of 256 using only 5.423Mb on-chip memory. Prior work such as [Jin and Maruyama 2012a] explores the possibility of a Full HD implementation on FPGA. However, their memory resource consumption scales fast with disparity range, more than 14.220Mb out of 14.976Mb on their platform for a disparity range of 60. Thus it cannot afford a disparity range larger than 60, which is usually not a suitable configuration for high resolution images.

6.5. Performance Comparison

Table II shows the speed comparisons between our proposed MCADSR and state-of-the-art real-time implementations for stereo matching. The processing speeds of different systems are presented in *frame rates (FPS)* and *million disparity estimation per second (MDE/s)*, which is equal to $IMG\_W \times IMG\_H \times DisparityRange \times FPS$.

Our proposed MCADSR implementation is able to deliver 47.6 FPS at 1920×1080 with a disparity range of 256 and the processing speed could reach as high as $25242 MDE/s$. For image sizes of 1024×768 with a disparity range of 128, 640×480 and 352×288 with a small disparity range of 64, the processing speeds are $13076 MDE/s$, $7028 MDE/s$ and $7279 MDE/s$, respectively. Due to different maximum frequencies, the implementation with image size of 352×288 slightly outperforms that with image size of 640×480.

Shan et al. [2012] and Ambrosch et al. [2009] achieve similar performance compared to ours due to the simplicity of SAD-based fixed-support algorithms. Jin and Maruyama [2012a] achieves a more than 2.5 times slower processing speed than ours.

Table III. Processing accuracy evaluation for stereo vision system

Image Set	Tsukuba			Venus			Teddy			Cones			
Algorithms	nocc.	all	disc.	nocc.	all	disc.	nocc.	all	disc.	nocc.	all	disc.	AVG
ADCensus [2011]	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
AdaptingBP [2006]	1.11	1.37	5.79	0.10	0.21	1.44	4.22	7.06	11.8	2.48	7.92	7.32	4.23
Jin and Maruyama [2012a]	1.38	1.84	7.36	0.30	0.48	2.09	7.41	12.7	17.5	3.44	9.19	9.90	6.13
VariableCross [2009a]	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
Proposed	**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. [2009b]	1.71	2.22	6.74	0.87	0.87	2.88	9.90	15.0	19.5	6.66	12.3	13.4	7.65
RealtimeBP [2006]	1.49	3.40	7.87	0.77	1.90	9.00	8.72	13.2	17.2	4.61	11.6	12.4	7.69
Zhang et al. [2011]	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
FastAggreg [2008]	1.16	2.11	6.06	4.03	4.75	6.43	9.04	15.2	20.2	5.37	12.6	11.9	8.24
RealtimeGPU [2006]	2.05	4.22	10.6	1.92	2.98	20.3	7.23	14.4	17.6	6.41	13.7	16.5	9.82
Ambrosch and Kubinger [2010]	5.81	7.14	22.6	2.61	3.33	25.3	9.79	15.5	25.7	5.08	11.5	15.0	12.5
Ttofis and Theocharides [2012]	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
Jin et al. [2010]	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. [2012]	9.49	11.1	36.0	7.10	8.65	41.7	16.9	25.4	40.6	10.6	20.2	28.3	21.3

1. Nocc.(non-occluded regions) are the errors only for the non-occluded regions.
2. All (all regions) are the errors in all regions excluding borders of the image.
3. Disc (discontinuity) are the errors only for the regions near depth discontinuities.
4. The AVG column by which the table is sorted shows the average percentage of bad pixels over all twelve columns.

And their memory consumption limits the improvement of processing speed. Zhang et al. [2011] adopts similar algorithm as ours but achieves only 12% processing speed compared to ours. This is because we adopt several optimization methods to relieve memory limitation to enable high-speed processing. Although GPU and DSP implementations achieve better processing speed than CPUs, they are far from real-time performance for high resolution images.

