A Real-time 3D Disparity-map Acquisition Hardware Architecture for a Multi-sliding-window-operation

Jong hak Kim, Chan oh Park, Jueng hun Kim and Jun dong Cho
Department of Electrical and Computer Engineering
Sungkyunkwan University
Suwon, Korea
jhakkim@vada.skku.ac.kr

Abstract— **As requirements of 3D contents have been increased, a matching algorithm to obtain a disparity-map becomes vibrant research field. This processing includes a multi-sliding-window-operation (MSWO) which requires high memory and processing-time consumption. In this paper, we propose an effective hardware architecture with convergence of on-chip memory and shift registers, and parallelized cores. We utilize census as a matching algorithm and a 7 by 7 window in a 160 by 90 image, with search range length of 42. We synthesize on Vertex 5 from Xilinx, operating at 100 MHz clock. Our proposed method has lower memory consumption, and search range length times faster than previous one.**

Keywords-component; real-time image processing, multi sliding window operation, individual line buffer, shift register, census, 3D;

I. INTRODUCTION

As an image processing technology has been improved, various applications which consist of multi-functions, a real-time system and a high resolution operation are developed. And some of them such as computational photography and a smart camera as single lens system applications require more powerful computability. A frankencamera which is an experimental platform for computational photography has the real-time hardware architecture with a software stack for experimenting various developed algorithms as in [1]. And a MeshEye which is a hybrid-resolution smart camera mote is suitable for camera applications and also network ones as in [2]. Therefore, those two applications need high computability for real-time and high resolution operations and effective memory management for multi-functions.

In the stereoscopic system, almost all of applications are based on analyzing a disparity-map. To obtain a disparity-map, we operate matching algorithms using left and right images both. And since they compare one block in the left image to plural blocks in the right image to find a matching point, they usually include the multi-sliding-window-operation (MSWO) which is critical to processing time and memory consumption. In addition, visual fatigue which occur dizziness and annoyance based on accommodation, convergence discrepancy and etc. as in [3] has risen as a significant issue. And as one of solutions to evaluate visual fatigue, a disparity-map is also analyzed and controlled as in [4].

For all these reasons, parallelized and pipelined architecture with effective memory management is indispensable. In this paper, we proposed high speed disparity-map acquisition hardware architecture, especially for the MSWO. We utilized individual line buffers [5] to decrease memory consumption, shift-register [6] for date-reuse, and parallelized architecture to increase processing speed. A census algorithm which is suitable for real-time hardware architecture is implemented to obtain a disparity-map.

The rest of paper is outlined as follow. We introduce a census algorithm and an additional preparatory step: search range estimation to decrease processing time in Section II. Our proposed hardware architecture will be presented in Section III. Experimental results will be given in Section IV, and finally, in Section V, we draw the conclusion.

II. DISPARITY-MAP AQUISITION ALGORITHM

A. Basic Cencept of Census

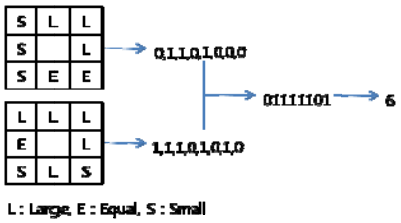

Figure 1. Example of Census process.

The census algorithm calculates hamming distance values and decides maximum value as a disparity. We could divide this process into four steps.

