An Advanced Video and Depth Depacking Architecture for 3D Applications

PIN-CHEN KUO, AN-JIE LIN, BIN-DA LIU AND JAR-FERR YANG
Advanced Optoelectronic Technology Center
Department of Electrical Engineering
National Cheng Kung University
Tainan, 701 Taiwan
E-mail: {ken132m; ken8612}@hotmail.com; {bdliu; jefyang}@mail.ncku.edu.tw

In this paper, a first hardware design for the 2D-compatible format and an architecture for the depth-image-based rendering (DIBR) is proposed. Compared to the format that uses one view plus one depth, the advanced 2D-compatible format reduces the bitrates of both color and depth frames. In order to achieve 3D image generation in real time, a hardware architecture of 2D-compatible format is firstly designed in this paper. In this paper, an edge-based DIBR algorithm is also proposed and implemented in to hardware architecture. The proposed edge-based DIBR algorithm achieves better image quality than the original DIBR algorithm. This paper combines these two architecture in to the advanced video and depth depacking architecture which can reduce the bitrates and increase the image quality of 3D video. Simulation results also show that image quality of the proposed system is better than that of the original DIBR algorithm. For the hardware cost, the proposed system requires 6.34k gates. The operation frequency can reach 100 MHz, which can supports the maximum frame size up to FHD (1920 × 1088) in real time.

Keywords: frame compatibility, packing format, DIBR, 3D video, 3DTV

1. INTRODUCTION

Products such as VCDs, DVDs, digital cameras, and three-dimensional television (3DTV) are commonly used for entertainment [1-5]. 3DTV has become popular recently. Stereo content is used to create 3D perception in 3DTV systems. An autostereo-scopic image rearranged from two different images is displayed to the viewer. The traditional method for constructing stereo images is to capture two or more images by cameras and send the streams to the receiver. This results in high quality, but the required transmission bandwidth and data storage are high.

Several depth-based methods [6], such as multi-view video plus depth (MVD) and layered depth video (LDV), have been considered as the inputs for 3DTV systems to reduce bandwidth for multi-view video transmission. For MVD transmission, the most popular frame-compatible formats are side-by-side (SBs) and top-and-bottom (TaB). However, these formats cannot be comfortably viewed on 2D displays and do not directly support naked-eye multi-view 3D displays. To solve the 2D viewing compatibility problem and further reduce transmission bandwidth, Yang et al. proposed a 2D-compa-
The depth-image-based rendering (DIBR) [8] technique has received a lot of attention in the research community of 3D broadcasting as a prospective technology for 3DTV systems. The DIBR algorithm requires an intermediate image and its corresponding depth information to synthesize associated virtual view images. The depth map is a 2D gray-level image that records the distance between objects and the camera in the 3D world. The nearest and farthest objects are given values of 255 and 0 in the depth map, respectively. The depth map can be captured using a Z-camera or obtained using algorithms such as stereo matching [9-11]. For 2D-to-3D conversion systems [12-15], the DIBR algorithm can generate any viewpoint when the depth map is available. According to one study [16], the transmission bandwidth of a gray-level image is at least 33% lower than that of a color image. Moreover, based on the special property of the depth map, the required bit rate of the depth map is only 20% that of the color image [17]. When a DIBR system is used to synthesize virtual views, there might be some differences between the synthesized images and the actual scene. DIBR degrades image quality and is time-consuming. The generation of higher-quality virtual view images in real time is of interest. Some hardware architectures for DIBR algorithms have been proposed [18-25] to reduce processing time. The present paper proposes a DIBR hardware architecture that uses the video and depth packing format [7] as the input signal to generate high-quality virtual view images.

The rest of this paper is organized as follows. Section 2 introduces the basic concepts of the 2D-compatible packing format and DIBR technology. Section 3 describes the proposed architecture, which includes 2D-compatible format depacking and DIBR hardware. Section 4 describes the hardware architecture of the proposed system in detail. Section 5 shows the experimental results. Finally, conclusions are given in Section 6.

