Open access • Journal Article • DOI:10.1109/TCSVT.2011.2148410

VLSI Architecture for Real-Time HD1080p View Synthesis Engine — Source link

Ying-Rung Horng, Yu-Cheng Tseng, Tian-Sheuan Chang

Institutions: MediaTek, National Chiao Tung University

Published on: 29 Apr 2011 - IEEE Transactions on Circuits and Systems for Video Technology (IEEE)

Topics: View synthesis, Algorithm design, Bilinear interpolation, High memory and Very-large-scale integration

VLSI Architecture for Real-Time HD1080p View Synthesis Engine

Ying-Rung Horng, Yu-Cheng Tseng, *Student Member, IEEE,* and Tian-Sheuan Chang, *Senior Member, IEEE*

***Abstract*—This paper presents a real-time HD1080p view synthesis engine based on the reference algorithm from 3-D video coding team by solving high computational complexity and high memory cost problems. For the computational complexity, we propose the bilinear interpolation to simplify the hole filling process, and the Z scaling method with floating-point format to reduce the cost of homography calculation. For the memory cost, we propose the frame-level pipelining to reduce the requirement of warped depth maps, and the column-order warping method to remove the Z-buffer in occlusion handling. With the 90 nm complementary metal-oxide-semiconductor technology, our view synthesis engine can archive the throughput of 32.4 f/s for HD1080p videos with the gate count of 268.5 K and the internal memory of 69.4 kbytes. The experimental result shows our implementation has the similar synthesis quality as the original reference algorithm.**

***Index Terms*—3-D video coding, view synthesis, VLSI design.**

I. Introduction

With fast development and popularity of 3-D displays and 3-D television (TV) systems, view synthesis engine becomes one of the most important components to synthesize single or multiple virtual-view videos in the stereoscopic TV or the free-viewpoint television [1], [2].

A common approach for view synthesis engine is the depth-image-based rendering (DIBR) algorithm [3]–[9]. The DIBR algorithm is based on the 3-D warping, which warps a video to another view according to depth maps. With single-view input, the DIBR algorithm suffers from large occluded holes in synthesized view. The hole size can be decreased by the depth smoothing methods [4]–[7], and then the holes can be filled by the interpolation method with depth and gradient information [5]. On the contrary, the DIBR algorithm with multiview inputs has smaller native holes because they could be recovered by other views. Thus, the remained holes are easier to be patched.

Manuscript received November 16, 2010; revised February 24, 2011; accepted April 10, 2011. Date of publication April 29, 2011; date of current version September 2, 2011. This work was supported by the National Science Council of Taiwan, under Grant NSC-99-2221-E-009-0189. This paper was recommended by Associate Editor T. Fujii.

Y.-R. Horng is with MediaTek, Inc., Hsinchu 30078, Taiwan (e-mail: yingrung.horng@mediatek.com).

Y.-C. Tseng and T.-S. Chang are with the Department of Electronics Engineering and Institute of Electronics, National Chiao-Tung University, Hsinchu 30010, Taiwan (e-mail: tyucheng@dragons.ee.nctu.edu.tw; tschang@dragons.ee.nctu.edu.tw).

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

Digital Object Identifier 10.1109/TCSVT.2011.2148410

The DIBR algorithms with multiview inputs could be classified into the one-step warping and the two-step warping as depicted in Fig. 1. The one-step warping directly warps the reference textures to the virtual texture according to the reference depth maps, while the two-step warping first warps the reference depth maps to the virtual depth map, and then uses it to generate the virtual texture. Rogmans *et al.* [10] and Morvan [11] show that the two-step warping could perform better because the sampling precision is higher in the virtual view. Moreover, the round-off sampling errors could be further eliminated by filtering the virtual depth map [12].

Based on the two-step warping, the 3-D video coding team of the ISO/IEC MPEG developed the high-quality view synthesis reference software (VSRS) algorithm [13], which consists of preprocessing, depth and texture warping, and hole filling. The VSRS algorithm can support the view synthesis for any arbitrary target view using the basic inputs of two reference depth maps and videos captured by misaligned cameras. This function can be directly extended to support the multiview synthesis. However, for the high definition (HD) video, such as 1280×720 (HD720p) or 1920×1080 (HD1080p), the complexity of VSRS algorithm is also dramatically increased. Therefore, to satisfy the demands of real-time processing of HD video, a very large scale integration (VLSI) hardware implementation is necessary.

In previous research, Fukushima *et al.* [14] developed a free-viewpoint rendering system using ray-space by central processing unit programming to reach the throughput of 12 f/s for 420×320 video. By graphics processing unit programming, Zitnick *et al.* [15] implemented a high-quality software renderer to support the processing of 5 f/s for 1024×768 video, and Rogmans *et al.* [10], [16] implemented a stereo-based view synthesis system to achieve more than 50 f/s for 450×375 video. In VLSI implementation, Chen *et al.* [17] proposed a DIBR hardware accelerator to reach the throughput of 25 f/s for 720×576 stereoscopic video.

However, the VSRS algorithm suffers from the high computational complexity and high memory cost in each step. The previous paper [18] has tried to reduce the memory cost of homography matrices by the linear interpolation approximation (LIA) method. But the following problems still need to be addressed. First, the cost of operators is increased significantly due to fractional and large bit-width numbers in the preprocessing step. Second, a large memory is demanded to store temporarily warped depth maps due to the camera rotation in the depth warping step. Third, a large Z-buffer

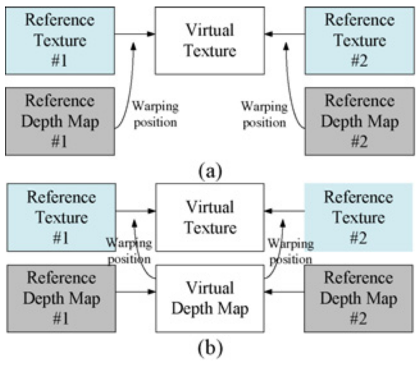


Fig. 1. DIBR algorithm. (a) With one-step warping. (b) With two-step warping.

is also needed to store current front depths for occlusion handling. Finally, the complicated hole filling step has to be iteratively processed on the whole frame by inpainting method.

To solve the above problems, we propose hardware-efficient algorithms with a hierarchical pipelining architecture. In the hardware-efficient algorithms, we propose a simple bilinear interpolation to replace the complicated hole filling, a Z scaling method to reduce the cost of operators, and the column-order warping method to remove the Z-buffer. The hierarchical pipelining adopts the frame-level pipelining to reduce the memory cost of warped depth maps, and then the column-level pipelining with a data packing method to increase the external access efficiency. The final implementation can support the real-time processing of HD1080p video and has similar quality as the original VSRS.

The rest of this paper is organized as follows. First, Section II describes the VSRS algorithm and its design challenges. Then, Section III proposes the hardware-efficient algorithms, and Section IV presents our hierarchical pipelining architecture. With the proposed algorithms and architecture, Section V demonstrates our implementation result and synthesis performance. Finally, Section VI concludes this paper.

II. Analysis of VSRS

A. VSRS Software

Fig. 2 shows the VSRS algorithm flow that consists of the preprocessing, depth forward warping, texture reverse warping, and hole filling. The first three steps are separately applied to the left-view and right-view, and the last one is applied only to the virtual-view. The detail of each step is described as follows.

1) *Preprocessing:* The preprocessing loads the camera parameters to calculate the homography matrices for the depth forward warping and the texture reverse warping. The homograph matrix $\boldsymbol{H}$ is a 3-by-3 matrix formed by

$$H = \begin{bmatrix} h_{00} & h_{10} & h_{20} \\ h_{01} & h_{11} & h_{21} \\ h_{02} & h_{12} & h_{22} \end{bmatrix}. \quad (1)$$

In the next warping processes, it can be used to find the warped position between views by

$$\begin{bmatrix} u_{dst} \\ v_{dst} \\ 1 \end{bmatrix} = H(d) \begin{bmatrix} u_{src} \\ v_{src} \\ 1 \end{bmatrix} \quad (2)$$

where (u_{src}, v_{src}) is in the source view, and (u_{dst}, v_{dst}) is the warped position in the destination view. Note that different depth levels need different homography matrices as shown in Fig. 3. Thus, the homography matrix in (2) has the argument of d, which is the depth level of the source position (u_{src}, v_{src}).

