Real-time Smoothing Filter for Three Dimensional Disparity Map Algorithm and Hardware Implementation

Jueng hun Kim
Sungkyunkwan University
School of Information and Communication
Suwon, Korea
jhunkim@vada.skku.ac.kr

Jong Hak Kim
Sungkyunkwan University
School of Information and Communication
Suwon, Korea
jhakkim@vada.skku.ac.kr

Ham Hun Ho
Sungkyunkwan University
School of Information and Communication
Suwon, Korea
hhham@vada.skku.ac.kr

Jun Dong Cho
Sungkyunkwan University
Department of Mobile system engineering
Suwon, Korea
jdcho@skku.edu

***Abstract*—Nowadays, 3D-image processing of stereovision that uses two camera lenses has been a vibrant research filed. This paper proposes an efficient hardware architecture for high performance and real-time smoothing filter that is applicable to enhance the disparity map image from 3D stereovision system. First, we maximally utilized the parallel and pipeline operations. Second, we adopted 11 by 11 sparse mask operations instead of using the 33 by 33 window to achieve faster processing time without sacrificing the quality obtained from 33 by 33 mask operation. Furthermore, our architecture showed faster in processing time and smaller in errors on the images, compared with the normal 33 by 33 mean filter. Our verification was accomplished on the Virtex5 XC5VLX330 FF1760 FPGA of Xilinx with using 100MHz system clock. In a 1280 by 720 video frames, our verification shows that the real-time image was processed with 325fps. Our achievement shows remarkably higher operation rates than the existing ordinary mean filtering method.**

I. Introduction

Recently, motivated by the demand of 3D movie and TV, the quality of 3D stereovision has been improved dramatically. Thus, a lot of image-processing algorithms, e.g, and image filtering to remove noise have been recently developed. In the process of image encoding and transmission, when the image passes the noised transmission lines, the main reason for image degradation is the influence of salt-pepper noise, that is, positive and negative pulse noise. Salt-pepper noise means random white gray value and block gray value. To remove the noise usually use mean filter.

However, mean filter may change the gray value of some uncontaminated pixels, moreover, when the image is contaminated badly, it cannot discriminate the real noised pixels due to its quickly declined de-nosing performance. A software simulation method [1, 4] overcame the disadvantage of smoothing filtering.

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology (20110016012).*)*

These algorithms are slow or unsuitable for real-time operation. Compared with software simulation of Gaussian smoothing filter [4], our hardware structure show faster and better result. We applied smoothing filter with sparse mask especially to remove error on the depth map image.

The main contributions of this paper are as follows:

1. Our proposed method is qualified to reduce noises on the 3D- disparity image.
2. This proposed hardware can be used for not only stereovision system but also multi-vision system.
3. We increase the speed of calculation using maximal pipelined and parallel processing.

The reminder of this paper is organized as follows.

Section 2 introduces problem formulation. Section3 introduces smoothing filter algorithm. Section 4 introduces hardware architectures. Section 5 introduces the experiment result. The last section shows the conclusion of this paper.

II. Problem Formulation

A. Comparision ordinary mask and sparse mask.

Since smoothing filter uses neighbor pixels as figuring out of information of target pixel, the large size of mask releases the food quality of image. From out result of simulation, the speed of the serial process of smoothing filter makes 1 pixel per 121 clock cycles. Since all pixels in the mask should be calculated to determine target pixel, this speed is not qualified for real-time image processing.

$$C = \frac{1}{K}\begin{bmatrix} p_0 & p_1 & p_2 & p_3 & \dots & p_{32} \\ p_{33} & p_{34} & p_{35} & p_{36} & \dots & p_{65} \\ p_{66} & p_{67} & p_{68} & p_{69} & \dots & p_{98} \\ p_{99} & p_{100} & p_{101} & p_{102} & \dots & p_{131} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{1057} & p_{1058} & p_{1059} & p_{1060} & \dots & p_{1080} \end{bmatrix}_{K=33\times 33} \qquad (1)$$

