International Journal of Hybrid Information Technology
Vol. 10, No.1 (2017), pp. 339-360
http://dx.doi.org/10.14257/ijhit.2017.10.1.29

FPGA Architecture for Real-time Depth Estimation System

Ran Liu[1,2,3], Donghua Cao[2,3], Zekun Deng[1], Zhenwei Huang[1], Miao Xu[2,3], Ruishuang Jia[2,3], Dehao Li[2,3] and Mingming Liu[2,3]

[1]*College of Communication Engineering, Chongqing University, Chongqing 400044, China*
[2]*Chongqing Key Laboratory of Software Theory & Technology, Chongqing 400030, China*
[3]*Key Laboratory of Dependable Service Computing in Cyber Physical Society of the Ministry of Education, Chongqing University, Chongqing 400044, China*
ran.liu_cqu@qq.com

Abstract

Depth estimation has been widely concerned. It estimates the depth map from a single-view image or multi-view images, which is an essential step for 2D-to-3D conversion. However, most of the depth estimation algorithms suffer from high computational complexity and high memory cost. This paper presents an overall FPGA architecture for real-time depth estimation system based on an improved hardware-friendly algorithm. In this architecture, two transposition modules, row-to-column and column-to-row transposition modules, are designed to rotate the images so as to meet the requirements of the algorithm. These modules can also be used for any other situations that need to rotate the image during hardware processing. In order to reduce the computational complexity, equations with power and division operations are substituted by the approximate equations with addition and subtraction operations. In addition, hierarchical pipelining is used to reduce the memory cost as well as improve the operating frequency, and the external memory is used to reduce the internal memory cost. Experimental results reveal that the proposed system can support real- time depth estimation and achieve reasonably good depth estimation results for the scenery videos.

Keywords*: 3D TV, 2D-to-3D conversion, depth estimation, depth map, FPGA*

1. Introduction

With the fast development of display technology, *Three Dimensional Television (3D TV)* becomes more and more popular [1, 2]. More and more people desire a wonderful *3D experience* through TV at home [3, 4]. However, the lack of 3D content has become one of mass market barriers for 3D media service to the home [5]. If the abundant 2D video can be converted to 3D video in real time, it can not only provide rich material for 3D display, but also significantly reduce the production cost [6, 7]. Hence, *2D-to-3D conversion* has become the key technique to 3D TV.

Depth map plays an important role in 2D-to-3D conversion, for it contains *depth information* of objects and it is essential for a *depth-image-based rendering (DIBR)* system. *Depth estimation*, which estimates the depth map from a single-view image or multi-view images [8, 9], is the common method for depth map acquirement. Since only a single-view image is available in 2D-to-3D conversion when implemented by hardware in this paper, the "depth estimation" mentioned in the rest all refers to estimate depth map from single-view image, unless otherwise specially stated.

Many researchers have tried to make accurate depth estimation so as to synthesized

ISSN: 1738-9968 IJHIT

satisfactory 3D views [10]. Usually, the *monocular depth cues*, such as *linear perspective*, *relative height*, *shadow*, *occlusion*, *texture*, *focus/defocus* and *color contrast* are extracted for estimation [7, 9]. For example, Jung *et al* [11] proposed a 2D-to-3D conversion technique based on *relative height depth cue*. They obtained the depth maps according to the *line tracers*. Different from the methods utilizing monocular depth cue, Jae-Il *et al* [6] proposed a depth estimation method using *object classification* based on the *Bayesian learning algorithm*. Using training data of six attributes, they categorized objects in a single-view image into four different types. Kim *et al* [12] proposed a 2D-to-3D conversion system based on *visual attention analysis*. Depth map is substituted by the generated visual attention map. Po *et al* [13] presented an automatic depth estimation algorithm using *block-based depth* from *motion estimation* and *color segmentation* for *depth map enhancement*. However, the computational complexity will be dramatically increased when video resolution increased (such as HD 720p or 1080p). Software implementation cannot satisfy the demand for *real-time* processing any more. For example, *DERS (depth estimation reference software)* costs more than 30s to generate one frame of the depth map with resolution of 640 × 480 [14, 15]. Therefore, *hardware implementation* is necessary and crucial.

Currently, there appear to be few studies on hardware implementation of depth estimation. Banz *et al* [16] proposed a real-time stereo vision system using *semi-global matching disparity estimation*. The system was based on stereo vision and could generate *disparity maps* of VGA images (640 × 480) with maximum disparity ranges of up to 128 pixels under real-time conditions (30 f/s) at a clock frequency as low as 39 MHz. Ching-Lung *et al* [17] proposed a real-time 1080p 2D-to-3D video conversion system, which could support real-time conversion of HD 1080p 2D video to 3D video with the throughput of 30 f/s. Li *et al* [14] proposed a hardware solution of real-time depth estimation based on stereo vision, which could reach the throughput of 131 f/s with the resolution of 1920 × 1080p. However, the system was based on *binocular disparity*, which could not satisfy the requirement of depth estimation from a single-view image.

In this paper, the algorithm based on relative height depth cue is modified for efficient hardware implementation. Equations with power and division operations in the original algorithm are substituted by the approximate equations with addition and subtraction operations to reduce the computational complexity. Though other algorithms may outperform the modified algorithm from the accuracy, they are much more complicated for hardware implementation and will result in higher hardware cost. Based on the *hardware-friendly algorithm*, an FPGA architecture for real time depth estimation system is proposed. Considering that the hardware-friendly algorithm is a column-based algorithm, a *row-to-column transposition* module is designed to rotate the row-based input image so it's rearranged in columns. In our FPGA architecture, *hierarchical pipelining* is used to reduce the memory cost as well as improve the operating frequency, and the external memory is used to reduce the internal memory cost. In addition, the FPGA architecture can be easily modified so as to support different resolutions. In the rest of this paper we take the resolution of HD 720p video as an example to illustrate the hardware implementation of depth estimation, unless otherwise specially stated.

The remaining portion of this paper is organized as follows. First, the hardware-friendly algorithm is described in Sec. 2. Then, hierarchical pipelining hardware architecture is presented in Sec. 3. Sec. 4 demonstrates the implementation results and the corresponding simulation results when the proposed algorithm and architecture are applied. Finally, Sec. 5 concludes the paper.

2. Hardware-Friendly Algorithm

In this section, we modify the algorithm based on relative height depth cue to optimize the trade-off between the accuracy of the depth map and the complexity. The original

equations with power and division operations in the algorithm are replaced by the approximate equations with addition and subtraction operations. The flowchart of the modified algorithm is shown in Figure 1. The algorithm consists of 4 steps: *color space conversion*, *edge detection*, *line tracing* and *depth assignment*. The details of each component are described as follows.

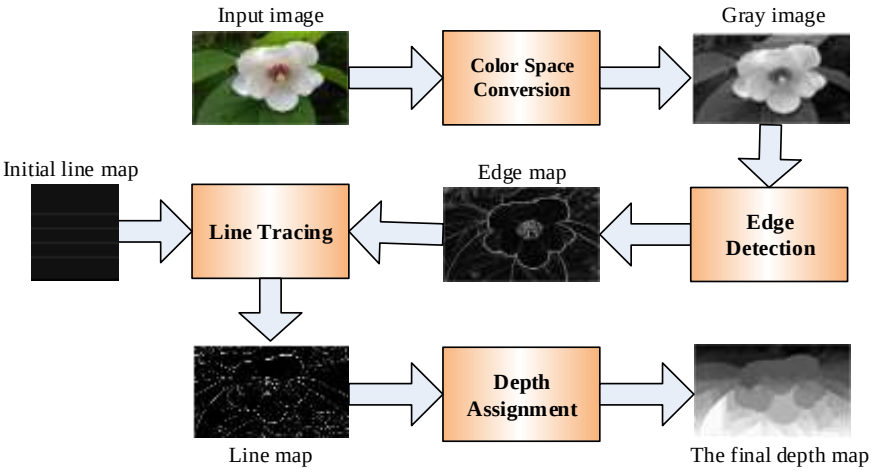


Figure 1. Flowchart of the Hardware-Friendly Algorithm

Step 1: *Color space conversion*

This step converts RGB image data to the YUV color space, hence in the following step component Y can be used as the gray value of a pixel approximately. Note that each component of the input pixel is 10 bits while each component of the output pixel is 8 bits.

Step 2: *Edge detection*

In this step, the classical *Sobel* operator is adopted to perform *edge detection*, and the *edge map* is outputted.

Step 3: *Line tracing*

Line tracing operation traces strong edge positions from the left border to the right border of the edge map, which generates a line map with horizontal and uncrossed lines [11]. The operation consists of two steps: ① parameter setting; ② line tracing from left to right.