2. RELATED WORK

2.1 2D-Compatible Packing and Depacking Format

The 2D-compatible vertical packing and depacking process [7] is shown in Fig. 1. The vertical resolution of the color image and depth map are downscaled to three fourths of the original resolution by using bi-cubic interpolation. Three depth pixels can be arranged into one RGB color pixel with a value of 0 to 255. Thus, every three depth pixels along the vertical direction can be rearranged into one RGB pixel. The conversion formula of a depth frame into a new depth frame is:

$$d_{\text{new}}(i, 1 + j, k) = d(i, 3 \times j + k)$$

where $i = 1, 2, \ldots, m; j = 0, 1, 2, \ldots, n = 1; k = 1, 2, 3;$ and $m$ and $n$ represent the width and height of the image, respectively. $d(i, j)$ denotes the original depth value of the pixel at $(i, j)$ and $d_{\text{new}}(i, j, k)$ is the newly color-packed depth pixel at $(i, j)$ with RGB channels, where $k = 1, 2,$ and 3 denote R, G, and B channels, respectively. Thus, each color-packed depth pixel represents three depth values along the vertical direction.
After downscaling and rearranging, the color-packed depth map is equally split into top and bottom parts and flipped upside down. Finally, a downscaled color frame is combined with two separated depth frames. Combining 3/4-resized color video and two halves of the flipped color-packed depth map (1/8), the final resolution of the packing format is 1920×1080 pixels. As shown in Fig. 1 (a), the newly packed video and depth frame is suitable for transmitting video streams for traditional 2D TV broadcasting systems. Fig. 1 (b) shows the flow chart for depacking the 2D-compatible video and depth packing format [7].

![Flow charts of 2D-compatible video and depth format](image)

Fig. 1. Flow charts of 2D-compatible video and depth format for (a) packing and (b) depacking processes.

2.2 DIBR Process

For generation of left and right views, the simplified DIBR algorithm shown in Fig. 2 consists of depth map preprocessing, 3D warping, and hole filling processes. Depth map preprocessing is usually achieved using a burring filter to reduce gap of depth to avoid large holes after 3D warping. The 3D warping process, which depends on the selected view, performs a horizontal shift of the pixels according to the preprocessed depth values and the camera model. Generally, a larger depth will lead to a larger disparity shift in virtual views. Due to depth differences and position quantization, the warped image may have many unknown pixels, which are called cracks or holes. A hole filling process is used to fill these holes to complete the synthesis of target views.

3. PROPOSED METHOD OF 2D COMPATIBLE FORMAT BASED DIBR

This paper proposes a hardware architecture for virtual view generation with the
2D-compatible video and depth packing format used as the input. The block diagram of the proposed architecture system is shown in Fig. 3. There are four main functional blocks: up-sampling, down-sampling, warping, and hole filling. In order to recover the scaled image in the 2D-compatible image, the depth map and color image have to be upscale to four thirds of the packed resolution. The up-sampling process is applied for resizing the color frame and depth map in to four times. After the up-sampling procedures, the resolutions of the color and depth frames are downscaled one third in the down-sampling stage. After scaling, the color and depth frames with their original vertical resolutions are available. In the third stage, the 3D warping in the original DIBR approach is performed. In the final stage, hole filling is applied to fill occluded regions.

Fig. 2. Block diagram of DIBR system for generating left and right virtual views.

Fig. 3. Block diagram of proposed depacking and DIBR system.

3.1 Filter Design

Several interpolation techniques have been used for resizing the resolution of images [26-28]. A scaling method involves a trade-off among efficiency, smoothness,
and sharpness. If the values of pixels are averaged, the image appears smooth. Otherwise, the boundary is jagged. The simplest interpolation method is nearest-neighbor interpolation [26], where the value of the nearest pixel is applied for the value of the target pixel. However, this simple method creates obvious artifacts. Another common method is bilinear interpolation, where the value of a newly generated point is a linear combination of the surrounding pixels [27]. Although bilinear interpolation is simple, it leads to some blurring. Other common methods are bi-cubic [28, 29] and B-spline [30] interpolation. These methods consider more pixels to determine the interpolated pixel, which can reduce the blurring effect at boundaries at the cost of higher computational complexity. Lanczos interpolation is a method that is implemented using sinc function with a Lanczos window. Turkowski and Gabriel [31] found that the Lanczos2 filter is the best compromise in terms of aliasing, ringing, and sharpness, compared with truncated sinc, Bartlett, cosine, and Hann-windowed sinc, for the interpolation of 2D images.