Equation (1) shows the ordinary 33 by 33 mean filtering. Proposed smoothing filter used sparse pixels so that the number of node and operation time is reduced. Suppose 33 by 33 mask mean filtering takes accumulation time with 1089 cycles.

$$C = \frac{1}{K}\begin{bmatrix} p_0 & 0 & 0 & p_1 & 0 & 0 & p_2 & \dots & p_{10} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ p_{11} & 0 & 0 & p_{12} & 0 & 0 & p_{14} & \dots & p_{21} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ p_{22} & 0 & 0 & p_{23} & 0 & 0 & p_{24} & \dots & p_{32} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{110} & 0 & 0 & p_{111} & 0 & 0 & p_{112} & \dots & p_{121} \end{bmatrix}_{K=121} \qquad (2)$$

On the other hand, sparse mask (11 by 11) operation time takes 121 cycles. The operation time is reduced 9 times compare with ordinary mask. Equation (2) shows the matrix for 11 by 11 sparse mask.

Our goal is to obtain a better result as well as speed. We decided to use sparse mask scheme for the faster performance yet to keep the quality of image result. We applied hardware implementation using memory partitioning, with parallel and pipeline structure. By adopting those implementation methods, the computational times are increased up to 121 times.

B. Comparison image quaility between mean filter andproposed smoothing filter.

We compared the mean filter and smoothing filter to know the quality of disparity map image. The Figure 1 shows the disparity map image from census algorithm [5].

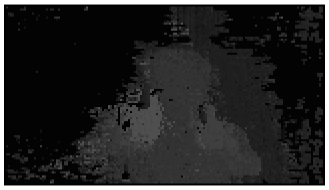


Figure 1. The image of Disparity map.

The Figure 2 shows the mean filtering result using 33 by 33 mask. The picture shows that the image turned blur, yet the edge of object cannot be found. Even the computational time was slower than proposed smoothing filter. To solve the operation time and increase the quality of image, we use the proposed smoothing filter as shown in figure 3. The result of smoothing filter has the better quality, keeps the edge of objects, makes image blur.

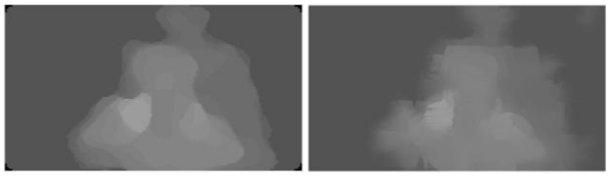


Figure 2. The image of Applying the mean filter(left), and smoothing filter (Right)

III. SMOOTHING ALGORITHM

Our proposed smoothing filter algorithm is based on matching two images and developing one new result image with sparse mask operation. One image frame is gray scale image from the real scene and the other image is disparity map image from census algorithm [5].

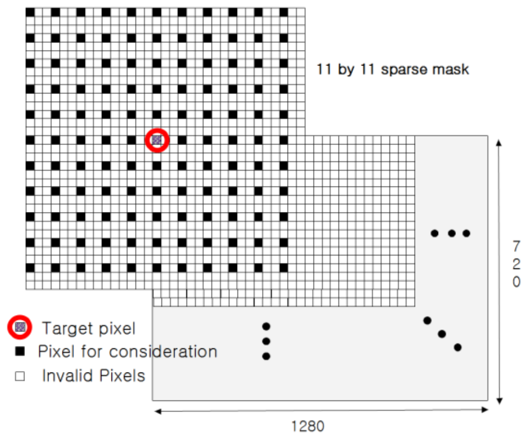


Figure 3. Our proposed 11 by 11 sparse mask extraced from 33 by 33 original mask..