① Parameter setting

Number of lines *n*: *n* is usually greater than 10 to guarantee the depth estimation result, and less than 80 to reduce the computational complexity. In this paper, *n* is set to 45 for HD 720p video. For HD 1080p video, *n* should be 68.

Line space d_{line}: As can be seen from Figure 1 and Figure 2, the line tracing starts with an initial *line map* in the initial state. The initial line map consists of n parallel lines with the same space (d_{line}) between them. Hence d_{line} equals int(H / n), where H represents image height. d_{line} is set to 16 for HD 720p video. It's also appropriate for HD 1080p video.

Horizontal coordinate of each line in initial line map: Due to the same space between lines, the initial horizontal coordinate of line i is set to $(H - 1) - i \times d_{line}$ (0-based). Note the number of line i is increased from bottom to top, as shown in Figure 2.

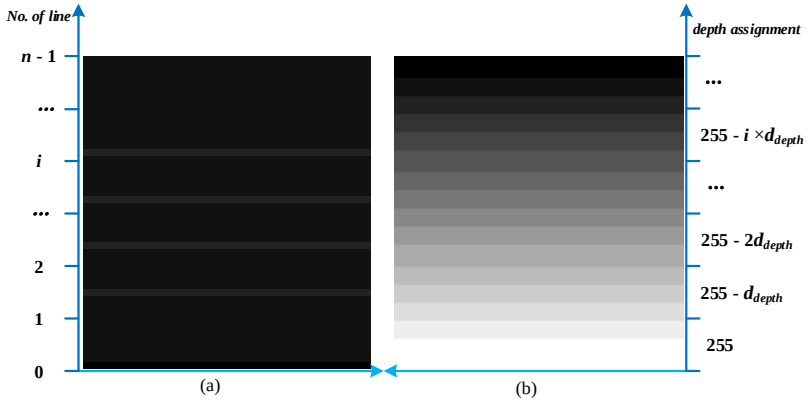


Figure 2. Initial Line Map with n (= 45) Lines and Initial Depth Map (H = 720)

② Line tracing from left to right

In this step, the *line tracer* of each *line traces* strong edges from an initial left point determined in step ① and selects the next point. For efficient hardware implementation, constraints in the original algorithm are modified as follows:

- *Edge tracing constraint*

$$E_1(x, y) = \exp(-edge(x, y) / a), \tag{1}$$

where $edge(x, y)$ and a denote the *edge value* at point (x, y) in the edge map and the control parameters, respectively [11].

It can be seen from Equation (1) that $0 < E_1(x, y) \leq 1$. Since $edge(x, y)$ is an 8-bit number from 0 to 255, Equation (2) can be used to approximate Equation (1). The range of value and the monotonicity of Equation (2) are the same with Equation (1), but Equation (2) is simpler for hardware implementation.

$$E_1(x, y) = (256 - edge(x, y)) / 256. \tag{2}$$

- *Smoothness, elasticity and total constraints*

The original smoothness E2(x, y) constraint, elasticity constraint E3(x, y) and total constraint T are described by Equation (3), (4) and (5), respectively.

$$E_2(x, y) = d_s(x, y) / b, \tag{3}$$

$$E_3(x, y) = d_e(x, y) / c, \tag{4}$$

$$T = \text{argmin}[\alpha E_1(x, y) + \beta E_2(x, y) + \gamma E_3(x, y)], \tag{5}$$

where α, β, γ are *weighting factors* for each constraint; $d_s(x, y)$ denotes the vertical distance between current point and the candidate point (x, y); $d_e(x, y)$ denotes the vertical distance between the initial point at the left border in the initial line map and the candidate point (x, y). Here α, β and γ are set to 4, 3 and 3, respectively.

The values of b and c are determined heuristically [11]. For HD 720p video, b and c are set to H / 4 (= 180). For HD 1080p video, b and c are set to H / 6 (= 180). Therefore, Equation (5) can be simplified to Equation (6) for both HD 720p and 1080p videos.

$$T = \text{argmin}[(256 - edge(x, y)) / 64 + d_s(x, y) / 60 + d_e(x, y) / 60]. \tag{6}$$

As can be seen from Equation (6), all the denominators are equal or approximately equal to 60. Hence we can simplify it to Equation (7). Equation (7) is the final equation used for depth estimation.

$$T = \text{argmin}[(256 - edge(x, y)) + d_s(x, y) + d_e(x, y)]. \quad (7)$$

Step 4: *Depth assignment*

Depth assignment must satisfy the following rules: depth values must be assigned in strictly descending order from bottom to top. Region between line $i + 1$ and line i will be assigned an identical depth value: $255 - i \times d_{depth}$, $0 \le i \le n - 1$, as shown in Figure 3. Here d_{depth} denotes the interval of depth value (equals to *int* $(255 / n)$) between two line traces. Thus, a staircase depth map will be generated.

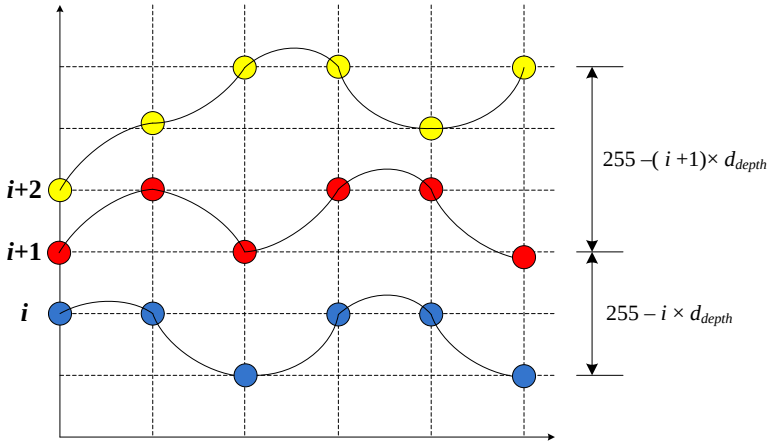


Figure 3. Illustration of Depth Assignment

3. Hardware Architecture

3.1. System Architecture

With the proposed hardware-friendly algorithms above, this section proposes an architecture for the depth estimation system.

Figure 4 shows the proposed architecture that consists of 7 components: edge detection, *edge data buffering*, *MPMC (multi-port memory controller)*, *edge data splitting*, depth estimation, *depth data buffering* and *depth data splitting*.

Edge detection component calculates edge map "*Edge*" from the current image line by line. As the depth estimation system is a column-based system, a row-to-column transposition module that consists of two components, edge data buffering and edge data splitting, is designed to rotate the row-based input image for column-based process. Depth estimation component receives the edge data and generates depth maps "*Dep*" column by column. Finally, the column-to-row module, which consists of depth data buffering component and depth data splitting component, rotates the column-based depth map to a row-based one for other applications. The details of each component are described as follows.

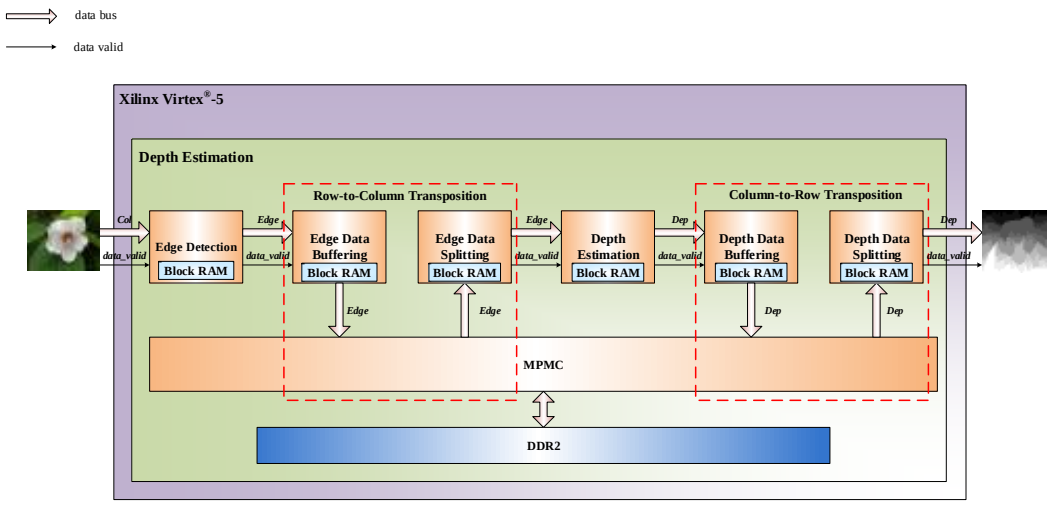


Figure 4. Overall Architecture of the Proposed Depth Estimation System

(1) *Edge Detection Component*

As mentioned in section 2, the classical Sobel operator is adopted for edge detection, in which *Block RAM* is used to buffer the needed gray data. For this technique, the output image depends significantly on the threshold selected which is an input to the Sobel operator. The threshold in this paper is set to 80. If the gradient magnitude is greater than 80, the corresponding edge data will be set to 255; otherwise, the edge data will be set to 0. Note that, since edge data at the boundary of image (the first and last row, the first and last column) have little influence on the estimated depth maps, they will be set to 0 in the proposed system.