The processing accuracy of the proposed stereo vision system is evaluated using the Middlebury stereo database [Scharstein and Szeliski 2002]. We compute the quality measure as the percentage of bad matching pixels between the computed disparity map $d_C(x, y)$ and the ground truth map $d_T(x, y)$ using a threshold $\delta_d = 1.0$ as indicated in (7), where N is the total number of pixels. In addition to computing this statistic over the whole image, we also focus on two different kinds of regions to support further analysis: non-occluded regions and regions near depth discontinuities. Fig. 15 compares the disparity map of our proposed algorithm and state-of-the-art stereo implementations on Middlebury datasets to give a visual indication of how well the methods perform. Table III presents the quantitative performance of our method and those of other local and global stereo methods.

$$B = \frac{1}{N} \sum \left(|d_C\,(x, y) - d_T\,(x, y)| > \delta_d \right) \tag{7}$$

Our system achieves good results on Middlebury evaluation and ranks second in evaluated FPGA-based systems. Jin and Maruyama [2012a] can provide relatively high-speed and accurate results for these datasets using a little more complex algorithm on FPGA. However, as described previously, it cannot afford a disparity range larger than 60 due to resource limitation. Our system performs better than [Zhang et al. 2011], which basically proposes a similar algorithm with ours, but adopts fixed vertical span in building support region in order to reduce hardware overhead and realize pipeline processing. In contrast, our proposed optimizing methods enable the full support of adaptive region, leading to increased matching accuracy. Other work which delivers more accurate results are all far from real-time processing at high resolution and large disparity range images.

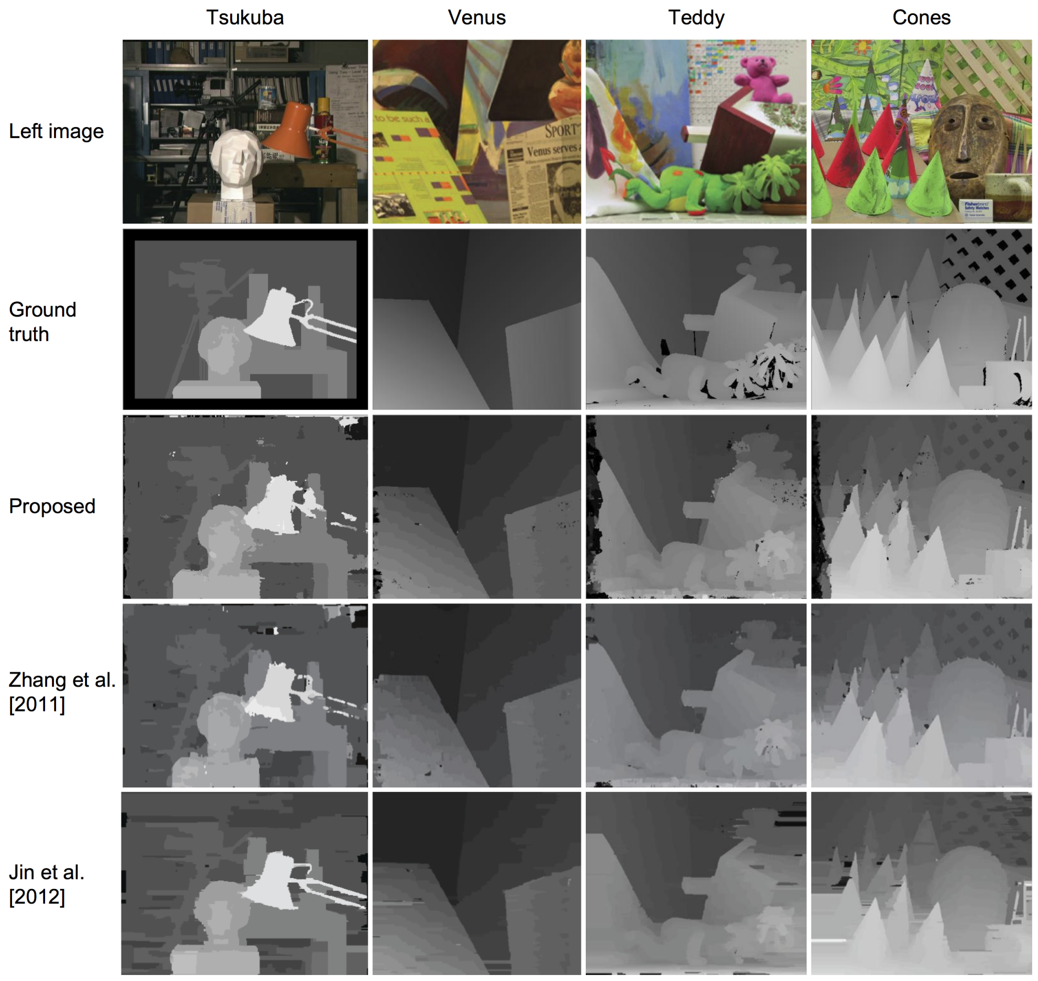


Fig. 15. Disparity map comparison of the proposed algorithm and two other implementations on Middlebury datasets of *Tsukuba, Venus, Teddy and Cones*