The warping direction in the VSRS algorithm includes the left-to-virtual and right-to-virtual for the depth forward warping, and their inverse directions for the texture reverse warping as shown in Fig. 4. Therefore, the associated homography matrices $\boldsymbol{H_{LV}}$, $\boldsymbol{H_{VL}}$, $\boldsymbol{H_{RV}}$, and $\boldsymbol{H_{VR}}$ need to be calculated. The following steps derive the homography matrices, $\boldsymbol{H_{LV}}$ and $\boldsymbol{H_{VL}}$, which can be applied to $\boldsymbol{H_{RV}}$ and $\boldsymbol{H_{VR}}$ as well.

First, the Z transform converts the depth level to the depth value by

$$Z = 1 \Big/ \left(\frac{d}{255} \left(\frac{1}{Z_{\min}} - \frac{1}{Z_{\max}} \right) + \frac{1}{Z_{\max}} \right) \quad (3)$$

where d is the depth level in the depth map and Z is the depth value. In addition, Z_{min} and Z_{max} are the minimum and

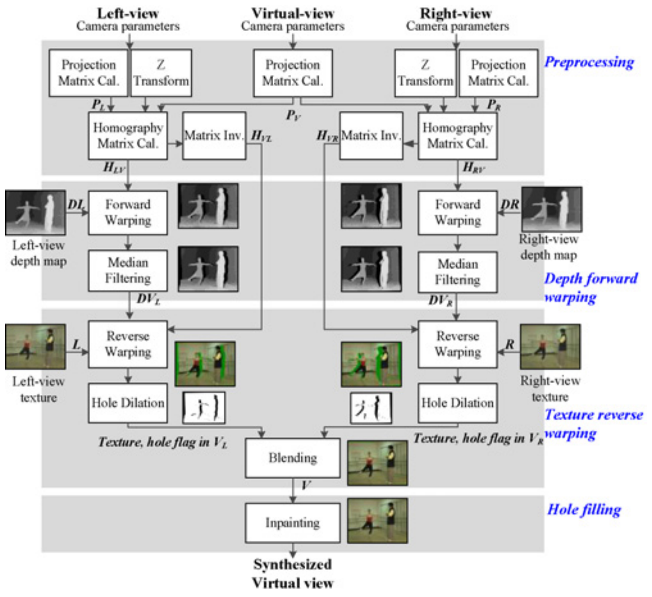


Fig. 2. Algorithm flow of VSRS.

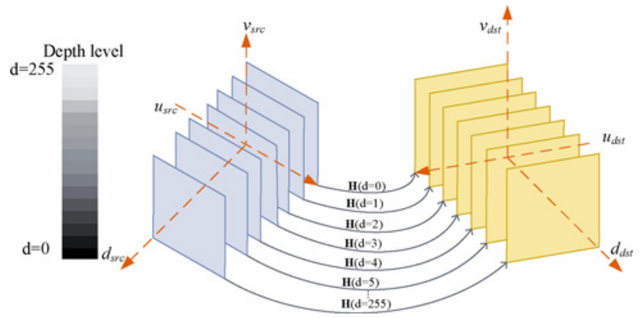


Fig. 3. Homography matrices for different depth levels.

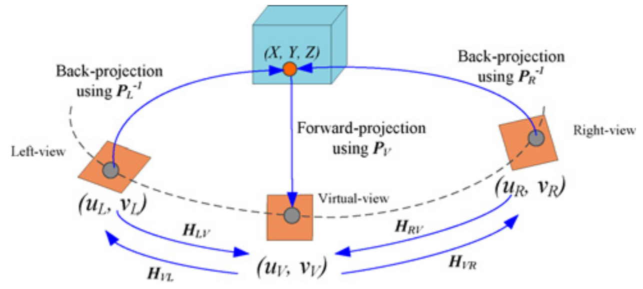


Fig. 4. Concept of 3-D warping.

maximum depth values in a scene. With the depth value Z, a pair of pixels between views can be calculated by

$$\begin{pmatrix} u_V & v_V & 1 \end{pmatrix}^T = \frac{s_L}{s_V} P_V P_L^{-1} \begin{pmatrix} u_L & v_L & 1 \end{pmatrix}^T \tag{4}$$

where $\boldsymbol{P_V}$ and $\boldsymbol{P_L}$ are projection matrices for projecting the object point (X, Y, Z) to the pixels (u_V, v_V) and (u_L, v_L), respectively. The terms of s_L and s_V are the scale factors that are related to the depth value Z and the translation element T_z.

The projection matrix calculation finds four pairs of pixels between the left-view and the virtual-view by (4). Then, these four pairs can be substituted into (2) to form an 8×8 linear system as follows:

$$\begin{bmatrix} u_{L,1} & v_{L,1} & 1 & 0 & 0 & 0 & -u_{V,1}u_{L,1} & -u_{V,1}v_{L,1} \\ 0 & 0 & 0 & u_{L,1} & v_{L,1} & 1 & -v_{V,1}u_{L,1} & -v_{V,1}v_{L,1} \\ u_{L,2} & v_{L,2} & 1 & 0 & 0 & 0 & -u_{V,2}u_{L,2} & -u_{V,2}v_{L,2} \\ 0 & 0 & 0 & u_{L,2} & v_{L,2} & 1 & -v_{V,2}u_{L,2} & -v_{V,2}v_{L,2} \\ u_{L,3} & v_{L,3} & 1 & 0 & 0 & 0 & -u_{V,3}u_{L,3} & -u_{V,3}v_{L,3} \\ 0 & 0 & 0 & u_{L,3} & v_{L,3} & 1 & -v_{V,3}u_{L,3} & -v_{V,3}v_{L,3} \\ u_{L,4} & v_{L,4} & 1 & 0 & 0 & 0 & -u_{V,4}u_{L,4} & -u_{V,4}v_{L,4} \\ 0 & 0 & 0 & u_{L,4} & v_{L,4} & 1 & -v_{V,4}u_{L,4} & -v_{V,4}v_{L,4} \end{bmatrix}$$
$$= \begin{bmatrix} u_{V,1} \\ v_{V,1} \\ u_{V,2} \\ v_{V,2} \\ u_{V,3} \\ v_{V,3} \\ u_{V,4} \\ v_{V,4} \end{bmatrix} \tag{5}$$

where h_{22} is equal to 1. To solve the linear system for $\boldsymbol{H_{LV}}$, the homography matrix calculation in the VSRS algorithm adopts the function cvFindHomography in the OpenCV library [19]. Finally, the inverse matrix of $\boldsymbol{H_{LV}}$ is calculated as $\boldsymbol{H_{VL}}$. The above steps need to be performed for each depth level, and 256 homography matrices for each warping direction are calculated.

2) *Depth Forward Warping:* For the depth forward warping in Fig. 2, it warps the input depth maps DL and DR, respectively, to the warped depth maps DV_L and DV_R in the virtual-view. In this process, the warped position (u_V, v_V) in the virtual-view DV_L can be acquired using (2) for each pixel (u_L, v_L) in DL. Then the depth value d at (u_L, v_L) is copied to DV_L. After pixel-by-pixel warping, the whole warped depth map DV_L is synthesized. The same steps are also used to warp DR to DV_R.

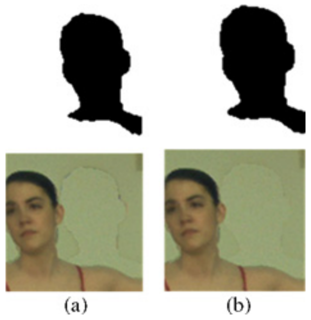


Fig. 5. Blended images. (a) Without hole dilation. (b) With hole dilation.

Since the depth maps DL and DR may have noise and the warping process may induce sampling alias, the warped depth maps DV_L and DV_R usually suffer from small noisy holes. Therefore, a 3×3 median filtering is applied to remove them [12], [20], [21].

3) *Texture Reverse Warping:* For the texture reverse warping in Fig. 2, it warps the reference textures L and R, respectively, to the virtual textures V_L and V_R according to the warped depth maps DV_L and DV_R. This process computes the warped position (u_L, v_L) in L for each pixel (u_V, v_V) in V_L, and then copies the reference texture in L to V_L to synthesize the texture of virtual-view V_L. In addition, the corresponding hole map is labeled for the positions without any texture. The same steps are also applied to warp R to V_R.

Then, the marked hole regions in the hole map are expanded by the hole dilation to avoid the synthesis artifacts as shown in Fig. 5. With the original and dilated hole maps, the two warped textures V_L and V_R are combined into a new texture V by a blending process according to the truth Table in Table I. In this table, the holes in the cases 1 and 2 are visible by two reference-views, and thus can be filled by a linear interpolation with the factor α associated with camera translation vector. In cases 3 to 6, those holes are only visible by one view, and can only be filled by the non-hole pixel $V_L(u, v)$ or $V_R(u, v)$. In the last case 7, this position is invisible from any view, and thus is handled by the successive hole filling process.