(2) *Edge and Depth Data Buffering components*

The edge data buffering component is the first component of row-to-column transposition, which performs the function of buffering and restructuring edge data. In this component, the amount of Block RAM in need is sixteen, for the reason that the input data width is 8 bits and the output data width is 128 bits. The size of each Block RAM is set to $W \times 8$ bits, where W represents image width, in order to buffer all of the edge data in a row. For HD 720p video, W is set to 1280,as shown in Figure 4. And for HD 1080p video, it is set to 1920. After sixteen rows are all stored, the data in Block RAM will be read out simultaneously and the ones with identical read address will be restructured. The restructured data will be further stored into external memory *DDR* through MPMC component.

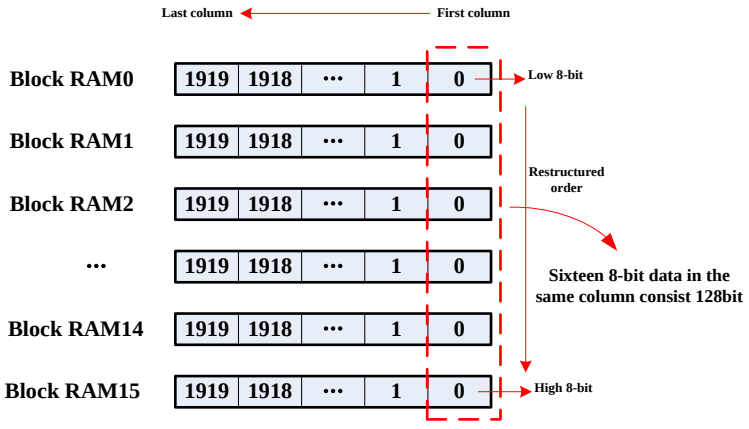


Figure 5. Schematic Diagram of the Component for 720p Video: Edge Data Buffering

The depth data buffering component is the first component of column-to-row transposition, whose function and hardware architecture are similar to those of the edge data buffering component. The main difference between the two components is the size of Block RAM. In depth data buffering component, the size is set to $H\times 8$bits, where H represents image height, so as to buffer all of the edge data in a column, and for HD 720p video and HD 1080p video, H is set to 720 and 1080 respectively.

(3) *MPMC Component*

MPMC component is the second component of both row-to-column transposition and column-to-row transposition, which controls the write and read operations of DDR. It stores edge maps and depth maps to DDR in the specific way presented in section 3.3, and reads them to the following component according to the timing requirement. In this component, the "*without a DCM(Digital Clock Management) but with a Testbench*" type of *MIG IP core* is adopted, in which *clk0*, *clk90* and other clock signals needed for this IP core are provided by the DCM. In order to meet the requirement of real-time processing, four address ranges are designed to buffer image frames, which are marked *E1*, *E2*, *D1*, *D2* in Figure 6. The schedule of write and read operations of DDR is shown in Figure 7.

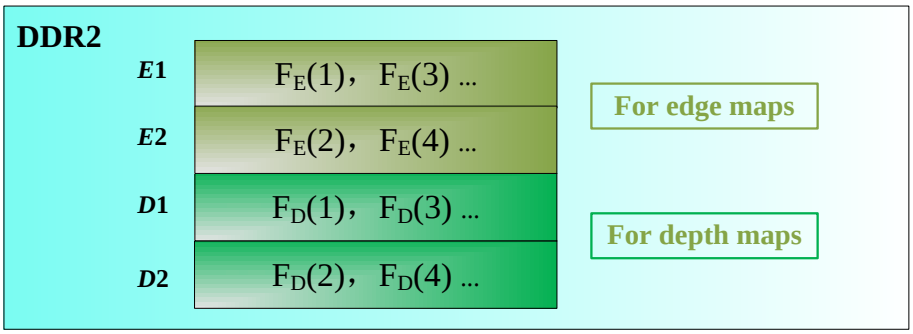


Figure 6. Address Distribution of DDR. $F_E(i)$, $F_D(i)$, *etc* Represent Edge Map and Depth Map

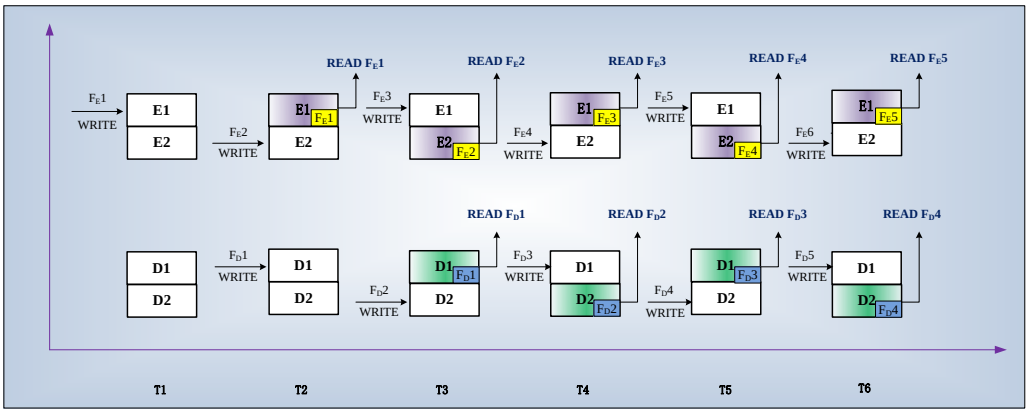


Figure 7. Schedule of Write and Read Operations of DDR

As shown in Figure 7, *E1* and *E2* are used to act as the *ping-pong buffer* for edge maps, which can be implemented by pipelining technology. Similarly, *D1* and *D2* are also used to perform as the ping-pong buffer for depth maps. The pipelining architecture and the corresponding operations of DDR will be thoroughly presented in section 3.3.

(4) *Edge and Depth Data Splitting Components*

The edge data splitting component is the last component of row-to-column transposition, in which the amount of Block RAM needed is four to split the output data

from MPMC component, since that the data in four columns is to be read out from DDR at the same time. The data in four columns are not separated, but mixed with each other in the fixed order as shown in Figure 8. In this process, the component first judges which column the current edge value belongs to, and then stores it to the corresponding Block RAM. After this, edge data in different columns will be stored into different Block RAM. Thus, the edge data can be transmitted column by column via reading the four Block RAM at different moments. The input data width of this component is 128-bit, while the output data width is 8-bit.

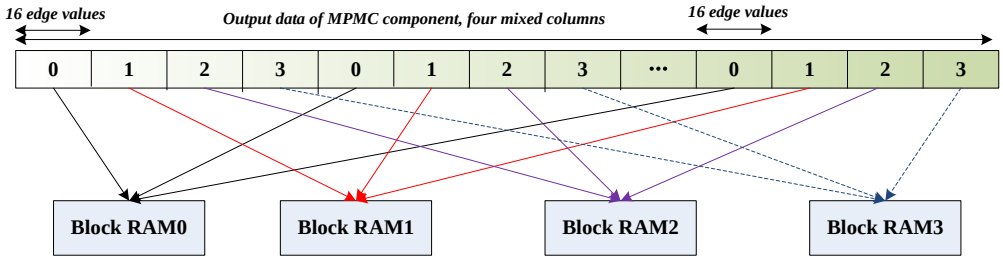


Figure 8. Schematic Diagram of the Splitting Operation. "0", "1", etc Represent which Column the Edge Value Belongs To

The depth data splitting component is the last component of column-to-row transposition. Similarly, the number of Block RAM adopted in the depth data splitting component is also four to implement the splitting operation. The structure of this component is similar to that of the edge data splitting component, which will not be presented in detail.

(5) *Depth Estimation*

The depth estimation component performs the function of estimating depth maps column by column. As described in section 2, this component can be divided into two parts: line tracing and depth assignment. In line tracing, parallel processing is performed to calculate the coordinates of all line traces in the same column. Depth assignment is implemented by controlling the write operation of Block RAM. The write address is first compared with the calculated coordinates to determinate which segmented region the write address belongs to. Then, the corresponding depth value of this region is written to the Block RAM. After implementing all addresses, a staircase depth map will finally be obtained.