Compared to algorithms such as VariableCross[2009a] and [Zhang et al. 2009b], both of which employ SAD and cross-based aggregation, our results on the Tsukuba image pair are not competitive. The Tsukuba image pair contains very noisy and repetitive regions near the lamp and the bookshelf in the background, which lead to ambiguity in disparity selection when using Mini-Census, while pixel-based SAD can handle this well. Though Mini-Census generally performs better, SAD sometimes provides proper results where Mini-Census suffers. Since different matching cost such as SAD is orthogonal to our aggregation schemes, they can be combined with our optimization schemes to further improve performance.

7. CONCLUSION

Stereo vision is a fundamental domain in many computer vision applications. The extra processing dimension of depth estimation for this domain brings in a big challenge of memory limitation. To address this challenge, we use a Mini-Census Adaptive Support Region (MCADSR) stereo matching algorithm as a case study. Three optimization methods, vertical-first aggregation, hybrid parallel processing, and hardware-friendly integral image, are used to relieve the memory limitation of this domain. These methods also change the irregular operations of adaptive support algorithm into regular ones and reduce operation amount using efficient data reuse. Due to the representative feature of the algorithm, these optimization methods are general and can be easily adopted in other problems in this domain.

We also proposes an full pipelined FPGA based customizable system with the above optimization methods. The combination of Mini-Census transformation and adaptive support region makes the algorithm have an overall average error rate of 7.65%. Meanwhile, the full pipelined processing on FPGA brings in high-speed processing which can process 47.6 fps (frames per second) and 129 fps for video of 1920×1080 resolution with a large disparity range of 256 and video of 1024×768 resolution with a large disparity range of 128 respectively.

Our future work will investigate run-time reconfigurable architecture for stereo vision domain to further reduce resource consumption and achieve more accurate results. We will also extend our optimization methods to other important applications in stereo vision domain.

REFERENCES

K. Ambrosch, M. Humenberger, W. Kubinger, and A. Steininger. 2009. SAD-based stereo matching using FPGAs. *Embedded Computer Vision* (2009), 121–138.

K. Ambrosch and W. Kubinger. 2010. Accurate hardware-based stereo vision. *Computer Vision and Image Understanding* 114, 11 (2010), 1303–1316.

N. Chang, Ting-Min Lin, Tsung-Hsien Tsai, Yu-Cheng Tseng, and Tian-Sheuan Chang. 2007. Real-Time DSP Implementation on Local Stereo Matching. In *Multimedia and Expo, 2007 IEEE International Conference on*. 2090–2093. DOI: http://dx.doi.org/10.1109/ICME.2007.4285094

N.Y.-C. Chang, Tsung-Hsien Tsai, Bo-Hsiung Hsu, Yi-Chun Chen, and Tian-Sheuan Chang. 2010. Algorithm and Architecture of Disparity Estimation With Mini-Census Adaptive Support Weight. *Circuits and Systems for Video Technology, IEEE Transactions on* 20, 6 (june 2010), 792–805.

L. De-Maeztu, A. Villanueva, and R. Cabeza. 2012. Near Real-Time Stereo Matching Using Geodesic Diffusion. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 34, 2 (2012), 410–416.

W. S. Fife and J. K. Archibald. 2013. Improved Census Transforms for Resource-Optimized Stereo Vision. *Circuits and Systems for Video Technology, IEEE Transactions on* 23, 1 (jan. 2013), 60–73. DOI: http://dx.doi.org/10.1109/TCSVT.2012.2203197

H. Hirschmuller and D. Scharstein. 2009. Evaluation of stereo matching costs on images with radiometric differences. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 31, 9 (2009), 1582–1599.