4) *Hole Filling:* For the remaining holes, the VSRS algorithm provides the advanced inpainting method [22] in the 3-D warping mode. The advanced inpainting method is performed on the whole frame iteratively to diffuse the non-hole pixels to holes. Then, the final virtual-view texture V is synthesized completely.

B. Straightforward Architecture

Fig. 6 shows a straightforward hardware architecture for the VSRS algorithm. In which, the two reference-view textures L, R, and their depth maps DL, DR are stored in the external memory [i.e., dynamic random access memory (DRAM)]. Corresponding to the VSRS algorithm flow in Fig. 2, the preprocessing first calculates the homography matrices $\boldsymbol{H}$ for the forward warping and the reverse warping. To save execution time, this preprocessing is only needed when the camera configuration is changed. Then, the forward warping

TABLE I
TRUTH TABLE FOR THE BLENDING PROCESS

Case	Hole Map Before Dilation		Hole Map After Dilation		Value for $V(u, v)$
	$V_L(u, v)$	$V_R(u, v)$	$V_L(u, v)$	$V_R(u, v)$	
1	1	1	1	1	$(1-\alpha)\ V_L(u, v)+\alpha V_R(u, v)$
2	1	1	0	0	$(1-\alpha)V_L(u, v)+\alpha V_R(u, v)$
3	1	1	1	0	$V_L(u, v)$
4	1	1	0	1	$V_R(u, v)$
5	1	0	x	x	$V_L(u, v)$
6	0	1	x	x	$V_R(u, v)$
7	0	0	x	x	0

0 = hole, 1 = non-hole, x = do not care

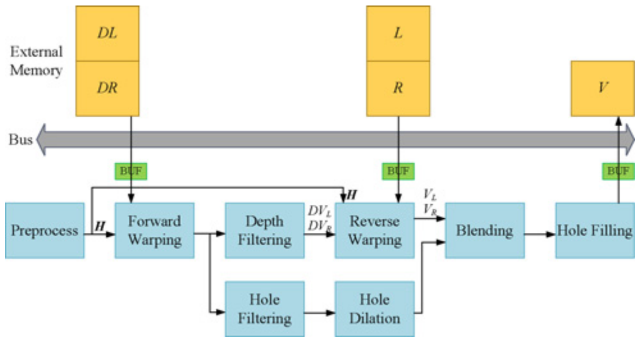


Fig. 6. Straightforward architecture.

TABLE II
Z SCALING METHOD FOR HIGH DYNAMIC RANGE PARAMETERS

	Before Z Scaling		Scalar Factor	After Z Scaling	
Sequence	Z_{max}	Z_{min}		Z_{max}	Z_{min}
Ballet	42	130	1	42	130
Breakdancers	44	120	1	44	120
BookArrival	23.345	54.471	1	23.345	54.471
Lovebird1	1560.122	156012.2	1/1024	1.523	152.355
Akko&Kayo	2342.249	12491.99	1/64	36.597	195.187
Newspaper	2715.182	9050.605	1/64	42.424	141.415
ChampagneTower	2281.358	7045.261	1/32	71.292	220.164
Kendo	448.2512	11206.28	1/64	7.003	175.098

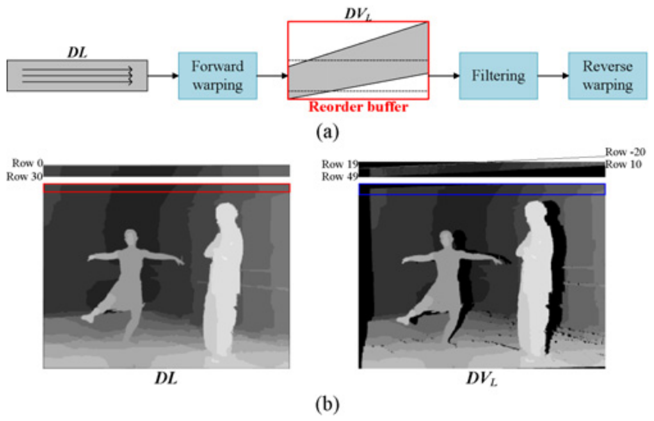


Fig. 7. Depth map warping from DL to DV_L. (a) Reorder buffer for DV_L. (b) Example in the sequence *Ballet* (1024 × 768).

generates the warped depth maps DV_L and DV_R, and the depth filtering applies a 3 × 3 median filtering to reduce their noise. According to DV_L and DV_R, the reverse warping renders the two virtual-view textures V_L and V_R using the reference textures L and R. At the same time, the corresponding hole maps are generated and dilated for the blending. With the warped textures and the hole maps, the blending process combines V_L and V_R to synthesize a new texture V. Finally, the hole filling adopts the inpainting technique to fill the remaining holes, and writes the synthesized result V to the external memory.

C. Design Challenges

In the above straightforward architecture, the implementation of view synthesis engine suffers from the following design challenges.

1) *High Computational Complexity in Preprocessing:* The preprocessing has to perform matrix multiplication and solve an 8-by-8 linear system, which requires adders, multipliers, and dividers. Moreover, these computations are in fractions, and all values have high dynamic range, as shown in Table II. It leads to high logic cost for all operators, especially for divider.

2) *Large Reorder Buffer for Warped Depth Map:* In the forward warping, the warped depth maps DV_L and DV_R require large memory space as a reorder buffer between the forward warping and the filtering as shown in Fig. 7(a). In which, the depth map DL is warped in a raster-scan order to DV_L, and thus a rectangle in DL would be warped to a trapezoid in DV_L due to camera rotation. However, the next 3 × 3 median filtering should be performed in DV_L row by row. Thus, the reorder buffer is necessary but with large size. For example in Fig. 7(b), the size of reorder buffers would be 1024 × 39 × 2 bytes, i.e., 79.9 kbytes, for two-view warping.

3) *Large Z-Buffer for Occlusion Handling:* In addition, the forward warping requires a large Z-buffer for occlusion handling [23]. For example, each pixel in the reference depth map DL is warped to DV_L in the forward warping. However, some pixels in DL will be warped to identical positions in DV_L. For this condition, the most front pixel (i.e., pixel with the smallest depth value) should occlude and overwrite other pixels (i.e., pixels with larger depth value). Thus, a Z-buffer is required to temporarily record the currently most front pixels in the warped depth map to handle occlusion. Therefore, the size of Z-buffer is proportional to the sequence-dependent vertical and horizontal disparity ranges. For example of *Ballet*, the vertical and horizontal disparity ranges are 197 and 55.

TABLE III
QUALITY OF THE PROPOSED BILINEAR INTERPOLATION IN DIFFERENT WINDOW SIZES

	Quality of Y-PSNR (dB)					
Window Size	*Ballet*	*Breakdancers*	*BookArrival*	*Lovebird1*	*Newspaper*	*Kendo*
5 × 3	33.18638	33.06250	36.41172	31.80200	30.67576	33.00001
9 × 5	33.20828	33.16606	36.37078	31.80157	30.67691	32.99998
13 × 7	33.21609	33.17187	36.35280	31.80039	30.67858	32.99997
17 × 9	33.21837	33.16193	36.34878	31.79952	30.67945	32.99996
21 × 11	33.22026	33.14299	36.34814	31.79897	30.67974	32.99996

Thus, the size of Z-buffer is $197 \times 55 \times 2$ bytes, i.e., 21.7 kbytes, for two views.

4) *Complicated Hole Filling:* The hole filling in the VSRS suffers from high computational complexity because the inpainting method is processed on the whole frame iteratively. Therefore, it is not suitable for hardware implementation, and a hardware-friendly hole filling method is demanded.

In summary, the straightforward architecture requires high memory cost due to a large reorder buffer and a large Z-buffer in the forward warping. It also costs high computational complexity due to the large bit-width in the preprocessing and the complicated hole filling method. To solve above problems, the following two sections present the proposed methods in the algorithm-level and architecture-level.

III. Proposed Hardware-Efficient Algorithm

This section proposes a hardware-friendly hole filling method to reduce the complexity, and the column-order warping method to remove the large Z-buffer.

A. Simple Hole Filling Method

To improve the hole filling, Müller *et al.* [24] used the depth information to recognize the background and extrapolate its texture to holes, and Oh *et al.* [25] proposed a depth-based inpainting method which also fills holes with background according to the depth information. However, the required depth information from DV_L and DV_R should be preserved until the final step, and thus the additional memories for depth maps are necessary.