3.1.1 Filter design for up-sampling

In order to resize the color frame and depth map in to four times, a Lanczos filter with four reference pixels is adopted in the proposed system. The reconstruction kernel of the Lanczos filter, called Lanczos kernel \( L(x) \), is expressed as:

\[
L(x) = \begin{cases} 
1, & x = 0 \\
\frac{a \sin(\pi x) \sin(\pi x / a)}{\pi^2 x^2}, & 0 < |x| < a \\
0, & x \leq 0 \text{ and } x \geq a
\end{cases}
\]  

(2)

where \( a \) is a positive integer that is used to control the width of the Lanczos kernel. In the up-sampling process, the size of the Lanczos kernel is \( 1 \times 4 \). The interpolation formula for one dimension is defined as:

\[
S(y) = \sum_{j=-a}^{a} s_j L(y - j)
\]

(3)

where \( s_j \) is the value of the pixel at coordinate \( y \) in the vertical direction in the original image and \( S(y) \) is the value of the interpolated pixel. The Lanczos2 filter \( (a = 2) \) is the best sinc filter due to the reduced aliasing, minimal ringing, and high sharpness [31]. Thus, the width of the Lanczos kernel \( a \) is set as \( 2 \) in the proposed system. The Lanczos kernel \( L_2(x) \) is shown in Fig. 4 (a).

In this procedure, the vertical resolution is upscaled four times for the color and depth frames. Between every two pixels of the packed color frame, three pixels should be interpolated. Even though the depth frame is rearranged, the concept of upscaling for the depth frame is the same as that for the color frame. The relative positions of the original pixels and interpolated pixels in the generated frame are shown in Fig. 5 (a). An example of the Lanczos kernel for coordinate \( x \) with relative coefficients is also shown in Fig. 5 (a), where \( A, B, C, \) and \( D \) are the known pixels, and \( a, b, \) and \( c \) are the target interpolated pixels.
Fig. 4. (a) Lanczos kernel with $a = 2$; (b) Modified Lanczos kernel for down-sampling.

Fig. 5. Relative positions and relationships; (a) up-sampling, and (b) down-sampling process.

All coefficients of the proposed system are defined as:

\[
\begin{align*}
L_2(0.25) &\approx 0.87735 \\
L_2(0.5) &\approx 0.57315 \\
L_2(0.75) &\approx 0.23534 \\
L_2(1.25) &\approx -0.08472 \\
L_2(1.5) &\approx -0.06348 \\
L_2(1.75) &\approx -0.01782
\end{align*}
\]  

(4)

In Eq. (4), these filter coefficients are too complicated for hardware implementation. In order to reduce hardware resources and increase computation speed, the filtering coefficients should be adjusted. In the hardware design, 2-based exponential function can be considered as a shift operation, which is easy to be implemented. Thus, the filtering coefficients are approximated by the combinations of two or three 2-based exponential functions as:
\[
\begin{align*}
L'_2(0.25) &= 2^{-1} + 2^{-2} + 2^{-3} \\
L'_2(0.5) &= 2^{-1} + 2^{-4} + 2^{-7} \\
L'_2(0.75) &= 2^{-3} + 2^{-4} + 2^{-5} + 2^{-6} \\
L'_2(1.25) &= -2^{-4} - 2^{-6} - 2^{-7} \\
L'_2(1.5) &= -2^{-4} - 2^{-7} \\
L'_2(1.75) &= -2^{-4} - 2^{-7} .
\end{align*}
\] 

After these approximations, these filtering coefficients can be easily implemented in hardware. Moreover, the degradation of image quality is too small to be observed because new coefficients are very close to the original ones. Thus, the filtering process can be realized by at most four shift operators and additions. The interpolated pixels are determined as:

\[
\begin{align*}
S(a) &= s(A) \times L'_2(-1.25) + s(B) \times L'_2(-0.25) + s(C) \times L'_2(0.75) + s(D) \times L'_2(1.75) \\
S(b) &= s(A) \times L'_2(-1.5) + s(B) \times L'_2(-0.5) + s(C) \times L'_2(0.5) + s(D) \times L'_2(1.5) \\
S(c) &= s(A) \times L'_2(-1.75) + s(B) \times L'_2(-0.75) + s(C) \times L'_2(0.25) + s(D) \times L'_2(1.25)
\end{align*}
\] 

3.1.2 Filter design for down-sampling

In this stage, the Lanczos filter is adopted due to its high performance in the boundary region. Unlike down-sampling, up-sampling keeps all the pixels’ information in the original images. However, the details in an unscaled image may be eliminated by the down-sampling procedure which degraded the image quality. To solve this problem, seven reference pixels are used to determine the value of interpolated pixels. The relative positions of pixels are shown in Fig. 5 (b).