Minxi Jin and T. Maruyama. 2012a. A fast and high quality stereo matching algorithm on FPGA. In *Field Programmable Logic and Applications (FPL), 2012 22nd International Conference on*. 507–510. DOI: http://dx.doi.org/10.1109/FPL.2012.6339266

Minxi Jin and Tsutomu Maruyama. 2012b. A real-time stereo vision system using a tree-structured dynamic programming on FPGA. In *Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays (FPGA '12)*. ACM, New York, NY, USA, 21–24. DOI: http://dx.doi.org/10.1145/2145694.2145698

S. Jin, J. Cho, X. Dai Pham, K.M. Lee, S.K. Park, M. Kim, and J.W. Jeon. 2010. FPGA design and implementation of a real-time stereo vision system. *Circuits and Systems for Video Technology, IEEE Transactions on* 20, 1 (2010), 15–26.

A. Klaus, M. Sormann, and K. Karner. 2006. Segment-Based Stereo Matching Using Belief Propagation and a Self-Adapting Dissimilarity Measure. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, Vol. 3. 15–18. DOI: http://dx.doi.org/10.1109/ICPR.2006.1033

Xing Mei, Xun Sun, Mingcai Zhou, Shaohui Jiao, Haitao Wang, and Xiaopeng Zhang. 2011. On building an accurate stereo matching system on graphics hardware. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*. 467–474. DOI: http://dx.doi.org/10.1109/ICCVW.2011.6130280

Daniel Scharstein and Richard Szeliski. 2002. A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. *International Journal of Computer Vision* 47 (2002), 7–42. Issue 1-3.

Yi Shan, Zilong Wang, Wenqiang Wang, Yuchen Hao, Yu Wang, Kuenhung Tsoi, Wayne Luk, and Huazhong Yang. 2012. FPGA based memory efficient high resolution stereo vision system for video tolling. In *Field-Programmable Technology (FPT), 2012 International Conference on*. 29–32. DOI:http://dx.doi.org/10.1109/FPT.2012.6412106

F. Tombari, S. Mattoccia, L. Di Stefano, and E. Addimanda. 2008. Near real-time stereo based on effective cost aggregation. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*. 1–4. DOI:http://dx.doi.org/10.1109/ICPR.2008.4761024

C. Ttofis and T. Theocharides. 2012. Towards accurate hardware stereo correspondence: A real-time FPGA implementation of a segmentation-based adaptive support weight algorithm. In *Design Automation Test in Europe Conference Exhibition (DATE), 2012*. 703–708. DOI:http://dx.doi.org/10.1109/DATE.2012.6176561

Olga Veksler. 2003. Fast variable window for stereo correspondence using integral images. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, Vol. 1. IEEE, I–556.

O. Veksler. 2005. Stereo correspondence by dynamic programming on a tree. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, Vol. 2. IEEE, 384–390.

L. Wang, M. Liao, M. Gong, R. Yang, and D. Nister. 2006. High-quality real-time stereo using adaptive cost aggregation and dynamic programming. In *3D Data Processing, Visualization, and Transmission, Third International Symposium on*. IEEE, 798–805.

Q. Yang, L. Wang, R. Yang, S. Wang, M. Liao, and D. Nister. 2006. Real-time global stereo matching using hierarchical belief propagation. In *The British Machine Vision Conference*. 989–998.

R. Zabih and J. Woodfill. 1994. Non-parametric local transforms for computing visual correspondence. *Computer VisionECCV'94* (1994), 151–158.

K. Zhang, J. Lu, and G. Lafruit. 2009a. Cross-based local stereo matching using orthogonal integral images. *Circuits and Systems for Video Technology, IEEE Transactions on* 19, 7 (2009), 1073–1079.

K. Zhang, J. Lu, G. Lafruit, R. Lauwereins, and L. Van Gool. 2009b. Real-time accurate stereo with bitwise fast voting on CUDA. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*. IEEE, 794–800.

Lu Zhang, Ke Zhang, Tian Sheuan Chang, Gauthier Lafruit, Georgi Krasimirov Kuzmanov, and Diederik Verkest. 2011. Real-time high-definition stereo matching on FPGA. In *Symposium on Field Programmable Gate Arrays*. 55–64.