This paper proposes the bilinear interpolation that is a hardware-friendly hole filling method, and does not require depth information. It performs a 2-D low-pass filter with the geometric distance weighting on holes as shown in Fig. 8. In the window, texture pixels are multiplied by the interpolation weightings and masked by the hole map to generate a new pixel. Then, the new pixel can be used for the interpolation of other holes.

Note that the window size is related to the size of hole region, and affects the filling quality as well as internal memory cost. Larger window could cover more available pixels with texture but may involve more noising texture. Table III compares the filling quality by the Y-peak signal-to-noise ratio (PSNR), which is the PSNR between the synthesized frame and the golden captured frame for luma channel Y only, under different window sizes in the test sequences of Table VI. In which, *Ballet* has larger holes, and thus its

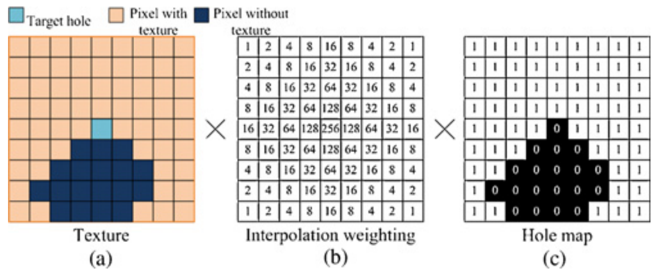


Fig. 8. Proposed bilinear interpolation method for the hole filling step. (a) Texture. (b) Interpolation weighting. (c) Hole map.

quality becomes better when the window size increases. The sequences *BookArrival*, *LoveBird1*, and *Kendo* have smaller holes, and thus the quality is slightly degraded when the window size increases. Thus, we select 9 × 5 in this paper for average quality.

B. Column-Order Warping

As mentioned in Section II, the Z-buffer for occlusion handling needs a large memory space in the forward warping. To reduce the Z-buffer, Morvan [11] proposed the occlusion-compatible scanning order which warps non-rectified images based on the epipolar geometry as shown in Fig. 9(a). In which, e and e' are the epipoles, and the epipolar lines pass through the epipoles. In the corresponding image planes of Fig. 9(b), the occlusion-compatible scanning order method warps depth pixels along the epipolar lines. Note that in the reference view, the warping order should be left-to-right for the left-side of e', and vice for the other side. In our design, only one warping direction is applied because the angle between cameras is usually small and the epipole does not appear in the image planes.

With the occlusion-compatible scanning order method, the Z-buffer can be removed. However, the data access is a problem. In general, the required source data is stored in the external DRAM by the way of the same image rows in the same DRAM rows. But the occlusion-compatible scanning order method should access the pixels along the epipolar lines as shown in Fig. 10(a). These different row accesses would lead to high DRAM row miss.

An approach to avoid the DRAM row miss is the row-order method that performs the warping process row by row, instead of epipolar line by epipolar line, as shown in Fig. 10(b). However, it will violate the original warping order. For example, the original method warps the six pixels in the order of 1, 2, 3, 4, 5, 6 as shown in Fig. 10(a). On the contrary,

TABLE IV
HARDWARE COST COMPARISON OF THE LARGEST DIVIDER

	Bit Width of Dividend (int./exp.+frac.)	Bit Width of Divisor (int./exp.+frac.)	Hardware Cost at 200 MHz (Gate Count)	Reduction (%)
Original	48 + 11	29 + 0	80.1 K	100
Z scaling	39 + 11	20 + 0	44.7 K (pipelined div.)	55.8
Z scaling +IEEE 754	9 + 23	9 + 23	2.9 K (sequential div. with 26 cycles)	3.3 3.3

TABLE V
ANALYSIS OF CONTINUOUS ACCESS IN A COLUMN

	Ballet		*Breakdancers*		*BookArrival*		*Lovebird1*		*Newspaper*		*ChampagneTower*		*Kendo*	
Column height	768		768		768		768		768		960		768	
L-V, R-V	5 – 4	3 – 4	5 – 4	3 – 4	10 – 8	7 – 8	8 – 6	5 – 6	3 – 5	6 – 5	37 – 38	39 – 37	1 – 2	3 – 2
Max. length	102	143	187	194	365	505	359	341	447	574	960	960	533	470
Max. trans. count	147	156	133	140	125	125	131	148	131	117	238	166	149	113

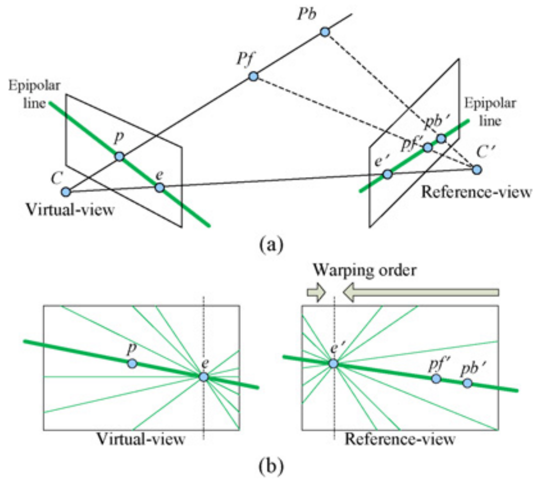


Fig. 9. (a) Epipolar geometry. (b) Concept of occlusion-compatible scanning order method.

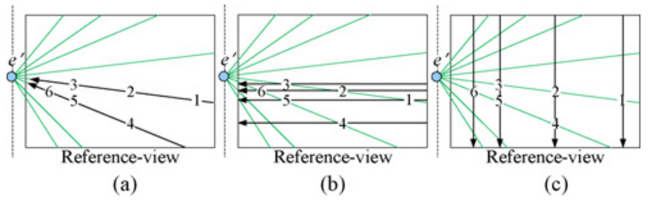


Fig. 10. Warping order in the forward warping process. (a) Original method. (b) Row-order method. (c) Column-order method. The numbers refer to the positions of example pixels.

the row-order method warps them in the order of 3, 2, 6, 1, 5, 4. However, the warping orders of 3, 2, 1 and 6, 5, 4 on the two epipolar lines are different from the original orders of 1, 2, 3 and 4, 5, 6, respectively. This will result in the incorrect warped depth maps as shown in Fig. 11(a) because of foreground pixels replaced by background ones.

To cope with above problem, we propose the column-order method that warps the six pixels in the order of 1, 2, 4, 3, 5, 6, as shown in Fig. 10(c). For the two epipolar lines, their warping orders are 1, 2, 3 and 4, 5, 6, which are identical to the original method. Thus, our warped depth map will be correct as shown in Fig. 11(b). Note that the proposed column-order warping method is suitable for the horizontal camera configuration. For the vertical camera configuration, we should adopt the row-order method.

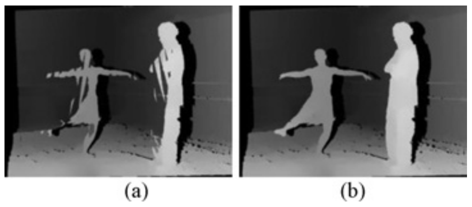


Fig. 11. Warped depth maps. (a) With row-order method. (b) With column-order method.

With the proposed column-order method, one image column is stored in one DRAM row to reduce the overheads of row miss. For maximizing the row-miss reduction, the size of DRAM row should be more than the size of image column. To sum up, by the proposed column-order warping, the Z-buffer could be removed with the least DRAM row miss, and the warped depth map has no quality degradation.

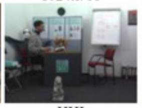
IV. PROPOSED ARCHITECTURE

With the above proposed hardware-friendly algorithms, this section proposes an architecture design for the view synthesis engine. The overall architecture with the proposed hierarchical pipelining is presented first. Then the critical components are described individually.