The Lanczos kernel in the down-sampling process has \( a = 2 \). The relative coefficients for filtering are modified. The size of the Lanczos kernel is \( 1 \times 7 \), as shown in Fig. 6. The modified coefficients of the Lanczos kernel can achieve quality similar to that for the original Lanczos kernel. Since the sum of the coefficients is equal to one, there is no degradation in the smooth region. The interpolated pixels in down-sampling are defined as:

\[
D' = \frac{-1}{32} A + \frac{9}{32} C + \frac{16}{32} D + \frac{9}{32} E + \frac{-1}{32} G .
\]

Down-sampling uses a ratio of one third based on the ratio of up-sampling (four). Since the adjacent pixels are almost the same after up-sampling, the down-sampling method applied in the proposed system selects only the first pixel every three pixels.

3.2 Edge-based DIBR

3.2.1 3D warping

For 3D warping, the depth values are transformed into disparity values, and then each pixel in the reference view is shifted to the new coordinates in the virtual view.
according to the corresponding disparity value. Disparity represents the parallax of an image between the left and right eyes. A pixel with depth value \( d_m \) located at coordinates \((x, y)\) is warped to coordinates \((x - (d_m - Z_c)/\text{scale}, y)\) in the virtual view, where \( \text{scale} \) and \( Z_c \) are the parameters of warping. In this paper, \( \text{scale} \) is set to 4. The major problem of DIBR is that more than one pixel may be warped to the same coordinates. This overlap occurs because the region is visible in the reference view but invisible in the target virtual view. Only the foreground object is visible in the overlap region. Therefore, the pixels with the largest disparity are the correct pixels for the target region.

\[
\text{Fig. 6. Modified Lanczos kernel for down-sampling.}
\]

To avoid the mismatch problem, two additional variables are used in 3D warping, namely \( \text{flag} \) and \( d \). The variable \( \text{flag} \) is used to record whether a pixel is occupied:

\[
\text{flag}_{x,y} = \begin{cases} 
1, & \text{\( (x,y) \) is occupied} \\
0, & \text{otherwise}
\end{cases}
\]  \hspace{1cm} (8)

Variable \( d \) is an intermediate value of the depth. Consider pixel \( p \) at \((x, y)\) with disparity \( \text{disp} \). The corresponding warping coordinates are \((x - \text{disp}, y)\). If the coordinates \((x - \text{disp}, y)\) are empty, \( \text{flag}_{(x - \text{disp}, y)} \) is equal to zero and the target pixels can be easily filled by \( p \). If \( \text{flag}_{(x - \text{disp}, y)} \) is equal to one, the coordinates \((x - \text{disp}, y)\) are occupied. The correct depth of this pixel can be chosen from a temporary disparity \( d_{\text{temp}} \) or \( \text{disp} \). \( d \) is defined as:

\[
d_{(x,y)} = \begin{cases} 
\text{disp}, & \text{\( \text{flag}_{(x,y)} = 0 \)} \\
\text{temp}, & \text{\( \text{flag}_{(x,y)} = 1 \) and \( d_{\text{temp}} > \text{disp} \)} \\
\text{disp}, & \text{\( \text{flag}_{(x,y)} = 1 \) and \( d_{\text{temp}} < \text{disp} \)}
\end{cases}
\]  \hspace{1cm} (9)

where \( d_{\text{temp}} \) is a temporary disparity of coordinates \((x, y)\).

If \( \text{flag}_{(x, \text{disp}, y)} \) is zero, \( d_{(x,y)} \) is filled with \( \text{disp} \). If \( \text{flag}_{(x, \text{disp}, y)} \) is one, to determine the disparity of coordinates \((x, y)\), a comparison is made between \( \text{disp} \) and \( d_{(x, y)} \). If \( \text{disp} \) is smaller than \( d_{(x, \text{disp}, y)} \), the value of coordinates \((x, y)\) is \( d_{(x, \text{disp}, y)} \). By using this condition in the warping process, a simple skill is applied to simplify this comparison. If the left view
is treated as the reference view to synthesize the right view, the warping order of pixels should be from the left of the frame to the right. If pixel $p_1$ at coordinates $(x, y)$ with disparity $d_1$ and pixel $p_2$ at $(x + \Delta x, y)$ with disparity $d_2$ are warped to the same coordinates, the relation between the two pixels can be defined as:

$$x - d_1 = x + \Delta x - d_2.$$  \hspace{1cm} (10)