First, we compare between a target pixel, which locates x, y coordination in the image, and neighbor pixels to label 1 or 0 as in (1).

$$\mathrm{diff}_{(x,y)}(i,j) = \begin{cases} 1 & \text{if } img(x+i, y+j) > img(x,y), \\ 0 & \text{if } img(x+i, y+j) \le img(x,y). \end{cases} \quad (1)$$

Second, we decide the same diff value in the left and right images as 1 to calculate hamming distance as in (2).

$$h(i,j) = \begin{cases} 1 & \text{if } \mathrm{diff}_{\mathrm{left}}(i,j) = \mathrm{diff}_{\mathrm{right}}(i,j), \\ & 0 \text{ else.} \end{cases} \quad (2)$$

Third, we sum up hamming distance values in a window as in (3). W and H are width and height of window, respectively. And k is from minimum disparity to maximum disparity which is the same as length of search range. Fig. 1 revises from step one to three.

$$s(k) = \sum_{i=1}^{W} \sum_{j=1}^{H} h_k(i, j) \quad (3)$$

And finally, we obtain a disparity value (DV) from finding the maximum value among hamming distance values as in (4). Therefore, we need computation of the MSWO to obtain one disparity value.

$$DV = \max(s(k)) \quad (4)$$

B. Multi-Sliding-Window-Operation (MSWO)

The MSWO is the extension of the single-sliding-window-operation. The single-sliding-window-operation needs one image and one sliding window. On the other hand, as shown in Fig. 2, block matching algorithms such as SSD(Sum of Squared Difference), SAD(Sum of Absolute Difference), Census, which apply the MSWO, need left and right images both to compare between one window from the left image and k windows from the right image to obtain k hamming distance values where k is a search range. Then we find the maximum value as the disparity of the target pixel among obtained hamming distance values. Therefore, it consumes processing-time at least k times more.

Figure 2. Example of the multi-sliding-window-operation.

C. Search Range Estimation

To decrease processing-time, a search range estimation process could be applied by preliminary examination of images locally and normalized search ranges as in [7], which has two steps.

First, we examine one quarter of images and accumulate hamming distance values of examined pixels. Then second, we find the maximum value and divide by 16 to establish a threshold. Finally, we subtract threshold from each hamming distance value, and decide valid search range value where a hamming distance value is over 0. Fig. 3 shows estimated search range.

Figure 3. Search range estimation process.

Thus, we obtain the decreased search range for disparity estimation process which is needed to examine all pixels densely. Applying the search range estimation could be an effective method to decrease processing-time. However, since this normalized search range is improper for some pixels, noise and holes issues could occur. And the proposed method processes data in k windows at once.

III. PROPOSED HARDWARE ARCHITECTURE

As described in Section II, to compute a disparity value of one target pixel using n by n window in an M by N image, we need n^2 pixel values of the target pixel and around it in one window from the left image and n by (k + n - 1) pixel values in k windows from the right image for a MSWO. To obtain entire pixel values as fast as possible, we need efficient memory management. And to compare one window to k windows at once, we also need parallelized cores. Therefore, as shown in Fig. 4, proposed hardware architecture utilizes individual line buffers with registers and parallelized plural cores.

Figure 4. Proposed hardware achitecture.

Figure 5. Example of serial to parallel data pipeline architecture.

A. *Memory and registers*

Individual line buffers consist of two-line buffers as in [5]. And it shows an efficient tradeoff between processing time and memory resources for conventional sliding window operation. However, if we apply a large size of window or MSWO, it consumes excessive memory. And as shown in Fig. 5, reusing data and loading plural data at once could be accomplished without individual line buffers as in [6]. However, it causes high complexity. Therefore, we combine each individual line buffer and shift registers, so that we could decrease memory consumption with low complexity by reusing data and loading plural data from the registers as shown in Fig. 6.

Figure 6. Proposed memory management unit.

The operation of individual line buffers are based on "ping-pong scheme". First, an input controller selects that which individual line buffers are operated. Second, after storing n lines image data on upper line of two lines in individual line buffers, we switch to lower line to store data and transfer image data from upper lines to each shift register that is connected to each individual line buffer. Then we repeat process but only switching to upper line and transferring from lower line. And shift registers load data at once. To accomplish this process, we need n by n individual line buffers if we apply without shift registers.