A. Hierarchical Pipelining

Fig. 12 shows the proposed hierarchical pipelining architecture consisting of frame level and column level. The frame

TABLE VI
TEST SEQUENCES

Sequence Frame size	***Ballet*** 1024x768	***Breakdancers*** 1024x768	***Book Arrival*** 1024x768	***Love bird1*** 1024x768
Texture	MSR	MSR	 HHI	ETRI
Depth map	MSR	MSR	DERS 4.9	DERS 4.9
Sequence Frame size	***Newspaper*** 1024x768	***Champagne Tower*** 1280x960	***Kendo*** 1024x768	
Texture	GIST	MPEG-FTV	MPEG-FTV	
Depth map	DERS 4.9	MPEG-FTV	MPEG-FTV	

MSR: Microsoft research [14]
HHI: Fraunhofer Heinrich-Hertz-Institut [30]
ETRI: Electronics and Telecommunications Research Institute
GIST: Gwangju Institute of Science and Technology

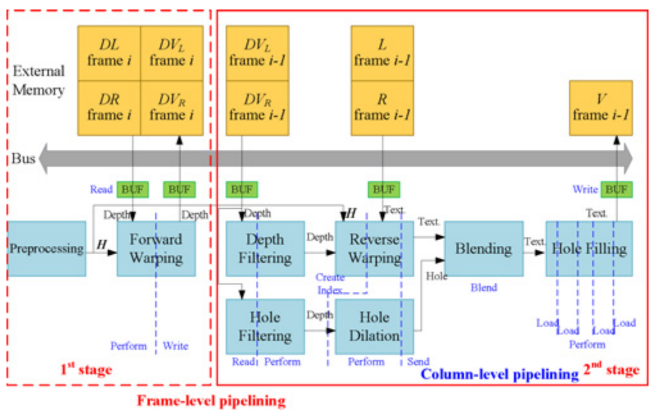


Fig. 12. Proposed hierarchical pipelining architecture.

level is a two stage pipelining. The first stage consists of preprocessing and forward warping, while the second stage consists of depth filtering, reverse warping, blending, and hole filling. Fig. 13(a) shows the schedule of the frame-level pipelining. When the first stage is processing on the frame i, the second stage is processing on the frame $i-1$. Between the two stages, the warped depth maps DV_L and DV_R are stored in the external memory, instead of the internal memory to reduce the internal memory cost.

The second hierarchical level in each frame level stage is column level. With the proposed column-order warping method in Section III, each process is performed column by column, and the depth maps and the textures are configured by the way of one image column on one DRAM row. Fig. 13(b) and (c) shows the schedule of column-level pipelining for the first and second stages in the frame-level, respectively. Note that all the processes from forward warping

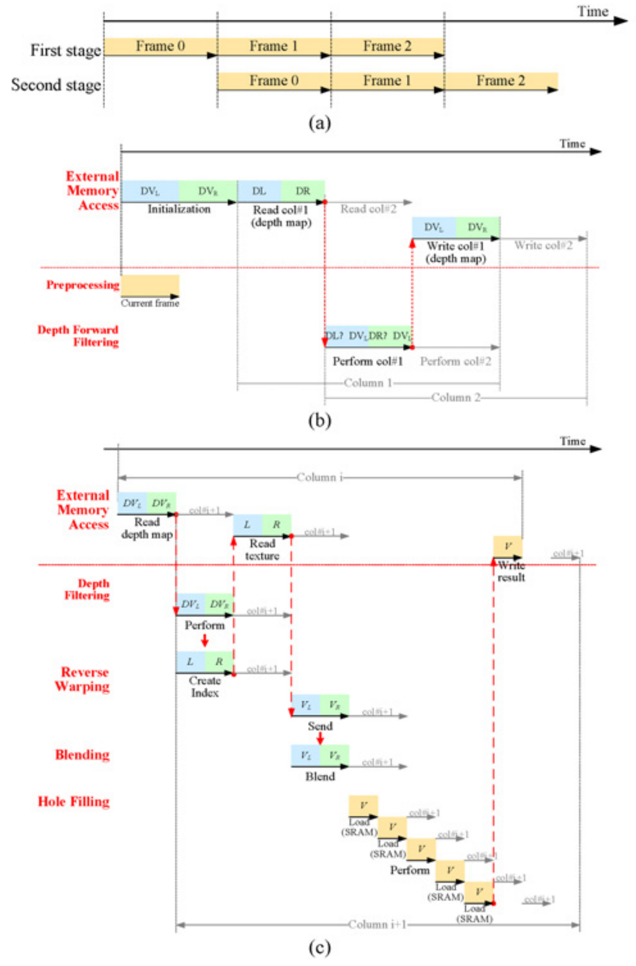


Fig. 13. Schedule of proposed architecture. (a) Frame level pipelining. (b) First stage in column level pipelining. (c) Second stage in column level pipelining.

to blending are first performed on left-view and then on right-view. In addition, the processes associated with textures are applied for the three channels Y, U, V simultaneously.

The detailed schedule of the column-level pipelining in Fig. 13(b) and (c) is presented with the architecture of Fig. 12 as follows. In the first stage of frame-level pipelining, the preprocessing first computes homography matrices for the current frame, and the warped depth maps DV_L and DV_R are initialized at the same time as shown in Fig. 13(b). For higher data access efficiency, this initialization can be assigned to a data memory access controller. Then, the forward warping reads the depth maps DL, DR from external memory in the Read stage, and performs the warping process in the Perform stage, and writes the warped depth maps DV_L, DV_R back to external memory in the Write stage. In the second stage of frame-level pipelining, the depth filtering and the hole filtering read the warped depth maps DV_L, DV_R in the Read stage, and performs median filtering in the Perform stage. With the filtered depth maps, the reverse warping then collects the warping indexes in the Create Index stage, and reads the texture column L, R from external memory in the Read stage, and finally sends the warping virtual-view V_L, V_R to the next component in the Send stage. Finally, the blending and hole

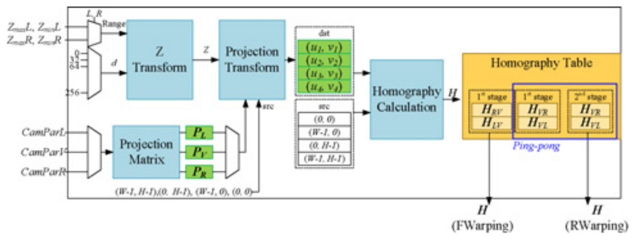


Fig. 14. Architecture of preprocessing component.

filling components generate the final results V, and write them to the external memory.

For the bus access in the schedule, we applied the round-robin policy for the request arbitration. The write request of forward warping and the read request of reverse warping have higher priority because the two accesses are irregular and are raised for higher counts.

B. Preprocessing Component

Fig. 14 illustrates the proposed preprocessing component that calculates the homography matrices $\boldsymbol{H_{LV}}$ and $\boldsymbol{H_{RV}}$ for the forward warping, and $\boldsymbol{H_{VL}}$ and $\boldsymbol{H_{VR}}$ for the reverse warping. In the architecture, the "Z Transform" first converts the depth level d to depth value Z by (3). Then, the "Projection Matrix" combines the camera parameters to form the projection matrices $\boldsymbol{P_L}$, $\boldsymbol{P_V}$, and $\boldsymbol{P_R}$. With a specific depth value and the projection matrices, the "Projection Transform" reads four pixels *src* to calculate their corresponding pixels *dst* in another view by (4). Finally, the "Homography Calculation" uses the four corresponding pairs to solve a homography matrix, and stores it into the homography table for the forward warping and the reverse warping. The corresponding inverse homography matrix is calculated by exchanging *dst* and *src*, and reusing the same "Homograph Calculation" to save hardware cost.

The design problems in this component are the fractional computation and the complicated "Homography Calculation." To solve the high dynamic range fractional computation, we propose a Z scaling method that decreases the dynamic range of Z_{max} and Z_{min} by a proper scale factor. This method will work properly because the related homography matrix could be scaled without changing the warping relation of *src* and *dst* in (2). The scale factor can be computed by

$$ScaleFactor = \min\left\{1, \quad 1/2^{\lceil \log_2 \max\{|Z_{\min}|, |Z_{\max}|\} \rceil - 8}\right\}. \qquad (6)$$

With the Z scaling method, Z_{min} and Z_{max} are restricted in an 8-bit value for the integer part. The proposed Z scaling method could significantly reduce the hardware cost of operators, especially for adders and subtractors. In addition, the IEEE 754 floating-point format is further applied to dividers. Table IV shows that the area of divider could be reduced to 3.3% of original one. The experimental result in Section V shows that the Z scaling method has a negligible impact to the synthesis quality.

For the complicated "Homography Calculation," a hardware-friendly algorithm is demanded to solve the linear system. A general method is matrix decomposition, such as

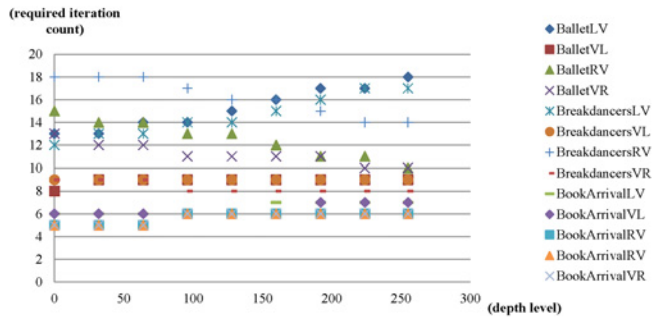


Fig. 15. Required iteration counts in Gaussian-Seidel method.

singular value decomposition, which can be accelerated by the systolic array architecture [26] and the processor-based architecture, called coordinate rotation digital computer [27], [28]. In this paper, we adopt the Gaussian-Seidel method [29] since its sequential computation requires less hardware cost and can easily control the precision of solution by iteration count. The iteration count is set as 20 to meet demands of all depth levels as shown in Fig. 15.