Eq. (10) can be further simplified into:

$$d_2 = d_1 + \Delta x.$$  \hspace{1cm} (11)

Eq. (11) shows that if two pixels are warped to the same coordinates, the disparity on the right side is larger than that on the left side. Based on this concept, the checking procedure of disparity can be skipped when the warping order is from left to right. Similarly, if the right view is treated as the reference view to synthesize the left view, the warping order is from right to left. An example is shown in Fig. 7. Using this warping order, the variable $d$ becomes useless. However, variable flag is still required to determine whether the coordinates are occupied.

3.2.2 Edge-based hole filling

Holes created in 3D warping make some areas invisible to one eye but visible to the other. Holes usually occur in the object boundaries, which have different depth values. However, the boundaries of the objects in color and depth frames could not be matched perfectly, as shown in Fig. 8. After 3D warping, if the foreground lies on the left-hand side of the object with larger disparity, holes will occur in the region between the foreground and background, as shown in Fig. 8 (c). These holes should be filled with the remaining information of the background.
Since the boundary region is filled with the wrong information, the quality degradation near the boundary is very obvious. This paper proposed a simple edge detection method to solve this problem:

\[
\text{Edge}(x, y) = \begin{cases} 
1, & \text{if } d_{x+1,y} - d_{x,y} > d_t \\
0, & \text{otherwise}
\end{cases}
\] (12)

where \(d\) is the disparity of the pixel and \(d_t\) is the threshold for the detection of large variance of disparity at the boundary. The pixel at coordinates \((x, y)\) is labeled as the edge if \(d_{x+1,y} - d_{x,y} > d_t\). When a boundary is detected, holes are filled using foreground information. Otherwise, if the boundaries of the color and depth frames match perfectly, the background objects are warped to the left side and the holes are filled with the background information on the right side, as shown in Fig. 9.

Fig. 9. Hole filling with right side background object.

4. METHODS OF 2D COMPATIBLE FORMAT DEPACKING AND DIBR

One of the drawbacks for the proposed system is its memory requirement in the hardware implementation. The large window size of a smoothing filter increases the memory requirement. As in the inpainting approach, the best matching block for hole filling is searched for in the search window. The required memory in the hardware implementation is determined by the size of the search window. Because the hardware implementation of the proposed system is designed for 3DTV and mobile phones, optimizing memory usage is an important issue.

4.1 Memory Utilization of Filter

Since resizing the color and depth frames requires a lot of memory in the depacking procedure, some pixels are processed at the same time to reduce the memory requirement. In order to utilize the reusability of memory between up- and down-sampling, we should realize the features of up- and down-sampling respectively. The reusability of the up- and down-sampling is very obvious that the reference pixels are repeated every four pixels in the up-sampling and three pixels in the down-sampling. The lowest common multiple of three and four, twelve, is adopted. However, more than
twelve line buffers are needed. Some reference pixels are also considered. The pixels needed for both up- and down-sampling are shown in Fig. 10.

Fig. 10. Twelve line buffers for storing intermediate result.

4.2 Memory Utilization for Depacking

After the color frame and depth frame are depacked, the relationship between the color and depth frames in the packing format is considered to synthesize the virtual view using DIBR. The result of 2D-compatible vertical packing is shown in Fig. 11. The first important part of this image is the upper depth frame after flipping. If we want to receive color and relative depth simultaneously in the corresponding position, the depth information must be stored. The extreme example of an access problem in the upper part is when the first row information of the upper color frame and the last row information of the upper depth frame are required at different times in the DIBR procedure. In this example, the whole upper depth frame should be stored in advance before the receiving process of the color frame. The lower part of the image with 2D packing has the same problem. Because the lower part of the depth frame is in the bottom part of the result, some information of the lower color frame should be stored. The extreme example of an access problem in the lower part is when the first row information of the lower color frame and the last row information of the lower depth frame are required at different times in the DIBR procedure. Therefore, the whole lower part of the image with 2D packing should be stored in advance.

