ARCHITECTURE DESIGN AND ANALYSIS OF IMAGE-BASED RENDERING ENGINE

Jui-Hsin Lai, Chieh-Li Chen, and Shao-Yi Chien

Media IC and System Lab
Graduate Institute of Electronics Engineering and Department of Electrical Engineering
National Taiwan University

ABSTRACT

Image-based rendering (IBR) is a technique to render the video from images, and it provides users to have more interaction and immersive experience in watching a video. In this paper, we integrate the computation of several IBR applications, analyze the bandwidth of memory access, and design an architecture to process the computation of IBR. Experimental results show that the proposed IBR Engine is able to render a video with resolution 720×480 and 30 frames per second, which is 12.7 times faster than a Core2Due 2.83 GHz CPU. For the extensions, IBR Engine can be embedded in the television system and lets viewers enjoy the functions from IBR.

Index Terms— IBR, DIBR, panorama, video rendering

1. INTRODUCTION

Over the last two decades, image-based rendering (IBR) has emerged as one of the most exciting applications of computer vision [1]. For the applications extended from IBR, the layer separation technique was employed to analyze videos contents, and the customized videos were rendered by reintegrating the contents [2], which provided users a novel experience in watching sports videos; 3D reconstruction techniques using multiple views of a scene could create interactive photorealistic experiences [3]; the rendering technique of human-behavior combining background modeling was able to generate the vivid game videos and let users play a tennis game after watching a match video [4].

As the progress in the development of IBR, more functions will be proposed to enrich the viewing experience in the next few years, and more people would be engaged and have a deeper demand for IBR functions. In addition, we think that more broadcasting videos would be presented with IBR functions in the coming years, and the TVs will not only display broadcasting videos but also have the rendering ability to provide IBR functions. Thus, how to design a processor, with low hardware cost, low power consumption and embedded in the TV system, would be a challenge. In this paper, we introduce several IBR applications and analyze their computation, data access and hardware cost. Then the IBR Engine is proposed to process the computation of IBR applications, which can render a video with resolution 720×480 and 30 frames per second (fps). The experimental results show that the average hardware utilization among the supporting applications is increased to 81.2% by using folding architecture, and the bandwidth of memory access is reduced 88.8% by employing the cache mechanism. Comparing to other processors, IBR Engine has the rendering speed 12.7 times faster than a Core2Due 2.83 GHz CPU and 2 times faster than a Quadro2 Pro GPU. The contributions of this work are listed below.

- The primary computation of IBR applications is to calculate the image projection, however different application has different projective transform. We analyze several transforms and propose a reconfigurable architecture to process these computations.
- The heavy bandwidth requirement of memory access is one of the challenges in designing IBR Engine. We use cache mechanism for data reuse and a compression scheme to decrease the data size, and memory bandwidth is reduced 88.8% and 97.7%, respectively.
- To the best of our knowledge, the proposed IBR Engine is the first processor specialized for the computation of IBR, which has higher computational ability and lower hardware cost than the CPU and GPU in the experiments.

The remainder of this paper is structured as follows. Section 2 introduces the supporting IBR applications, and their computation and bandwidth of memory access are analyzed in Section 3. Architecture design of IBR Engine is described in Section 4. Section 5 presents experimental results along with discussions. Finally, the major findings of the paper are summarized in Section 6.

2. SUPPORTING IBR APPLICATIONS

2.1. Panorama

Panorama (PA) is an image stitched by photographs captured from each viewing angle, which gives users an immersive ex-

978-1-61284-350-6/11/$26.00

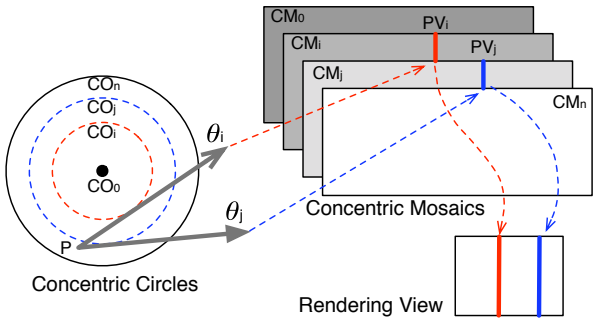

Fig. 1. Illustration of concentric mosaics.

perience by viewing the scene in their demand. The rendering technique is to build a cylindrical model using camera's focal length as the radius, and all the captured photographs are projected on it. Therefore, the primary computation of panorama is the projection transform from cylindrical coordinate to frame coordinate. The equations of cylindrical projection are shown in the fallows.

$$x' = f \arctan \frac{x}{f}, \tag{1}$$

$$y' = f \frac{y}{\sqrt{x^2 + f^2}}, \tag{2}$$

where $(x^{'}, y^{'})$ is the frame coordinate, (x, y) is the cylindrical coordinate of panorama, and f is camera's focal length.