For the homography table in Fig. 14, we apply the LIA method [18] to reduce the elements of homography table from 256 to 8. With the Z scaling method, a homography matrix needs 154×2 bits for $\boldsymbol{H_{base}}$ and $\boldsymbol{H_{inc}}$ to interpolate the finer one. Note that the memories of $\boldsymbol{H_{VR}}$ and $\boldsymbol{H_{VL}}$ are doubled to support the ping-pong mechanism of the two frame-level pipelining stages. Thus, the total memory cost for homography table is $154 \times 2 \times 8 \times 6$ bits, i.e., 14.8 kbytes.

C. Forward and Reverse Warping Components

The forward warping component in Fig. 16(a) uses the depth value from DL and DR to calculate the warped position (u_{dst}, v_{dst}), and warps the depth value of DL and DR, respectively, to DV_L and DV_R. On the contrary, the reverse warping component in Fig. 16(b) uses the depth value from DV_L and DV_R to calculate the warped positions, and warps the texture of L and R to V_L and V_R. In these two components, the "WarpSet," "Linear Interpolation," and "Matrix Multiplication" are used to calculate the warped position.

In these two component designs, the major issue is their external memory access. The two components access the external memory according to the warped positions. However, the successive warped positions are not at the same image column. Thus, the accesses would cross multiple rows in external memory, and the access efficiency is low due to only partial data available in each transmission.

To improve the access efficiency, we propose data packing method that could collect the accesses on an identical memory row into consequent transmissions. The data packing method is implemented as the data packing module and the packing buffer in Fig. 16. Fig. 17 shows the proposed method in the reverse warping as an example. First, in the Create Index stage, the data packing module creates the index table according to the DV_L, which consists of the leading address and the length of each texture segment. Then, in the Read stage, the data packing module fetches the textures L segment by segment, and stores them in the input buffer with marks in the valid

TABLE VII
COMPARISON OF AVERAGE PSNR FOR TEN FRAMES

Sequence	Camera No. of *L*, *V*, *R*	Frame No.	Original VSRS_3_5 (dB)	Proposed Algorithm (dB)	Proposed Implementation (dB)
Ballet	5, 4, 3	90–99	33.081	33.208	33.372
Breakdancers	5, 4, 3	81–90	32.984	33.166	33.121
BookArrival	10, 8, 7	0–9	36.385	36.371	36.499
Lovebird1	5, 6, 8	0–9	31.791	31.802	31.800
Newspaper	3, 5, 6	0–9	30.683	30.677	30.778
ChampagneTower	37, 38, 39	0–9	33.367	33.367	33.361
Kendo	1, 2, 3	0–9	33.000	33.000	33.250
Δ PSNR (to VSRS_3_5)			**0.000**	**0.043**	**0.127**

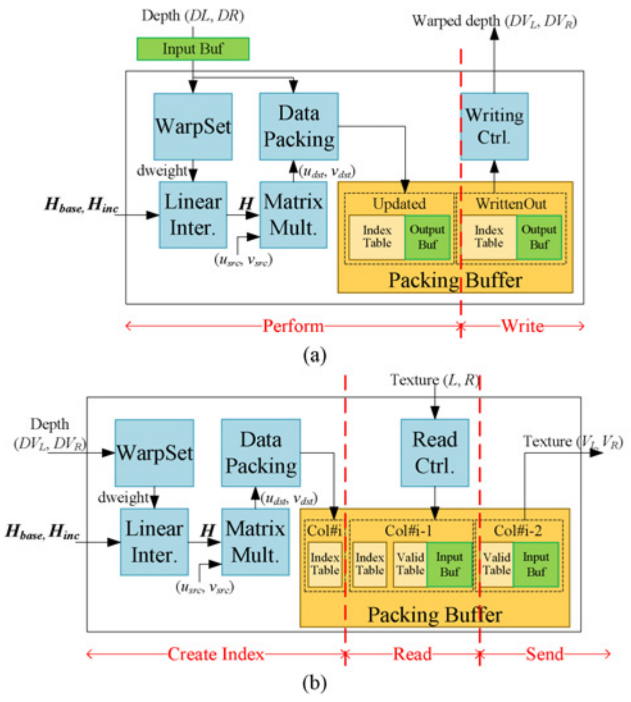


Fig. 16. Architecture of (a) forward warping and (b) reverse warping.

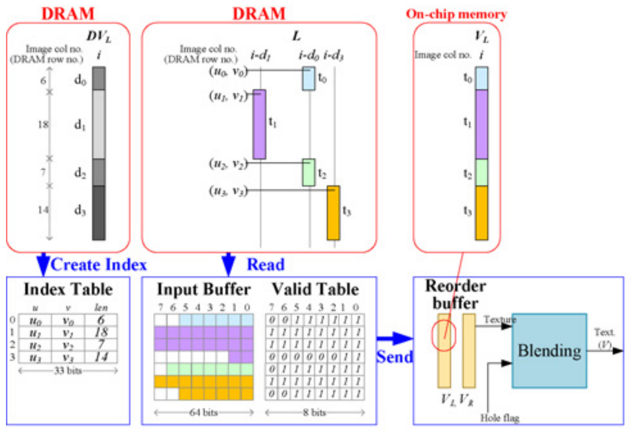


Fig. 17. Proposed data packing method in reverse warping with blending.

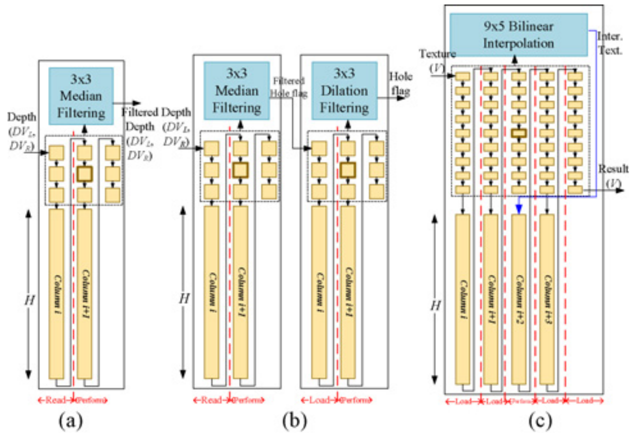


Fig. 18. Architecture of circular FIFO in filtering components. (a) Depth filtering. (b) Hole filtering and dilation. (c) Hole filling.

table. If the length of a texture segment is more than 64-bit, additional one transmission will be issued. Finally, in the Send stage, the information of input buffer and the valid table would be sent to the reorder buffer in the next blending process. The same flow can also be applied to the forward warping for writing the warped depth maps DV_L, DV_R to external memory.

Table V analyzes the continuous access in different test sequences where L-V refers to the selected left-view and target virtual-view and R-V refers to the selected right-view and target virtual-view. The maximum length of continuous access is equal to the column height, and thus the *len* in the index table requires 11 bits for HD1080p video. In this table, the maximum transmission count means the practical count to transmit a whole column through a 64-bit bus. Therefore, we select 256 for the sizes of index table, valid table, and I/O buffers.

D. Filtering Components

The filtering components in Fig. 12 includes the depth filtering, hole filtering, hole dilation, and hole filling components. For these filtering components, we propose the circular first-in first-out (FIFO) architecture as shown in Fig. 18 to reuse data in the internal memory. The proposed architecture consists of a computational module and a circular FIFO buffer, which has a register array and several column memories. Each filtering component is presented as follows.