The disparity map image is referred to as “disparity” between the right and left side of images generated from two real scenes taken by two cameras. Our smoothing filter function calculates average value from 121 pixels within the range of 11 by 11 sparse mask. The actual mask size is shown in Figure 3 with the coverage of 33 by 33 pixels. We used black color pixels in Figure 3 so as to build 11 by 11 size of mask. We choose the 3n-2th pixel along the vertical and horizontal lines to reduce mask operation time and sizes.

This 11 by 11 sparse mask makes the image blur and still removes the errors such as pepper and salt effects, and holes. Our experiments result showed a comparable or better result to compare with case of 33 by 33 mask. Moreover, we reduce the size of mask operation so that the performance also increased. The equation (3) shows the proposed smoothing filter calculation.

P= transformed pixel value.

$$p=\frac{\sum_{\alpha=-\frac{n-1}{2}}^{\frac{n+1}{2}}\left(\sum_{\beta=\frac{n-1}{2}}^{\frac{n+1}{2}}(x+2\alpha,y+2\beta)\right.}{n^2} \quad (3)$$

IV. HARDWARE ARCHITECTURE

Figure 4 shows the overall hardware architecture of the smoothing filter. The hardware will generate 1280 by 720 - 8bit size of result image. This hardware flow consists of 7 steps: 1) getting the gray and disparity map image from each memory (1280 by 720 sized image will be released pixel by pixel) 2) compare between gray image value (compare) 3) perform multiplexing for releasing depth value; 4) store and wait for parallel and pipeline processing (R1~R11); 5) Sum up for accumulator output pixels (AC and Carry look ahead adder); 6) figure out depth value counting (depth counter); 7)divide depth value from depth value counter (Div.) After these process, the final result is stored in DDR2 RAM.

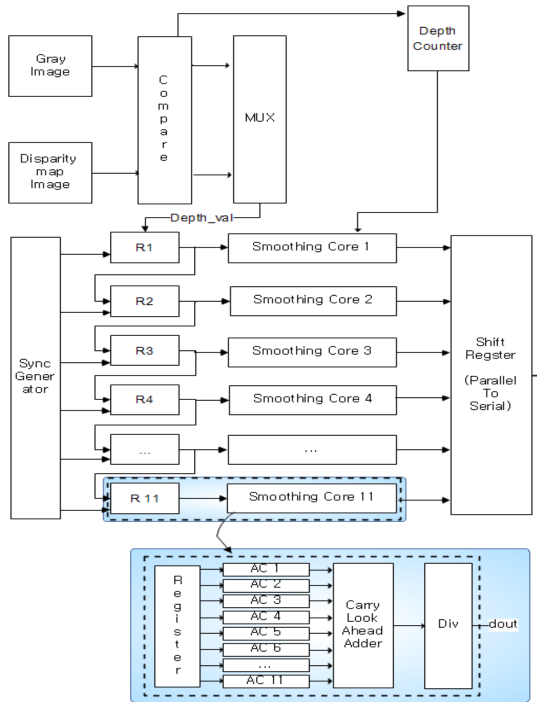
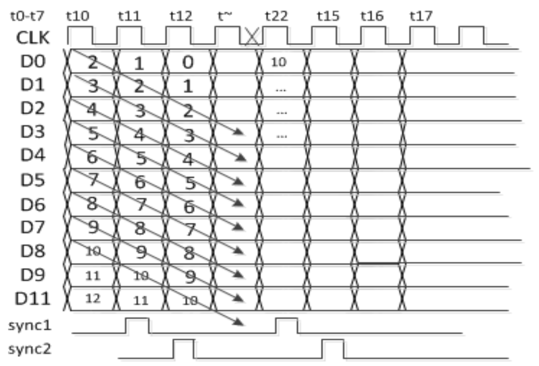


Figure 4. The image of Applying the proposed smoothing filter.

Due to the limitation of using internal block ram in the Xilinx FPGA chip, we used registers to be formed as a Chain structure. The R1 has eleven 8bit flip-flops to store pixels. The eleven registers to R1-R11 formed as pipeline structure. All pixels are can be calculated simultaneously at the following synch signal.