Fig. 11. Example of 2D-compatible vertical packing format.

The memory requirement of the upper part is the size of the upper depth frame after rearrangement, and that of the lower part is even higher. Because the color frame is above the lower depth frame, the memory requirement is half of the color frame and the lower depth frame. The memory size is four times that of the upper part because of the first row of the lower color frame and its corresponding depth which locates in the last
row of the lower depth frame. Because of the huge amount of memory required for the upper and lower parts in the frame, memory reuse is considered. A simple method would be to separate the processes of up- and down-sampling for the upper and lower parts. The upper part needs the upper depth frame, and the lower part needs the lower part of 2D-compatible vertical packing image. Combining the requirements of these two parts, the memory size is that of the lower part, which equals half of the frame.

4.3 Overall Architecture of Proposed System

The architecture of the proposed system, which combines depacking and DIBR, is shown in Fig. 12. In the architecture, the half frame buffer is used for storing the upper part of the image, which is packed first. Then, the first point in each row are sent into up-sampling unit, the modified Lanczos filter for up-sampling the color and depth frames with twelve line buffers for each frame. After the up-sampling, the first point in each row is saved in the up-sampling buffer, the second point in each row is sent into up-sampling unit and so on. In the same time, the modified Lanczos filter is used to down-sample the color and depth frames with twelve line buffers for the first point in each row. In the 3D warping stage, all pixels in the color frame are moved horizontally based on their corresponding disparity values, which are derived from the depth information. Finally, the values of neighboring pixels are used to fill holes by using a simple edge-oriented filling method. After the whole process in the architecture, the first point in each row in the output buffer is output pixel by pixel. When the processing of upper part of 2D-compatible image is finished, the lower part of 2D-compatible image is stored into the half frame buffer.

5. SIMULATION RESULTS AND DISCUSSION

To evaluate the proposed architecture, eight sequences were tested in simulations. Table 1 shows the resolutions of the test sequences. Image samples of the test sequences
are shown in Fig. 13. The performance of the proposed method was evaluated in terms of the peak signal-to-noise ratio (PSNR), which is an objective method for judging image quality since it only calculates the color difference between two images.

Table 2 shows the PSNR results of depacking the 2D-compatible packing format. For the depacking procedure, a two-stage filter is applied to resize the vertical resolution. By using the modified Lanczos filter for both up- and down-sampling, the PSNR results are 41.9 and 43.01 dB for color and depth frames on average, respectively. “Teddy” and “Cones” have complex backgrounds in the color frame and depth map, and thus the results for the other six images have better quality. However, the visual quality of these two images is almost the same as the original images.

<table>
<thead>
<tr>
<th>Image size</th>
<th>S01</th>
<th>S02</th>
<th>S03</th>
<th>S04</th>
<th>S05</th>
<th>S06</th>
<th>Teddy</th>
<th>Cones</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>S01</td>
<td>1920×1088</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S02</td>
<td>1024×7681</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S03</td>
<td>900×540</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PSNR (dB)</th>
<th>S01</th>
<th>S02</th>
<th>S03</th>
<th>S04</th>
<th>S05</th>
<th>S06</th>
<th>Teddy</th>
<th>Cones</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color</td>
<td>48.91</td>
<td>42.72</td>
<td>39.03</td>
<td>44.31</td>
<td>44.32</td>
<td>41.28</td>
<td>37.45</td>
<td>37.24</td>
<td>41.90</td>
</tr>
<tr>
<td>Depth</td>
<td>55.38</td>
<td>40.93</td>
<td>39.64</td>
<td>47.33</td>
<td>48.38</td>
<td>39.06</td>
<td>36.17</td>
<td>37.17</td>
<td>43.01</td>
</tr>
</tbody>
</table>