Fig. 18(a) shows the circular FIFO buffer for depth filtering. In which, the depth pixel is pushed into the circular FIFO buffer and moves along the arrow direction cycle by cycle. Only the depth pixels in the register array are used by the me-

TABLE VIII
PERFORMANCE OF OUR IMPLEMENTATION

Technology process	UMC 90 nm CMOS			
Clock frequency (Hz)	200 M			
External bus (bit)	64			
Equivalent gate count (excluding internal memory)	268.5 K			
Internal memory	69.3 kbytes			
Sequence	*Ballet*	*Breakdancers*	*BookArrival*	*Lovebird1*
Frame size (pixel)	1024 × 768	1024 × 768	1024 × 768	1024 × 768
Execution time (cycles/frame)	2 599 876	2 497 127	2 266 644	2 262 138
Frame rate (f/s)	76	80	88	88
Throughput (pixel/s)	59.76 M	62.91 M	69.20 M	69.20 M
Bandwidth usage (%)	60.77	61.92	60.81	61.90
Sequence	*Newspaper*	*ChampagneTower*	*Kendo*	
Frame size (pixel)	1024 × 768	1280 × 960	1024 × 768	
Execution time (cycle/frame)	2 265 499	3 450 755	2 225 373	
Frame rate (f/s)	88	57	89	
Throughput (pixel/s)	69.20 M	70.04 M	69.99 M	
Bandwidth usage (%)	58.85	58.38	58.02	

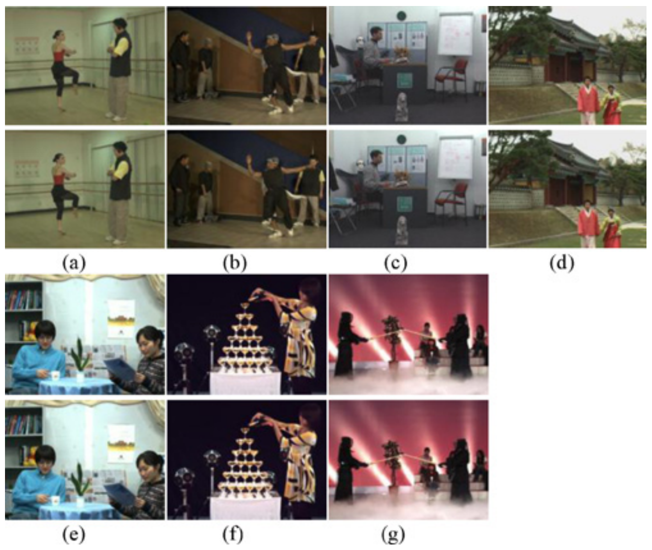


Fig. 19. Synthesis result where the top textures are original result of VSRS and the bottom textures are our implementation result. (a) *Ballet*. (b) *Break-Dancers*. (c) *Book Arrival*. (d) *Love bird1*. (e) *Newspaper*. (f) *Champagne Tower*. (g) *Kendo*.

dian filtering, which is implemented by multiple comparator-trees to select the median depth value. Fig. 18(b) shows the cascaded circular buffer for the hole filtering and the hole dilation. The value of 1 is pushed into the circular FIFO if the input depth pixel is zero and is identified as a hole. To aid this identification, DV_L and DV_R in the external memory should be initially reset to zero as shown in the schedule of Fig. 13(b). The hole filtering is implemented by adders and a comparator. If the summation of 3 × 3 holes is more than 5, the filtered hole flag would be 1. In addition, the hole dilation is implemented by a simple Boolean function. In Fig. 18(c), the hole filling generates new texture for holes. The output of bilinear interpolation module is sent to the memory of middle column, $Coulumn\, i + 2$ because the output is used by the process for other holes.

To sum up, these filtering components takes advantage of the circular FIFO architecture to cooperate with other components in the column-level pipelining, and well reuse the data in the internal memories for column-based process.

E. Blending Component

The blending component first fills the textures V_L and V_R into reorder buffers according to the input buffer and valid table in the reverse warping component as shown in Fig. 17 Then, the blending component uses the textures in the reorder buffers and hole information to blend the two columns to a new one by the truth table in Table I. Then, the final remained holes are handled by the hole filling component mentioned above.

V. EXPERIMENTAL RESULTS

A. Synthesis Performance

Table VI lists the test sequences which are provided by various research institutes [14], [30], [31]. The corresponding depth maps are provided by the same research institutes or estimated by the reference software DERS 4.9 [32].

Table VII compares the synthesis performance in the average of PSNR for ten frames, which is computed using the captured virtual-view and the synthesized one. The synthesis performance shows that our proposed hardware-friendly algorithm and implementation have the slight difference of 0.043 dB and 0.127 dB, respectively. In addition, the corresponding synthesized results are demonstrated in Fig. 19.

B. Implementation Result

The proposed architecture has been implemented by Verilog and synthesized by Synopsys Design Compiler (DC) tool with the 90 nm CMOS technology process. Our view synthesis engine can support the frame size of HD1080p at most under

TABLE IX
COMPARISON BETWEEN OUR DESIGN AND THE STRAIGHTFORWARD ARCHITECTURE

	Straightforward Architecture		Our Design	
Computation	Homography (PRE)	256 × 256-bit	Homography (PRE)	8 × 154-bit
	Inpainting (HF)	Irregular	Bilinear inter. (HF)	hardware-friendly
Memory cost (kbytes)	Homo. table (PRE)	32.8	Homo. table (PRE)	14.8
	Input buf. (FW)	8.2	Input buf. (FW)	8.2
	Reorder buf. (FW)	149.8	Output buf. (FW)	2.2
	Z-buf. (FW)	21.7	Index table (FW)	4.2
	FIFO (depth median)	4.4	FIFO (depth median)	4.4
	FIFO (hole median)	0.5	FIFO (hole median)	0.5
	FIFO (hole dilation)	0.5	FIFO (hole dilation)	0.5
			Input buf. (RW)	14.5
			Index table (RW)	5.2
			Valid table (RW)	1.3
	Reorder buf. (BLD)	4.4	Reorder buf. (BLD)	4.4
	FIFO (HF)	7.1	FIFO (HF)	7.1
	Output buf. (HF)	2.2	Output buf. (HF)	2.2
	Total	**231.6**	**Total**	**69.3**
Required external bandwidth (Mbytes/frame)	Two depth maps	4.15	Six depth maps	12.44
	Three textures	9.33	Three textures	9.33
	Total	**13.48**	**Total**	**21.77**
		(25.3%)		**(40.8%)**
			Simulation	**(61.9%)**

PRE: preprocessing; FW: forward warping; RW: reverse warping; BLD: blending; HF: hole filling.

TABLE X
COMPARISON OF HARDWARE COST AND SYNTHESIS QUALITY WITH OTHER IMPLEMENTATIONS

	Chen [17]	Lin [18]	Our Design
Technology process	TSMC 180 nm	UMC 90 nm	UMC 90 nm
Algorithm	Depth pre-filtering 3-D image warping	Homography only Based on VSRS 3.0	VSRS 3.5
Homography entry count × bit width	–	2 × 118-bit	8 × 154-bit
Clock frequency	80 MHz	200 MHz	200 MHz
Hardware cost (including memory)	162 K gate counts	30.7 K gate counts	765.2 K gate counts
Memory cost	9.26 kbytes	0.24 kbytes	69.3 kbytes
Throughput	20.7 Mpixels/s (25 f/s at SDTV)	–	67.1 Mpixels/s (32.4 f/s at HD1080p)
Synthesis quality (compared algorithm)	N.A.	−0.0059 dB (VSRS 3.0)	+0.127 dB (VSRS 3.5)

the 200-MHz clock frequency. Table VIII lists its performance analyzed by the DC tool. In which, with gate-level simulation, our design reaches different throughputs and bandwidth usages for various sequences because the external access is content-dependent and addressed by the pixel's depth value. The external access is controlled by the data packing module. If the data packing module's buffer is full, the associated computing modules should stop until the data packing module can receive new access tasks. In average, our view synthesis engine could achieve the throughput of 67.1 Mpixels/s, i.e., 32.4 f/s for HD1080p video. In addition, the external bandwidth usage is about 58–61%.

Table IX further lists the detailed hardware cost, and compares it with the straightforward architecture. For the computation, our design could significantly reduce the homography entries by the LIA method [18] and our proposed Z scaling method in the preprocessing. In addition, the irregular inpaiting is replaced by the hardware-friendly bilinear interpolation for the hole filling. For the memory cost, our design could decrease the memory cost to 30% because the large reorder buffer for warped depth maps and the large Z-buffer for occlusion handling have been saved. Although our frame-level pipelining technique results in higher external bandwidth, the simulation bandwidth is affordable.

Table X compares our design with the other implementations, in which, Chen's design [17] applies the different algorithm and has lower throughput. Lin's design [18] only implements the homography calculation and has slightly quality degradation. In summary, our proposed view synthesis engine could achieve the real-time processing for HD1080p resolution as well as low hardware cost.