3.2. Hierarchical Pipelining

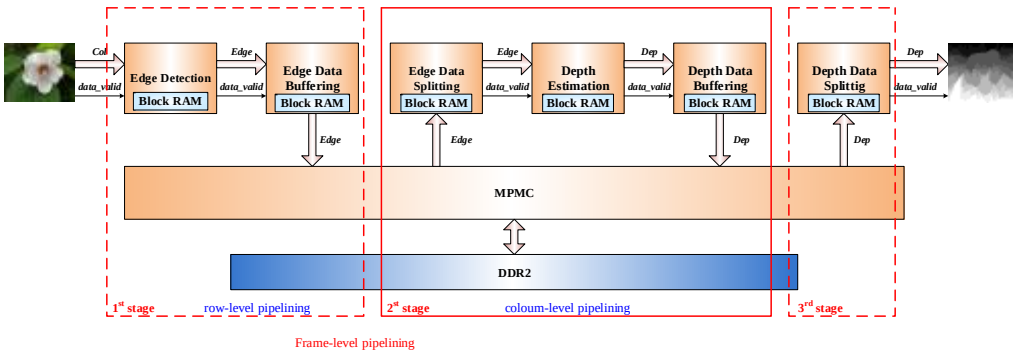


Figure 9. Proposed Hierarchical Pipelining Architecture

The hierarchical pipelining architecture is proposed to meet the requirement of real-time processing, which consists of frame-level, row-level and column-level pipelining, as shown in Figure 9. The detailed description is stated as follows.

(1) *Frame-level pipelining*

The frame-level is a three-stage pipelining. The first stage consists of the edge detection component and the edge data buffering component, the second stage consists of the edge data splitting component, depth estimation component and depth data buffering component, and the third stage consists of the depth data splitting component. The schedule of frame-level pipelining is shown in Figure 10. When the first stage is processing on the frame *i*, the second stage is processing on the frame *i – 1* and the third stage is processing on the frame *i – 2*. To reduce the internal memory cost, the generated edge maps and depth maps in these three stages are all stored into external memory DDR instead of internal memory.

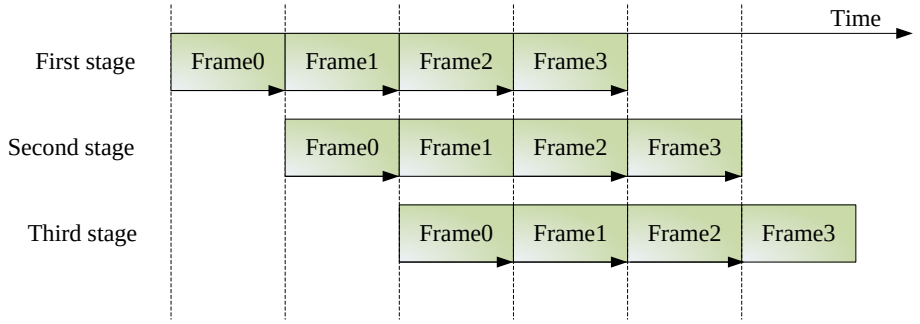


Figure 10. Schedule of Frame-Level Pipelining

(2) *Row-level pipelining and Column-level pipelining*

In the first stage of frame-level pipelining, row-level pipelining is adopted to improve the operating frequency and reduce memory cost [18, 19]. Figure 11 shows the detailed schedule of the proposed row-level pipelining. As shown in Figure 11, the schedule can be divided into two parts. First, edge data are calculated in the first part, which contains two stages. When the first stage is processing on the row *i + 1*, the second stage is processing on the row *i*. Then, the calculated edge data are stored into Block RAM in the Write stage of the second part. Finally, the data stored into Block RAM in the second part is read simultaneously, and the output data is structured as shown in Figure 5. Here, the sizes of Block RAM are all configured to be $W \times 8$bits, where W represents image width. For HD 720p video and HD 1080p video, W is set to 1280 and 1920 respectively.

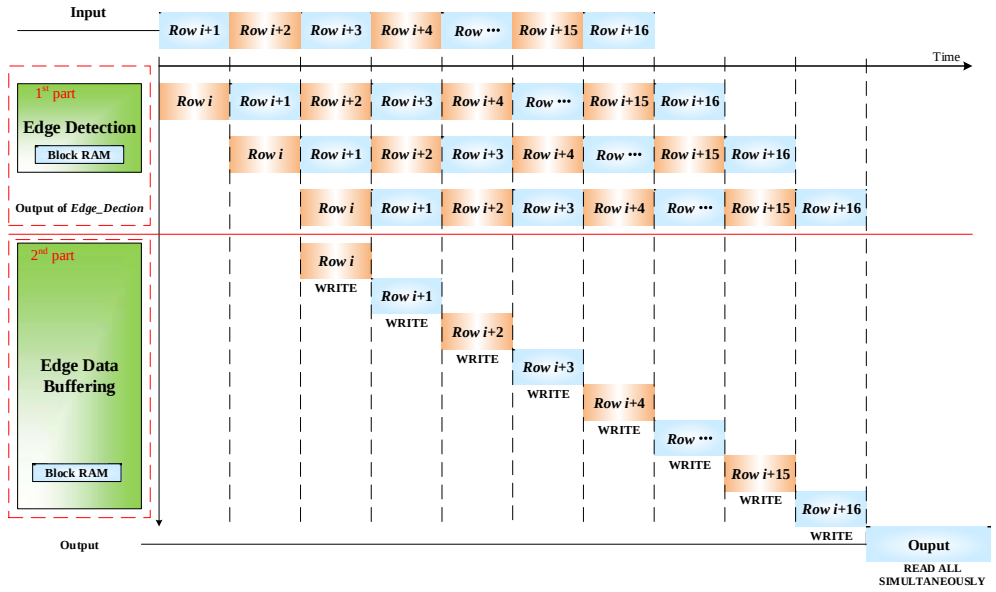


Figure 11. Schedule of Frame-Level Pipelining

Column-level pipelining is adopted in the second stage of frame-level pipelining, which consists of the depth estimation component and the depth data buffering component. The structure of column-level pipelining is similar to that of row-level pipelining, as shown in Figure 12. In this process, depth data are first estimated in the first part. Then, the estimated depth values are stored into Block RAM in Write stage of the second part. Finally, the output will be obtained by reading all Block RAM. The sizes for all Block RAM are set to $H\times$ 8bits, where H represents image height. For HD 720p video and HD 1080p video, H is set to 720 and 1080 respectively.

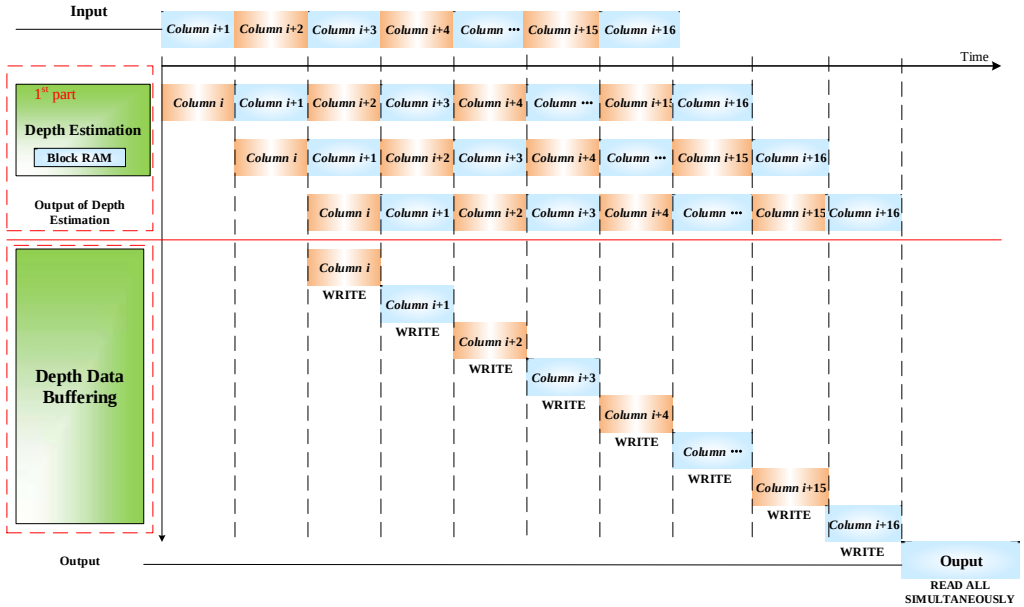


Figure 12. Schedule of Column-Level Pipelining

3.3. Row-To-Column Transposition

As mentioned in section 3.1, row-to-column transposition is essential to be performed, for the reason that depth maps are estimated column by column whereas the needed edge data are transmitted line by line. Figure 4 shows the architecture for this transposition. In this process, edge data are first buffered and restructured by the edge data buffering component. Then, the restructured data is stored into DDR by MPMC component. Finally, the data read out from DDR is transmitted to the edge data splitting component to undergo the split operation. After these procedures, edge data transmitted line by line are eventually changed to be transmitted column by column. The theory and framework of the edge data buffering component and the edge data splitting component have already been described in section 3.1. This section focuses on the write and read operations of DDR. The detailed description is presented as follows.