Processing time of pipelined census cores could be the same as one census core. Therefore, the most influential factor of entire processing-time to compare to conventional methods is storing and loading time of memory which consumes one clock per one pixel. Let's assume that a size of image, which is 8 bits gray, is M by N. And a size of window is n by n, and the total processing time, storing time and loading time denote as T_T, T_S, T_L, respectively. And T_T is equal to $T_S + T_L$.

To decide the size of minimum image buffer, we should consider storing and loading time. If storing time is faster than loading time, we have to store entire image to avoid overwriting. And if storing time is the same as loading time, we could store only n lines of the image. Then in the other case, cores have to be delayed until storing enough pixel values. Therefore, storing and loading time per one pixel are one clock and n^2 by k clocks using conventional 1-way method. Then storing and loading time of our method and individual line buffers only as in [5] are both one clock per one pixel. Since conventional method is the first case, an M by N image should be stored. Based on this fact, T_S for the conventional method is M by N clocks, and memory consumption is M by N bytes. And since other methods are the second case, T_S could be M by n. However, memory consumption of individual line buffers only as in [5] is k times more than our method. And T_L is each loading time by the number of examined target pixels. Therefore the total processing time of 1-way is as:

$$T_T^1 = T_S^1 + T_L^1 = M \times N + (M \times N \times n^2 \times k) \quad (5)$$

And the total processing time of individual line buffer and our method with (n by n)-way parallel processing is as:

$$T_T^{nxn} = T_S^{nxn} + T_L^{nxn} = M \times n + (M \times N) \quad (6)$$

The total processing time of parallel processing methods are the same. However, our method consumes less memory than individual line buffers only as in [5].

Figure 7. Example of the census core where n is 3.

B. *Census cores*

Since memory management unit transfers entire image data in one window from the left image and k windows from the right image, census cores should be parallelized and pipelined effectively. We applied 42 parallelized census cores which are the same as the length of search range. And as shown in Fig. 7, each census core consists of four stages. In the first stage, we compare between the target pixel and neighbor pixels as in (1). And we decide that compared values from the left and right images are the same or not to obtain hamming distance in the second stage as in (2). Then we sum up hamming distance in the rest stages as in (3). The output of each census core includes hamming distance and search range value. Then we

compare 42 hamming distance values from each census core and search the maximum. Finally, we obtain a disparity value from the search range value of the maximum hamming distance as in (4).

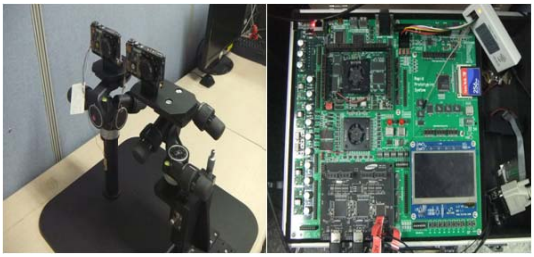

Figure 8. Experiment platform.

IV. Experimental Results

All modules are designed with Verilog and synthesized on Virtex 5-VLX330 FPGA using Xilinx's ISE. Fig. 8 shows the experiment platform. Input data are transferred from each camera to two HDMI input ports. And a HDMI tile synchronizes two input data from two cameras. The image processing is operated on FPGA. Then we obtained a result on the monitor which is connected to an output port.

Our proposed architecture focuses on real-time processing. Since conventional 1-way method is the worst case, we designed the pipelined 1-way method to compare effectively. As shown in Table 1, processing time of the pipelined 1-way method is k times faster than conventional 1-way one. And (n by n)-way methods, which are individual line buffer only and the proposed method, are k by n^2 times faster than conventional 1-way method. However, memory capacity of individual line buffers only method could consume more if we apply a large size of window or a MSWO. And our proposed method utilized 7 by 7 window and 42 search range. Therefore, individual line buffers only method consumes the highest memory capacity which is 15040 bytes, and 1-way processing methods, which are conventional 1-way and pipelined 1-way, consume equally 14400 bytes. Then only 2240 bytes and shift registers are consumed with our proposed method. Fig. 9 shows results image.