Each register connected to side by side and each output of flip-flops is entered into data path. Using an enable signal in the flip-flop, the set of registers organized vertically will work alternatively. Using data reuse method with these methods, the controlling method for parallel processing and data reuse is possible. The specific example of data path and data flow for the data reuse operation is shown in figure 5.

Figure 5. Our proposed Parallel and pipeline processing scheme for data reuse



A. *Memory Partitioning*

Generally, line buffer is used for storing image pixel data. However, a huge deep line buffer memory consumes more power and decrease the speed for input output access [6]. We used several small-sized memories using memory partitioning, as depicted in Figure 4. We organized a set of D-type flip-flops (R1~R11) instead of using a huge line-buffer or a set of small-sized line buffer.

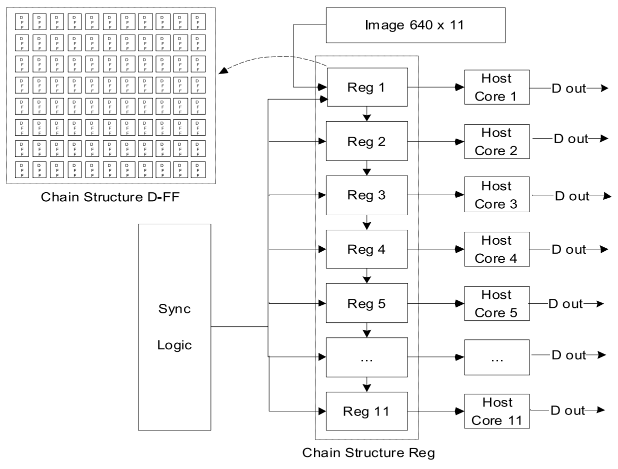


Figure 6. The example of memory partitioning using Registers.

B. *Accumulator implementation*

The figure 7 shows the single accumulator [3] that we used for summation operation.

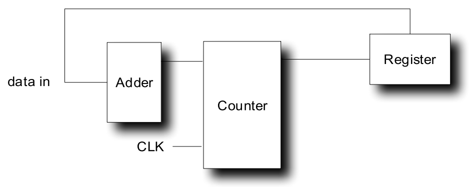


Figure 7. The accumulator for summation.

The single accumulator performance is fast enough for single operation. However, we had to sum 121 pixels with considering middle of target pixel. This single process releases a pixel per 121 clock cycles.

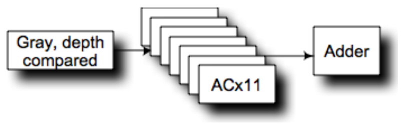


Figure 8. The parallelized accumulator

We made parallel accumulator to solve operation time problem shown in figure 8. We used 11 accumulators to get the result for 12 clock cycles. The total summation of depth value for a mask operation is released at a cycle. However, this speed is little slow for real time operation. To achieve the goal performance, we used the 11 accumulation logic eleven times. As a result, we could get a pixel at a cycle.

V. EXPERIMENT RESULT

All modules are implemented with Virtex5-VLX330 FGPA using Xilinx-ISE 12.3 version programmed by Verilog HDL code. Figure 9 and 11(Left) shows the gray scale image from original image and figure 9 and 11(Right) shows the disparity map image from 3D census algorithm [5]. Figure 10 and 12 shows the smoothing filter remove error in the disparity map image so that image is much clearer. Furthermore, edges of objects in the image are enhanced. As a result, we have much accurate object information.

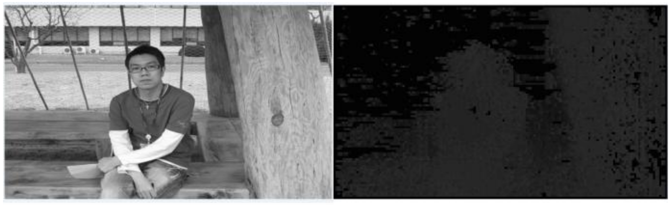


Figure 9. Gray scale image(left) Disparity map image (right) I.