(1) *Write Operation of DDR*

After being restructured by the edge data buffering component, the reconstructed 128-bit data will be further stored into DDR by MPMC component. It is fairly easy to expect that edge data in different columns could be stored into different address ranges, which will be efficient to read them column by column. However, the data is stored into DDR burst by burst, not one by one. Each burst contains four 128-bit data, as shown in Figure 13. The write address will not be updated until the current burst is stored.

Based on the storage mechanism described above, the format that edge map stored in DDR is depicted in Figure 14, in which *A0*, *A1*, *etc* denote the base address of each address range. As shown in Figure 14, the data in four columns are contained in each address range. The four columns are not separated, but mixed with one another in a fixed order. As shown in Figure 11， for HD 720p video, 320 address ranges are needed to store one frame of the edge map and each address range includes 360 addresses, which can be calculated as 720 × 4 / 8 = 360. Here, 720, 4, 8 represent image height, column number and 64-bit (eight 8-bit data) per address in DDR, respectively. Similarly, for HD 1080p video, 480 address ranges are needed to store one frame of the edge map and each address range includes 540 addresses, which can be calculated as 1080 × 4 / 8 = 540.

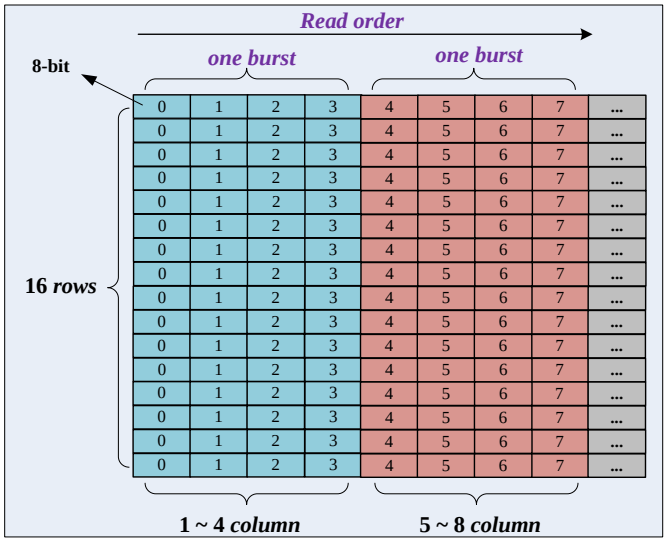


Figure 13. Illustration of Burst for Edge Maps. "0", "1", *etc* Represent the Column Number

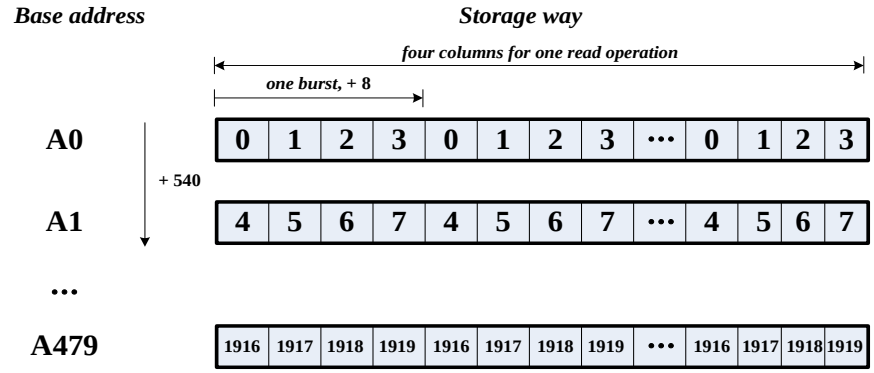


Figure 14. Format that Edge Map Stored In DDR for 720p Video. "0", "1", *etc* Represent the Column Number

(2) *Read Operation of DDR*

Similarly, the read operation is also performed burst by burst. In this process, the read address is increased progressively from the base address of one range to another. For example, the read address is increased progressively from *A0* to *A1*. Thus, all data between *A0* and *A1* will be read out. In other words, four columns will be read out from DDR at the same time, as shown in Figure 14. After the splitting operation, the edge maps will be converted to be transmitted column by column.

Note that the time interval between two adjacent read operations must be sufficient. This is because the depth estimation component will cost a relatively long time to cope with the data in four columns. Data miss will occur if the time interval is insufficient, since the previous four columns have not been dealt well with yet, but the new columns are being transmitted in.

3.4. Column-To-Row Transposition

Column-to-row transposition is also crucial to be performed, since depth maps are estimated column by column while they are required to be transmitted line by line for most applications. The architecture and the basic principle of column-to-row transposition are similar to those of row-to-column transposition, which has already been described in section 3.3. Here, only the difference in write operation of DDR is presented.

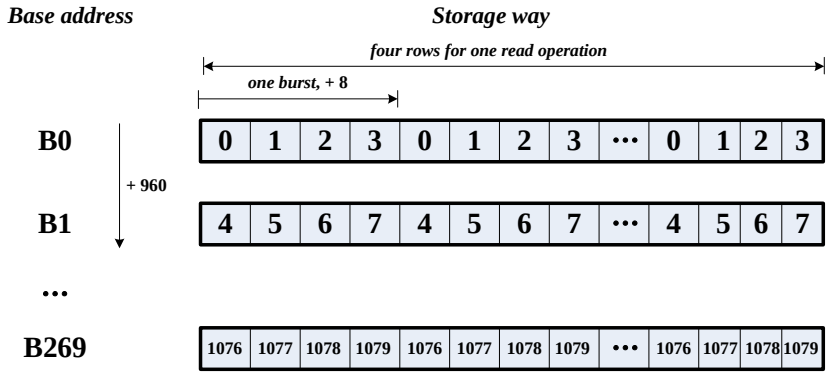


Figure 15. Format that Depth Map Stored in DDR for 720p Video. "0", "1", *etc* Represent the Row Number

Figure 15 depicts the format that depth map stored in DDR. The difference is that each address range contains different size addresses in column-to-row transposition. As shown in Figure 11, taking HD 720p video for example, each address range contains 640

addresses in column-to-row transposition, instead of 360 in row-to-column transposition. This is because in column-to-row transposition, four rows are stored in each address range. The interval can be calculated as 1280 × 4 / 8 = 640, where 1280, 4, 8 represent image width, row number and 64-bit (eight 8-bit data) per address, respectively. Similarly, for HD 1080p video, each address range contains 960 addresses in column-to-row transposition, which can be calculated as 1920 × 4 / 8 = 960, where 1920, 4, 8 represent image width, row number and 64-bit (eight 8-bit data) per address, respectively.

4. Experiment

In this paper, "*Ballet*", "*Breakdaners*" and other seven sequences are used for experiments [20], as shown in Figure 16. In this experiment, taking 720p video for example, the hardware-friendly algorithm was compared with other depth estimation algorithms. Meanwhile, two important performances of the hardware architecture, implementation performance and system simulation, were evaluated.

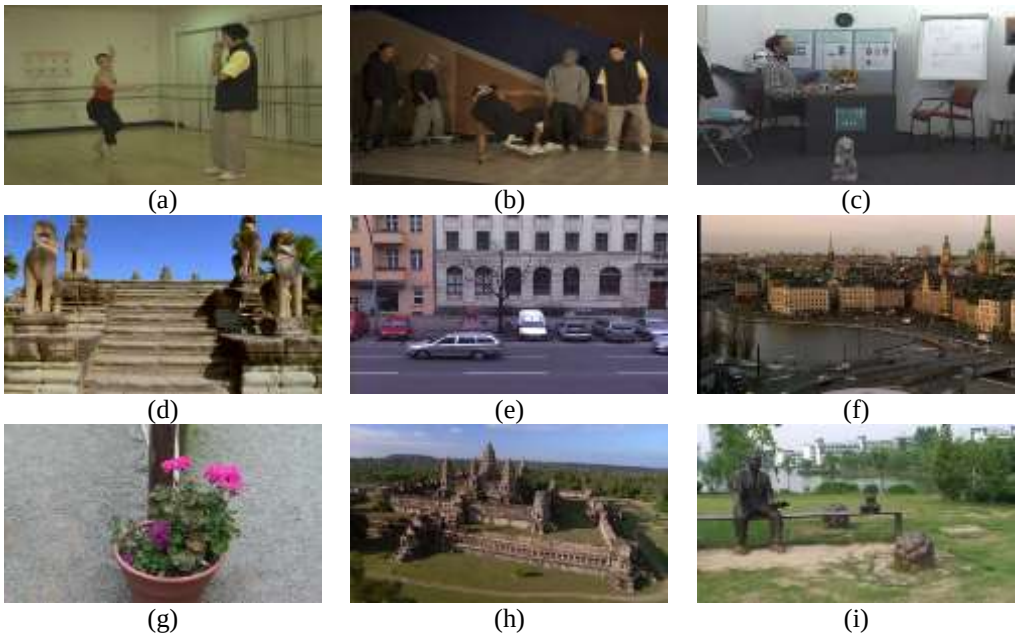
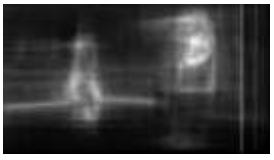
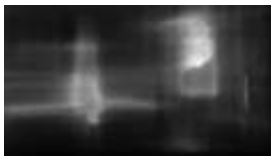
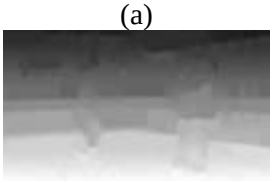
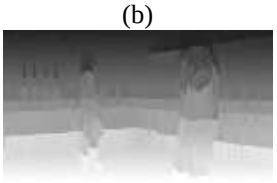