Fig. 13. Samples of test images for (a) S01, (b) S02, (c) S03, (d) S04, (e) S05, (f) S06, (g) Teddy, and (h) Cones.
The PSNR comparisons of “Teddy” and “Cones” are shown in Table 3. In Table 3, the DIBR method proposed in a previous study [8] is applied for the simulations. The MVD format and the 2D-compatible format [7] were respectively input to the DIBR. As shown in Table 3, the PSNR of DIBR [8] with the 2D compatible format is similar to that of DIBR [8] with the MVD format, indicating that the 2D-compatible format does not affect the quality of synthesis views. The proposed edge-based DIBR and original DIBR with the 2D-compatible format are also compared in Table 3. The simulation results reveal that the proposed edge-based DIBR algorithm has better image quality than that for the original DIBR. In Table 3, warping parameter scale is set to 4. The results show that the image quality after packing and depacking is almost the same for synthesizing the virtual view. After the simple edge detection, the quality is slightly increased and the boundary is preserved as shown in Figs. 14-17, which show the synthesized left view for “Teddy” and “Cones”. From the synthesized left views, the original DIBR creates serious boundary effects. The proposed edge-based DIBR algorithm solves most of the boundary effect and improves image quality. However, some boundary effects in the Figs. 16 and 17 still cannot be completely solved in the border of neighboring edges and the objects with similar characteristics. More information such as the characteristics of objects and color information of texture should be referred in the hole filling process. Moreover, Figs. 14-17 also reveal that the 2D-compatible format does not affect the quality of synthesis views.

<table>
<thead>
<tr>
<th>Scale = 4</th>
<th>Teddy</th>
<th>Cones</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIBR [8] with MVD format</td>
<td>27.11</td>
<td>24.16</td>
</tr>
<tr>
<td>DIBR [8] with 2D-compatible format</td>
<td>27.21</td>
<td>24.19</td>
</tr>
<tr>
<td>Edge-based DIBR with 2D-compatible format</td>
<td>27.60</td>
<td>24.42</td>
</tr>
</tbody>
</table>

Fig. 14. Comparison of Teddy (left view): (a) DIBR with MVD format, (b) DIBR with depacking format, (c) edge-based DIBR with depacking format, and (d) ground truth.

Fig. 15. Comparison of Cones (left view): (a) DIBR with MVD format, (b) DIBR with depacking format, (c) edge-based DIBR with depacking format, and (d) ground truth.
Table 4 shows the synthesis results for different DIBR architectures. Because the size of required memory is very large, it is assumed that all required memory buffers are moved to the outside the chip. Thus, the total number of logic gates of the proposed functional unit is 6.34k which is less than other architecture. The maximum frame size of the proposed design is 1920×1088. By ignoring the time of the writes and loads of the memory buffers, the operation frequency is 100 MHz. Frame rate of the proposed architecture is smaller than other architecture because more calculation causes more processing time. Moreover, the results show that the proposed architecture achieves real time application. As shown in Table 4, the proposed architecture has a lower operation frequency and less gate count than the other methods. Since all recent DIBR hardware architectures listed in Table 4 do not specify image quality, the performance comparisons of these architectures become impossible [18-24]. However, the architecture proposed in [25] provides the similar quality as the software implementation (original DIBR algorithm). According to the comparisons in Table 3, the image quality of the proposed architecture is better than that of [25]. Moreover, this paper is the first research, which is based on the design for the 2D-compatible format.

![Fig. 16. Details of Cones (left view): (a) DIBR with MVD format, (b) DIBR with depacking format, (c) edge-based DIBR with depacking format, and (d) ground truth.](image1)

![Fig. 17. Details of Cones (left view): (a) DIBR with MVD format, (b) DIBR with depacking format, (c) edge-based DIBR with depacking format, and (d) ground truth.](image2)

In this architecture, all required memory buffers are outside the chip because of the large memory requirements. To solve this problem, the 2D-compatible format should be adjusted. In the future, some hardware friendly 2D-compatible format such as horizontal 2D-compatible format, which places the color-depth map on both left and right sides, can be proposed to reduce the memory requirements. Some hardware comparisons are also
listed in Table 4. The proposed architecture synthesizes FHD virtual views in real time. FHD of 3DTV is not fully developed so this architecture still has potential for development. Moreover, if the memory requirement problem is solved, the resolution of the proposed architecture can be increased accordingly.

Table 4. Synthesis results and comparison of architectures.