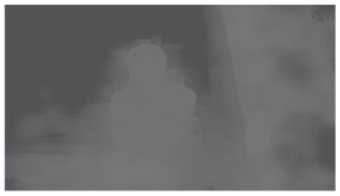


Figure 10. Our proposed Smoothing filter result image I.

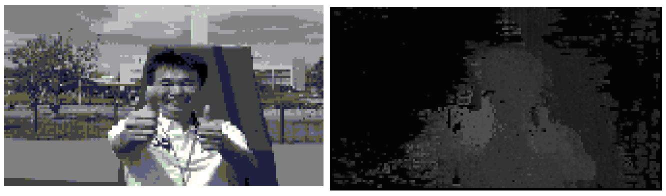


Figure 11. Gray scal image(left) Disparity map image(right) II.

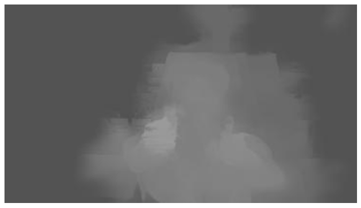


Figure 12. Our proposed Smoothing filter result II.

The processing speed of our smoothing algorithm reached approximately 10640 clocks. This speed is practical for real time operation. Our hardware architecture runs 121 times faster than the serial architecture. Nonetheless, parallel with pipeline structure is occupied the size of chip and consume more power.

TABLE I. COMPARISON BETWEEN SERIAL, PARALLEL, AND PARALLEL WITH PIPELINE PROCESSING

Method	**Serial**	**Parallel**	**Parallel with Pipeline**
Speed (100MHz)	**1 pixel per 121 clock cycle**	**1 pixel per 11 clock cycle**	**1 pixel per 1 clock cycle**
Slice LUTs	982	1515	2774
Register	1081	1641	2032
Data Width	8bit	96bit	1056bit

VI. CONCLUSION

Our proposed algorithm and hardware architecture achieves real-time processing with shared-time, and reading-time is adjusted to match core`s processing-time and pipeline whole architecture. This filter makes it possible to find objects in the depth image better than without smoothing filter.

We conclude that proposed smoothing filter can be used to remove image noise and improve the disparity map information. In the future, we will implement this hardware design for small mobile device.

REFERENCES

[1] Sean G Patronis and Linda S. DeBrunner, “Sparse FIR Filters and the Impact on FPGA Area Usage”, Signal, Systems and Computers, 2008 42nd Asilomar Conferrence, pp. 1862-1866, Oct. 2008.

[2] Welson Sun and Michael J. Neuendorffer, “FPGA Pipeline Synthesis Design Exploration Using Module Selection and Resource Sharing”, Computer-Aided Design of Integrated Circuits and systems, pp. 254, Feb. 2007.

[3] Behrooz Parhami, “Computer Arithmetic Algorithms and Hardware Design”, NY Oxford University Press 2000.

[4] Pei-Yung Hsiao, Shin-Shian Chou, and feng-Cheng Huang, “Generic 2-D Gaussian Smoothing Filter for Noisy Image Processing, TENCON 2007 IEEE Region 10 Conferrence, pp. 1 Oct. 30 2007.

[5] Chang, N.Y-C, Tsung-Hsien Tsai, Bo-Hsiung Hsu, Yi-Chun Chen, and Architecture of Disparity Estimation With Mini-Census Adaptive Support Weight”, IEEE Trans. On Circuit and System for Video Technology, pp. 792, June 2010

[6] Yoon-Gu Kim and Yong-Jin Jeong, “Low Power Design of Filter Based Face Detection Hardware”, IEEK, pp. 89-95, July. 2008.