Figure 16. Test sequences.(a) Ballet; (b) Breakdancers; (c) Book Arrival; (d) Stair; (e) Outdoor;(f) House; (g) Flower; (h) AngkorWat; (i) Lawn

4.1. Comparison of Algorithms

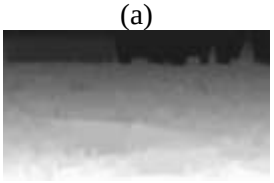
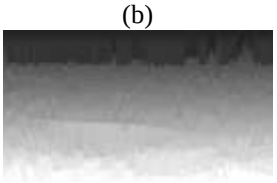
In this experiment, comparison of different depth estimation algorithms focuses on the assessment of both quality of estimated image and time complexity. We compare the proposed hardware-friendly algorithm with algorithms based on visual attention ("VA")[12], visual attention and joint bilateral filter ("VA + JBF")[12], relative height and joint bilateral filter("RH + JBF")[11], and in the comparison, the "VA + JBF" algorithm and the "RH + JBF" algorithm adopt the same joint bilateral filter and the mask sizes are both set to 19×19 in uniformity. The result of each algorithm is shown in Figure 17, and the corresponding running time is depicted in Figure 18(a). The detailed running time of each algorithm for Ballet is shown in Figure 18(b). As shown in Figure 17, and Figure 18, although the "VA + JBF" algorithm and "VA" algorithm have better performance than the hardware-friendly algorithm, they result in much higher time complexity. The "RH + JBF" algorithm and the hardware-friendly algorithm are both based on relative height depth cue. The depth map generated by "RH + JBF" algorithm is smoother than that by

hardware-friendly algorithm, but the "RH + JBF" algorithm costs much longer time. A large proportion of the running time is consumed on joint bilateral filter. Since the joint bilateral filter is complicated and time-consuming for hardware implementation, the joint bilateral filter is abandoned. Meanwhile, the equations with power and division operations are substituted by the approximate equations with addition and subtraction operations. Thus, the "RH + JBF" algorithm is converted to the proposed hardware-friendly algorithm in this paper. In conclusion, the hardware-friendly algorithm performs better in the trade-off between image quality and complexity in contrast to other algorithms. What's more, it is robust for scenery images.

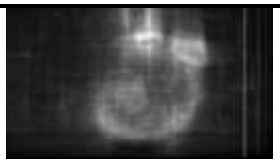



(a) (b)


(c) (d)

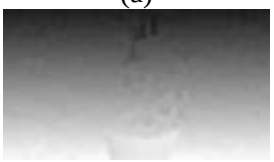
Ballet




(a) (b)


(c) (d)

House

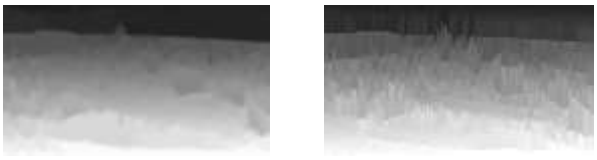
(a) (b)


(c) (d)

Flower



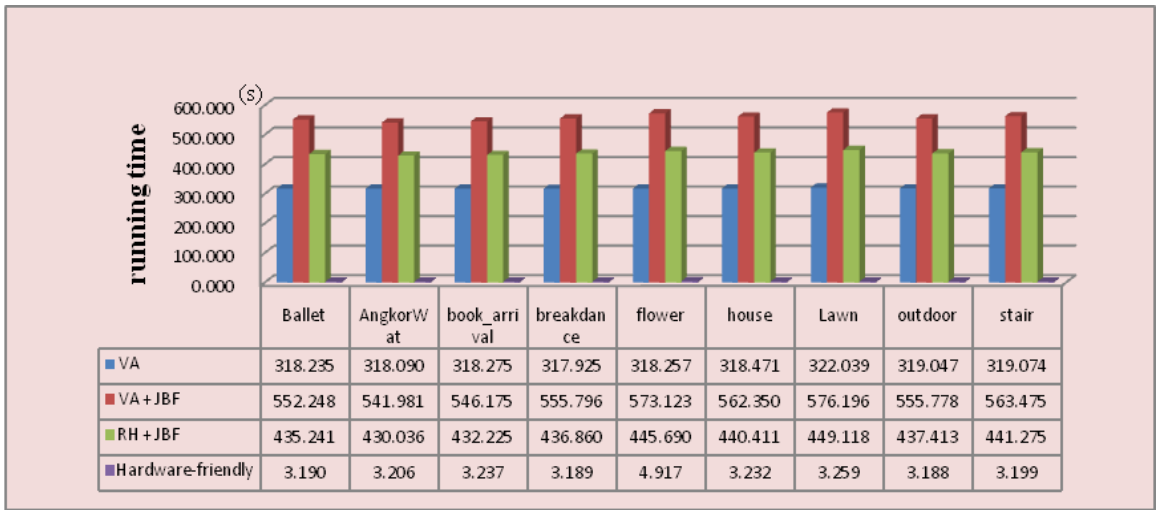

(a) (b)



(c) (d)

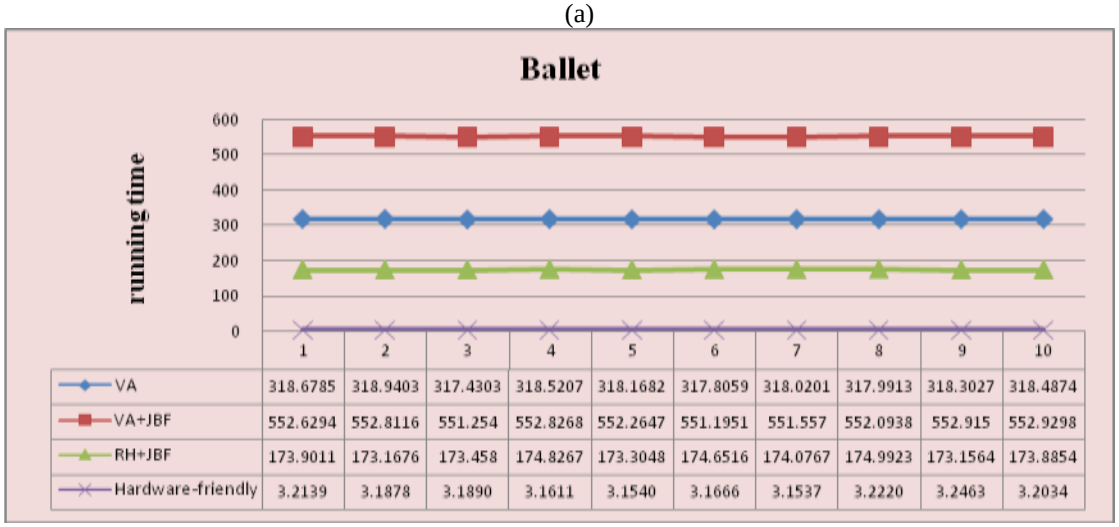
AngkorWat

Figure 17. Result of Each Algorithm. (a) VA; (b) VA + JBF; (c) RH + JBF; (d) Hardware-Friendly Algorithm



	Ballet	AngkorWat	book_arrival	breakdance	flower	house	Lawn	outdoor	stair
VA	318.235	318.090	318.275	317.925	318.257	318.471	322.039	319.047	319.074
VA+JBF	552.248	541.981	546.175	555.796	573.123	562.350	576.196	555.778	563.475
RH+JBF	435.241	430.036	432.225	436.860	445.690	440.411	449.118	437.413	441.275
Hardware-friendly	3.190	3.206	3.237	3.189	4.917	3.232	3.259	3.188	3.199

(a)



	1	2	3	4	5	6	7	8	9	10
VA	318.6785	318.9403	317.4303	318.5207	318.1682	317.8059	318.0201	317.9913	318.3027	318.4874
VA+JBF	552.6294	552.8116	551.254	552.8268	552.2647	551.1951	551.557	552.0938	552.915	552.9298
RH+JBF	173.9011	173.1676	173.458	174.8267	173.3048	174.6516	174.0767	174.9923	173.1564	173.8854
Hardware-friendly	3.2139	3.1878	3.1890	3.1611	3.1540	3.1666	3.1537	3.2220	3.2463	3.2034

(b)

Figure 18. Running Time. (a) Average Running Time of Each Algorithm; (b) Detailed Running Time of Sequence Ballet

4.2. Implementation Result

The proposed architecture has been implemented on Xilinx xc5vlx330t development board. As described above, internal memory resources are needed for all components, especially for the edge data buffering component and the depth data buffering component. The detailed internal memory cost is listed in Table 1. The Xilinx xc5vlx330t board can satisfy the internal memory requirement, for it has 11M-Byte internal memory resources.