VI. CONCLUSION

This paper presented a low cost high throughput view synthesis engine based on VSRS with the proposed hardware friendly algorithms and efficient implementations. We reduced the computational complexity by adopting the bilinear interpolation for the hole filling, and the Z scaling method with the floating-point format for the homograph calculation. The memory cost was also reduced by removing the Z-buffer with the column-order warping method, and saving the reorder buffer by the frame-level pipelining. The final implementation can deliver the synthesis results with slightly quality degradation, and support the real-time processing of HD1080p video with the gate count of 268.5 K and the memory cost of 69.3 kbytes. Further extension to support stereoscopic view synthesis can be easily achieved by duplicating the proposed design.

REFERENCES

[1] M. Tanimoto, "Free viewpoint television: FTV," in *Proc. Picture Coding Symp.*, Sep. 2004.

[2] M. Tanimoto and M. Wildeboer, "Framework for FTV coding," in *Proc. Picture Coding Symp.*, May 2009, pp. 429–432.

[3] C. Fehn, "Depth-image-based rendering (DIBR), compression and transmission for a new approach on 3D-TV," *Proc. SPIE Conf. Stereoscopic Displays Virtual Reality Syst.*, vol. 5291, pp. 93–104, May 2004.

[4] C. Vázquez, W. J. Tam, and F. Speranza, "Stereoscopic imaging: Filling disoccluded areas in image-based rendering," *Proc. SPIE 3-DTV, Video, Display V*, vol. 6392, pp. 123–134, Oct. 2006.

[5] C.-M. Cheng, S.-J. Lin, S.-H. Lai, and J.-C. Yang, "Improved novel view synthesis from depth image with large baseline," in *Proc. IEEE ICPR*, Dec. 2008, pp. 1–4.

[6] L. Zhang and W. J. Tam, "Stereoscopic image generation based on depth images for 3D TV," *IEEE Trans. Broadcast.*, vol. 51, no. 2, pp. 191–199, Jun. 2005.

[7] Y.-R. Horng, Y.-C. Tseng, and T.-S. Chang, "Stereoscopic image generation with directional Gaussian filter," in *Proc. IEEE ISCAS*, May.–Jun. 2010, pp. 2650–2653.

[8] W.-Y. Chen, Y.-L. Chang, S.-F. Lon, L.-F. Ding, and L.-G. Chen, "Efficient depth image based rendering with edge dependent depth filter and interpolation," in *Proc. IEEE ICME*, Jul. 2005, pp. 1314–1317.

[9] Y. K. Park, K. Jung, Y. Oh, S. Lee, J. K. Kim, G. Lee, H. Lee, K. Yun, N. Hur, and J. Kim, "Depth-image-based rendering for 3DTV service over T-DMB," *Signal Process.: Image Commun.*, vol. 24, nos. 1–2, pp. 122–136, Jan. 2009.

[10] S. Rogmans, J.-B. Lu, P. Bekaert, and G. Lafruit, "Real-time stereo-based view synthesis algorithms: A unified framework and evaluation on commodity GPUs," *Signal Process.: Image Commun.*, vol. 24, nos. 1–2, pp. 49–64, Jan. 2009.

[11] Y. Morvan, "Acquisition, compression and rendering of depth and texture for multi-view video," Ph.D. thesis, Dept. Electr. Eng., Eindhoven Univ. Technol., Eindhoven, The Netherlands, Apr. 2009.

[12] Y. Mori, N. Fukushima, T. Fuji, and M. Tanimoto, "View generation with 3D warping using depth information for FTV," in *Proc. IEEE 3DTV-CON*, May 2008, pp. 229–232.

[13] *View Synthesis Reference Software (VSRS)*. Version 3.5 [Online]. Available: http://wg11.sc29.org/svn/repos/MPEG-4/test/tags/3D/view_synthesis/VSRS_3_5

[14] N. Fukushima, T. Yendo, T. Fujii, and M. Tanimoto, "Real-time arbitrary view interpolation and rendering system using ray-space," *Proc. SPIE 3-DTV, Video, Display IV*, vol. 6016, pp. 1–12, Nov. 2005.

[15] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, "High-quality video view interpolation using a layered representation," *ACM Trans. Graphics*, vol. 23, no. 3, pp. 600–608, Aug. 2004.

[16] S. Rogmans, J. Lu, and G. Lafruit, "A scalable end-to-end optimized real-time image-based rendering framework on graphics hardware," in *Proc. IEEE 3DTV-CON*, May 2008, pp. 129–132.

[17] 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 *Proc. IEEE ICME*, Jul. 2006, pp. 2069–2072.

[18] P. C. Lin, P. K. Tsung, and L. G. Chen, "Low-cost hardware architecture design for 3D warping engine in multiview video applications," in *Proc. IEEE ISCAS*, May–Jun. 2010, pp. 2964–2967.

[19] *Open Source Computer Vision* [Online]. Available: http://opencv.willowgarage.com/wiki

[20] *Reference Softwares for Depth Estimation and View Synthesis*, document M15377, ISO/IEC JTC1/SC29/WG11, Apr. 2008.

[21] *View Synthesis Algorithm in View Synthesis Reference Software 2.0 (VSRS2.0)*, document M16090, ISO/IEC JTC 1/SC29/WG11, Feb. 2008.

[22] A. Telea, "An image inpainting technique based on the fast marching method," *J. Graphics, GPU, Game Tools*, vol. 9, no. 1, pp. 25–36, 2004.

[23] *View Synthesis Software and Assessment of its Performance*, document M15672, ISO/IEC JTC1/SC29/WG11, Jul. 2008.

[24] K. Müller, A. Smolic, K. Dix, P. Merkle, P. Kauff, and T. Wiegand, "View synthesis for advanced 3D video systems," *EURASIP J. Image Video Process.*, vol. 2008, no. 438148, pp. 1–11, 2008.

[25] K.-J. Oh, S. Yea, and Y.-S. Ho, "Hole-filling method using depth based in-painting for view synthesis in free viewpoint television (FTV) and 3D Video," in *Proc. Picture Coding Symp.*, May 2009.

[26] R. P. Brent, F. T. Luk, and C. F. V. Loan, "Computation of the singular value decomposition using mesh-connected processors," *J. Very Large Scale Integr. Comput. Syst.*, vol. 1, no. 3, pp. 242–270, 1985.

[27] J. R. Cavallaro and F. T. Luk, "CORDIC arithmetic for an SVD processor," *J. Parallel Distributed Comput.*, vol. 5, no. 3, pp. 271–290, Jun. 1988.

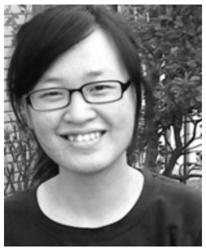
[28] R. Andraka, "A survey of CORDIC algorithms for FPGAs based computers," in *Proc. ACM/SIGDA Int. Symp. FPGA*, Feb. 1998, pp. 191–200.

[29] A. Kaw, *Introduction to Matrix Algebra*, 1st ed. Tampa, FL: Univ. South Florida, 2008, ch. 8.

[30] *HHI Test Material for 3D Video*, document M15413, ISO/IEC JTC 1/SC 29/WG 11, Apr. 2008.

[31] Tanimoto Laboratory. (2008). *MPEG-FTV* [Online]. Available: http://www.tanimoto.nuee.nagoya-u.ac.jp/mpeg/mpeg_ftv.html

[32] *Depth Estimation Reference Software* [Online]. Available: http://wg11.sc29.org/svn/repos/MPEG-4/test/tags/3D/depth_estimation

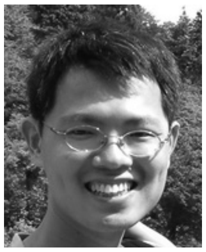


Ying-Rung Horng received the B.S. and M.S. degrees in electronic engineering from National Chiao-Tung University, Hsinchu, Taiwan, in 2008 and 2010, respectively.

She is currently an Engineer with MediaTek, Inc., Hsinchu.

Yu-Cheng Tseng (S'07) received the B.S. degrees in electronic engineering from National Chiao-Tung University (NCTU), Hsinchu, Taiwan, in 2006. He is currently a Ph.D. student with the Department of Electronics Engineering, NCTU.

His current research interests include 3-D video processing, and hardware architecture design and implementation.



Tian-Sheuan Chang (S'93–M'06–SM'07) received the B.S., M.S., and Ph.D. degrees in electronic engineering from National Chiao-Tung University (NCTU), Hsinchu, Taiwan, in 1993, 1995, and 1999, respectively.

He is currently an Associate Professor with the Department of Electronics Engineering, NCTU. From 2000 to 2004, he was a Deputy Manager with Global Unichip Corporation, Hsinchu. His current research interests include (silicon) intellectual property and system-on-a-chip design, very large scale integration signal processing, and computer architecture.