<table>
<thead>
<tr>
<th>Technology</th>
<th>Gate count</th>
<th>Image size</th>
<th>Operation frequency</th>
<th>Frame rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>[18] TSMC 0.18 μm</td>
<td>104.201k</td>
<td>1280 × 1024</td>
<td>71.4 MHz</td>
<td>n.a.</td>
</tr>
<tr>
<td>[19] XC4VSX25 FPGA</td>
<td>8.253k</td>
<td>1920 × 1080</td>
<td>189.7 MHz</td>
<td>90 fps</td>
</tr>
<tr>
<td>[20] UMC 90 nm</td>
<td>32.5k</td>
<td>1920 × 1080</td>
<td>n.a.</td>
<td>30 fps</td>
</tr>
<tr>
<td>[21] XC5VLX330 FPGA</td>
<td>31.923k</td>
<td>1920 × 1080</td>
<td>100 MHz</td>
<td>n.a.</td>
</tr>
<tr>
<td>[22] Stratix II 60 FPGA</td>
<td>2.849k</td>
<td>320 × 240</td>
<td>150 MHz</td>
<td>30 fps</td>
</tr>
<tr>
<td>[23] Vertex 5 FPGA</td>
<td>9.177k</td>
<td>n.a.</td>
<td>151 MHz</td>
<td>n.a.</td>
</tr>
<tr>
<td>[24] TSMC 0.18 μm</td>
<td>162.41k</td>
<td>720 × 576</td>
<td>80 MHz</td>
<td>25 fps</td>
</tr>
<tr>
<td>[25] Cyclone III FPGA</td>
<td>8.554k</td>
<td>1920 × 1080</td>
<td>157.9 MHz</td>
<td>60 fps</td>
</tr>
<tr>
<td>Proposed TSMC 0.18 μm</td>
<td>6.34k</td>
<td>1920 × 1080</td>
<td>100 MHz</td>
<td>20 fps</td>
</tr>
</tbody>
</table>

6. CONCLUSIONS

This study proposed an architecture for realizing 2D-compatible video and depth depacking and DIBR. A hardware-friendly resizing method is applied for reconstructing the color and depth frames with high quality. When the resizing ratio of packing is three fourths, the resizing procedure of depacked video and depth frames is divided into two stages. The Lanczos filter is applied in both stages because of its reduced aliasing, minimal ringing, and high sharpness. In the first stage, the modified Lanczos kernel was proposed with simplified coefficients for up-sampling in hardware. In order to increase the accuracy of down-sampling, more reference pixels are used with the modified Lanczos filter in the down-sampling stage. To reduce the requirement of hardware resources, twelve line buffers are used to store the results of resizing. The four rows of data are output to the DIBR system. The memory requirement is reduced in 3D warping. Simulation results show that the proposed depacking system can reconstruct images with high quality. The PSNR values are 27.6 dB for “Teddy” and 24.42 dB for “Cones” for the color frames. The depacked images have the same quality as that of the original images after DIBR. The boundary of foreground objects is preserved by the proposed simple edge detection method.

REFERENCES


Pin-Chen Kuo (郭品岑) received the B.S. degree in Electrical Engineering from National Dong Hwa University, Hualien, Taiwan, in 2009. She obtained the M.S. degree in Electrical Engineering from National Cheng Kung University, Tainan, Taiwan, in 2011. Her research interests include 3D audio/video rendering and processing. She is currently pursuing Ph.D. degree at National Cheng Kung University in Taiwan.
An-Jie Lin (林安傑) received the B.S. degrees in Electrical Engineering from National Chung Hsing University, Taichung, Taiwan, in 2012. He obtained the M.S. degree in Electrical Engineering from National Cheng Kung University, Tainan, Taiwan, in 2014. His research interest is 3D audio/video rendering.

Bin-Da Liu (劉濬達) received the Ph.D. degree in Electrical Engineering from National Cheng Kung University, Tainan, Taiwan, in 1983. Since 1977 he has been on the faculty of the National Cheng Kung University, where he is currently a Distinguished Professor with the Department of Electrical Engineering. His current research interests include low power circuits, neural network circuits, sensory and biomedical circuits, fuzzy/neural circuits, and audio/video signal processors. He is a Fellow of IEEE.

Jar-Ferr Yang (楊家輝) received the BS degree from the Chung-Yuan Christian University, Taiwan in 1977, and MS degree from the National Taiwan University, Taiwan in 1979, and Ph.D. degree in Electrical Engineering from the University of Minnesota, Minneapolis, USA in 1988. He joined the National Cheng Kung University (NCKU) as an Associate Professor and promoted to full Professor and Distinguished Professor in 1994 and 2004, respectively. His research areas include multimedia processing, coding, and recognition and their applications in smart living services. He is a Fellow of IEEE for his contributions to fast algorithms and efficient realization of video and audio coding.