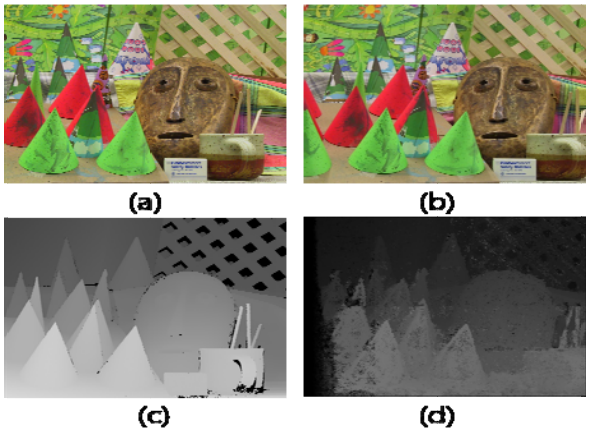

Figure 9. Result images. (A : Left image, B : Right image, C : Ground Depth image, D : Census result image)

TABLE I. Compare proposed method with previous methods on XC5VLX330-1FF1760

Used methods	Source	
	Processing time (sec)	***Memory consumption (Bytes)***
Conventional 1-way	296 ms	14400 byte
Pipelined 1-way	7.056 ms	14400 byte
Individual line buffers only	155.2 us	15040 byte
Proposed architecture	155.2 us	2240 byte + shift registers

V. Conclusion

In this paper, our proposed method achieves real-time processing with effectively pipelined architecture and low memory consumption by convergence of individual line buffers and shift registers. Considering improvement of accuracy, we should apply advanced matching algorithm with complex computation. Our experimental result shows good trade-off between processing time and memory consumption, which is suitable for a complex algorithm with a MSWO.

Acknowledgment

This research was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (20110016012).

References

[1] Andrew Adams, Eino-Ville Talvala, Sung Hee Park, David E. Jacobs, Boris Ajdin, Natasha Gelfand, Jennifer Dolson, Daniel Vaquero, Jongmin Baek, Marius Tico, Hendrik P. A. Lensch, Wojciech Matusik, KariPulli, Mark Horowitz and Marc Levoy, "The Frankencamera: An Experimental Platform for Computational Photography," ACM Trans. Graph. (SIGGRAPH) 29 (2010), 29:1-29:12.

[2] S. Hengstler, D. Prashanth, S. Fong and H. Aghajan, "Mesh-Eye : A Hybrid-Resolution Smart Camera Mote for Applications in Distributed Intelligent Surveillance," in Proceedings of the 6th International Symposium on Information Processing in Sensor Networks(IPSN'07), pp.360-369,2007.

[3] Sumio Yano, Masaki Emoto, Tetsuo Mitsuhashi, "Two factors in visual fatigue caused by stereoscopic HDTV images," Displays 25:141-150(2004).

[4] Jaeseob Choi, Donghyun Kim, Bumsub Ham, Sunghwan Choi and Kwanghoon Sohn, "Visual fatigue evaluation and enhancement for 2D-plus-depth video," 2010 17th IEEE International Conference on Image Processing(ICIP), pp.2981-2984, 2010.

[5] Jong-hak Kim, Jun-dong Cho, "A Real-time 3D Image Refinement Using Two-line Buffers," 2011 13th International Conference on Advanced Communication Technology, pp. 778-781, 2011.

[6] Michael I. Gordon, William Thies, and Saman Amarasinghe, "Exploiting coarse-grained task, data, and pipeline parallelism in stream programs," In 14th International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 151–162, New York, NY, USA, 2006. ACM Press.

[7] Hyoung-seok Ko, Jun dong Cho, "Real-time Stereo matching system for high-definition images," Thesis,(Master). Sungkyunkwan Univ., pp. 43-46,2011.