Experimental results reveal that the proposed architecture can support real time processing of depth estimation. Taking the HD 720p video for example, Table 2 lists the performance analyzed by Xilinx ISE. In average, the proposed architecture can achieve throughput of 52.73M pixels/s, *i.e.*, 60 f / s for the HD 720p video. The other logic consumptions are listed in Table 3.

Table 1. Block RAM Cost of Each Component for 720p video

Component	Size of Block RAM	Number of Block RAM	Total cost
Edge Detection	1280 × 8-bit	4	40K-bit
Edge Data Buffering	1280 × 8-bit	16	160K-bit
Edge Data Splitting	720 × 8-bit	4	22.5K-bit
Depth Estimation	720 × 8-bit	1	5.625K-bit
Depth Data Buffering	720 × 8-bit	16	90K-bit
Depth Data Splitting	1280 × 8-bit	4	40K-bit
Total			358.125-Kbit = 44.8kBytes

Table 2. Performance of the Implementation for 720p video

Clock Frequency	200MHz
External Bus(bit)	64
Bandwidth	210.94M
Internal Memory	44.8kBytes
External Memory	3.52MBytes
Frame Size	1280 × 720
Frame Rate(f/s)	60
Throughput(pixels/s)	52.73M

Table 3. Consumptions of Logic Sources for 720p Video

Logic Sources Type	Consumptions	Total	Usage Rate
Number of Slice Registers	13858	207360	6%
Number of Slice LUTs	12178	207360	5%
Number of fully used LUT-FF pairs	8891	17145	51%
Number of bonded IOBS	186	960	19%
Number of Block RAM/FIFO	57	324	17%

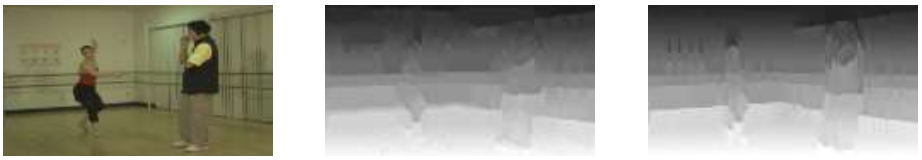
Number of BUFG/BUFGCTRLs	6	32	18%

4.3. System Simulation

As shown in Table 4, compared to depth maps estimated by the "RH + JBF" algorithm without joint bilateral filter by software, the average PSNR of ten frames is 29.7456, which means that depth maps estimated by the proposed hardware architecture are similar to those estimated by software. The degradation locates in an acceptable range and can be neglected. The corresponding simulation results are depicted in Figure 19.

Table 4. Average PSNR of Ten Frames

Test Sequences	Camera Number	Frame Number	PSNR
Ballet	0	0-9	29.9347
Breakdancers	0	0-9	28.5237
Book Arrival	8	10-19	29.5256
Stair	0	10-19	30.0529
Outdoor	0	0-9	29.5538
House	1	0-9	29.0307
Flower	0	30-39	31.8467
AngkorWat	0	0-9	31.3123
Lawn	0	0-9	27.9303
Average			29.7456



Ballet

House

Flower

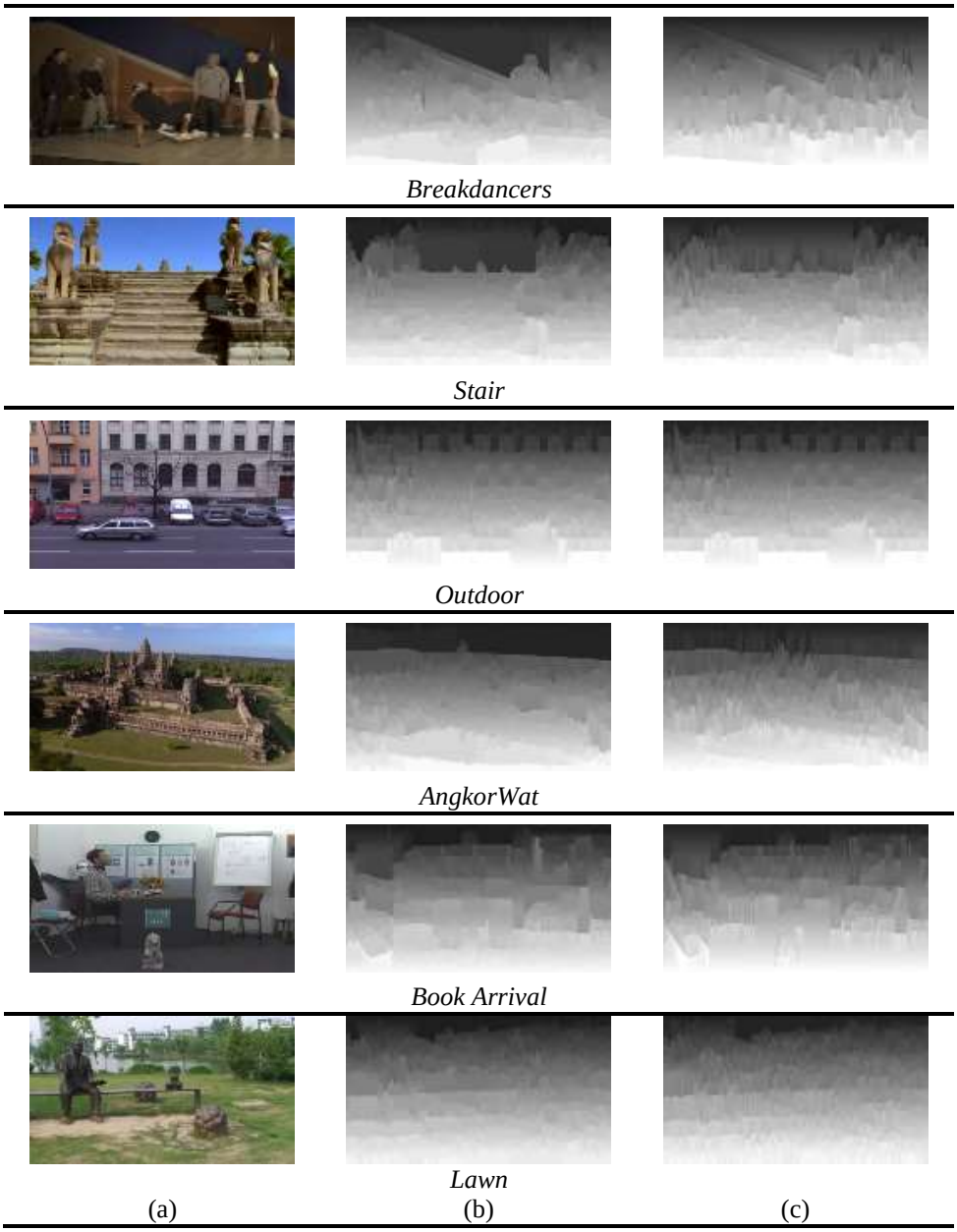


Figure 19. Simulation Results. (a) Color Image; (b) Depth Maps Estimated by Software; (c) Depth Maps Estimated by Hardware

5. Conclusions

Depth map has been widely used in 2D-to-3D conversion system for its rich depth information of objects. The accuracy of depth map will strongly affect the quality of synthesized 3D views. However, due to high complexity of depth estimation algorithms, software implementation cannot meet the requirement of real-time processing. To solve the problems, this paper presents a real-time depth estimation system based on the proposed hardware-friendly algorithm and its efficient implementation. In the hardware-friendly algorithm, the equations with power and division operations are substituted by the approximate equations with addition and subtraction operations to reduce the computational complexity. In the efficient implementation, considering that the proposed algorithm is a column-based algorithm, in this paper, row-to-column transposition is

proposed to resolve the conflict. Meanwhile, pipelining technology is adopted to reduce the memory cost, as well as improve the operating frequency. External memory DDR is also used to reduce the internal memory cost. The final implementation reveals that the proposed system can perform reasonably good real-time depth estimation for the scenery videos. For other scenes of videos, the accuracy of depth estimation is degraded. How to improve the algorithm and the hardware architecture to suit all scenes will be our future work.

Acknowledgements

This work was jointly supported by the Graduate Student Research Innovation Project of Chongqing (No. CYS14021), the National Natural Science Foundation of China (No. 61201347), and the Fundamental Research Funds for the Central Universities (No. CDJZR13185502).

References

[1] S Choi, B Ham and K Sohn, "Space-Time Hole Filling With Random Walks in View Extrapolation for 3D Video", Ieee Transactions on Image Processing, vol. 22, no. 6, **(2013)**, pp. 2429-2441.
[2] Y. Shao-Jun, W. Liang-Hao, L. Dong-Xiao and Z. Ming, "A Real-Time Full HD 2D-to-3D Video Conversion System Based on FPGA", in Image and Graphics (ICIG), 2013 Seventh International Conference on, **(2013)**, pp. 774-778.
[3] M. Koppel, M. Ben Makhlouf and P. Ndjiki-Nya, "Optimized adaptive depth map filtering", in Image Processing (ICIP), 2013 20th IEEE International Conference on, **(2013)**, pp. 1356-1360.
[4] R. Liu, H. Xie, F. Tian, Y. Wu, G. Tai, Y. Tan, W. Tan, H. Chen and L. Ge, "Hole-filling Based on Disparity Map for DIBR", KSII Transactions on Internet and Information Systems (TIIS), vol. 6, no. 10, **(2012)**, pp. 2663-2678.
[5] M. Schmeing and X. Jiang, "Time-consistency of disocclusion filling algorithms in Depth Image Based Rendering", in 5th 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video, 3DTV-CON 2011, May 16, 2011 - May 18, 2011. 2011. Antalya, Turkey: IEEE Computer Society, pp. Bilkent University; Grundig; IEEE; MPEG-IF.
[6] J. Jae-Il and H. Yo-Sung, "Depth map estimation from single-view image using object classification based on Bayesian learning", in 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON), 2010, **(2010)**, pp. 1-4.
[7] Z. Liang, C. Vazquez and S. Knorr, "3D-TV Content Creation: Automatic 2D-to-3D Video Conversion", IEEE Transactions on Broadcasting, **(2011)**, vol. 57, no. 2, pp. 372-383.
[8] X. Xu, L.-M Po, C.-H. Cheung, L. Feng, K.-H. Ng and K.-W. Cheung, Ieee, "Depth-aided Exemplar-based Hole Filling for DIBR View Synthesis", in 2013 Ieee International Symposium on Circuits and Systems, **(2013)**, pp. 2840-2843.
[9] S. Li, F. Wang and W. Liu, "The overview of 2D to 3D conversion system", in Computer-Aided Industrial Design & Conceptual Design (CAIDCD), 2010 IEEE 11th International Conference on, **(2010)**, pp. 1388-1392.
[10] S. M. Muddala, M. Sjostrom and R. Olsson, "Depth-based inpainting for disocclusion filling", in 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video, 3DTV-CON 2014, July 2, 2014 - July 4, 2014. Budapest, Hungary: IEEE Computer Society, **(2014)**.
[11] Y. J. Jung, A. Baik, J. Kim and D. Park, "A novel 2D-to-3D conversion technique based on relative height depth cue", in Stereoscopic Displays and Applications XX, January 19, 2009 - January 21, 2009. San Jose, CA, United states: SPIE, **(2009)**.
[12] J. Kim, A. Baik, Y. J. Jung and D. Park, "2D-to-3D conversion by using visual attention analysis", in Stereoscopic Displays and Applications XXI, January 18, 2010 - January 20, 2010, San Jose, CA, United states: SPIE, pp. The Society for Imaging Science and Technology (IS and T); The Society of Photo-Optical Instrumentation Engineers (SPIE), **(2010)**.
[13] L.-M. Po, X. Xu, Y. Zhu, S. Zhang, K.-W. Cheung and C.-W. Ting, "Automatic 2D-To-3D video conversion technique based on depth-from-motion and color segmentation", in 2010 IEEE 10th International Conference on Signal Processing, ICSP2010, October 24, 2010 - October 28, 2010. Beijing, China: Institute of Electrical and Electronics Engineers Inc., **(2010)**, pp. 1000-1003.
[14] [14]H.-J. Li, G.-W. Teng, Z.-Y. Zhang, P. An, R. Ma, J.-W. Wang and F.-Q. Wu, "Hardware solution of real-time depth estimation based on stereo vision", in Audio, Language and Image Processing (ICALIP), 2012 International Conference on, **(2012)**, pp. 39-44.
[15] Z. Qiuwen, A. Ping, Z. Yan, S. Liquan and Z. Zhaoyang, "An improved depth map estimation for coding and view synthesis", in Image Processing (ICIP), 2011 18th IEEE International Conference on., **(2011)**, pp. 2101-2104.

[16] X. Wang, X. Jin, X. Xu and M. Zhang, "Joint bilateral image interpolation", Journal of Image and Graphics, vol. 16, no. 12, **(2011)**, pp. 2117-2123.
[17] T. Sung-Fang, C. Chao-Chung, L. Chung-Te and C. Liang-Gee, "A real-time 1080p 2D-to-3D video conversion system", in Consumer Electronics (ICCE), 2011 IEEE International Conference on., **(2011)**, pp. 803-804.
[18] W.-Y. Chen, Y.-L. Chang, H.-K. Chiu, S.-Y. Chien and L.-G. Chen, "Real-time depth image based rendering hardware accelerator for advanced three dimensional television system", in 2006 IEEE International Conference on Multimedia and Expo, ICME 2006, July 9, 2006 - July 12, 2006. Toronto, ON, Canada: Inst. of Elec. and Elec. Eng. Computer Society, **(2006)**, pp. 2069-2072.
[19] P.-C. Lin, P.-K. Tsung and L.-G. Chen, "Low-cost hardware architecture design for 3D warping engine in multiview video applications", in 2010 IEEE International Symposium on Circuits and Systems: Nano-Bio Circuit Fabrics and Systems, ISCAS 2010, May 30, 2010 - June 2, 2010. Paris, France: IEEE Computer Society, **(2010)**, pp. 2964-2967.
[20] G. Zhang, J. Jia, T.-T. Wong and H. Bao, "Consistent depth maps recovery from a video sequence", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 6, **(2009)**, pp. 974-988.

Authors

Ran Liu, he received his BE, ME, and DE degrees in computer science from Chongqing University, Chongqing, China, in 2001, 2004, and 2007, respectively. He worked as a postdoctoral researcher for Homwee Technology Co., Ltd., Chengdu, China, from 2008 to 2010. He is now an associate professor at the College of Communication Engineering and the College of Computer Science, Chongqing University, China. His research interests include 3D imaging, medical image processing, and medical imaging

Donghua Cao, she is currently pursuing her master's degree at the Key Laboratory of Dependable Service Computing in Cyber Physical Society of the Ministry of Education, Chongqing University, Chongqing, China. She received her bachelor's degree from Qufu Normal University, Shandong, China, in 2013. Her research interests include 3D TV and stereo image processing.

Zekun Deng, she is currently pursuing her master's degree at the College of Communication Engineering, Chongqing University, Chongqing, China. She received her bachelor's degree from Chongqing University, Chongqing, China, in 2014. Her research interests include 3D TV and medical image processing.

Zhenwei Huang, he is currently pursuing his master's degree at the College of Communication Engineering, Chongqing University, Chongqing, China. He received his bachelor's degree from Chongqing University, Chongqing, China, in 2012. His research interests include 3D TV and stereo image processing.

Miao Xu, she is currently pursuing her master's degree at the Key Laboratory of Dependable Service Computing in Cyber Physical Society of the Ministry of Education, Chongqing University, Chongqing, China. She received her bachelor's degree from Shanxi Normal University, Shanxi, China, in 2014. Her research interests include 3D TV and stereo image processing.

Ruishuang Jia, she is currently pursuing her master's degree at the Key Laboratory of Dependable Service Computing in Cyber Physical Society of the Ministry of Education, Chongqing University, Chongqing, China. She received her bachelor's degree from Northeast University at Qinhuangdao, Hebei, China, in 2014. Her research interests include 3D TV and IC design.

Dehao Li, he is currently pursuing his master's degree at the Key Laboratory of Dependable Service Computing in Cyber Physical Society of the Ministry of Education, Chongqing University, Chongqing, China. He received his bachelor's degree from Hubei University for Nationalities, Hubei, China, in 2014. His research interests include 3D TV and stereo image processing.

Mingming Liu, he is currently pursuing his master's degree at the Key Laboratory of Dependable Service Computing in Cyber Physical Society of the Ministry of Education, Chongqing University, Chongqing, China. He received his bachelor's degree from Xidian University, Shanxi, China, in 2014. His research interests include 3D TV and stereo image processing.