2.2. Concentric Mosaics

Concentric mosaics (CM) is a set of mosaics constructed on concentric circles [5], and each mosaic can be seen as a panorama taken by the camera rotating on the concentric circles as shown in Fig. 1. As for the viewing effects, CM not only provides users an arbitrary viewing angle like the viewing effects of panorama but also gives users abilities to watch a rendering view at a arbitrary standing position inside the concentric circles. The primary computations of CM are the calculation of viewing angle θ and the corresponding slit image PV_i from the mosaics.

2.3. Depth-Image-Based Rendering

Depth-image-based rendering (DIBR) is a key technology in advanced 3D TV system [6]. The principle of 3D viewing effects is to render both video frames from the viewing angles of right-eye and left-eye. Thus, the primary computation of DIBR is to calculate the disparity from the depth map. The equations of DIBR are shown below.

$$x_L = x_c + (\frac{t_x}{x}\frac{f}{Z}), \tag{3}$$

$$x_R = x_c - (\frac{t_x}{x}\frac{f}{Z}), \tag{4}$$

where x_c is the horizontal coordinate of the intermediate view. x_L and x_R are the horizontal coordinates of the left-eye view and right-eye view, respectively. Z is the depth value, f is camera's focal length and t_x is the human-eye distance.

2.4. Tennis Video 2.0

Tennis Video 2.0 (TV2) is a new presentation of sports videos [2], which brings three properties to enhance viewing experience—Structure, Interactivity, and Scalability. Structure allows people to browse game videos and watch highlights immediately. Interactivity provides people with functions to watch enriched game video rendered in real-time. Scalability enables the video to be scalable in the bitstream size with the video quality maintained. The key technique in TV2 is to render the foreground objects on the background scenes. The computation of video rendering is based on the perspective transformation,

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} m_0 & m_1 & m_2 \\ m_3 & m_4 & m_5 \\ m_6 & m_7 & 1 \end{bmatrix} \begin{bmatrix} x_b \\ y_b \\ 1 \end{bmatrix}, \tag{5}$$

where m_i is the perspective parameter, and (x_b, y_b) and $(x/w, y/w)$ are the coordinates in the background scene and rendering view, respectively.

2.5. Tennis Real Play

Tennis Real Play (TRP) is an interactive tennis game system constructed with models extracted from videos of real matches [4], and the key techniques include video-based player rendering and court rendering. The rendering model of TRP is more complex than above applications, which needs to build the 3D structure of background scene and renders the frame by adjusting the intrinsic and extrinsic parameters of the camera. The equation is shown below.

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \begin{bmatrix} f_0 & 0 & x_0 \\ 0 & f_0 & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R} \mid \mathbf{t} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{6}$$

where f_0 is camera's focal length and (x_0, y_0) is the offset coordinate of intrinsic parameters. The rotation matrix $\mathbf{R}$ and translation matrix $\mathbf{t}$ are the extrinsic parameters. By modifying these camera parameters, a virtual 2D scenes can be rendered from 3D structure in any viewing angle.

3. ANALYSIS OF HARDWARE ARCHITECTURE

3.1. Computation Analysis

There are several challenges while implementing and integrating these IBR applications into an uniform computation

Table 1. Average transmission bandwidth (bits/s) of supporting applications under various cache sizes.

	PA	CM	DIBR	TV2	TRP
Memory Bandwidth	6.45×10^9	3.93×10^9	6.62×10^9	5.40×10^9	4.27×10^{10}
Memory Bandwidth with Cache Size 128 bits	2.02×10^9	1.66×10^9	2.57×10^9	2.75×10^9	1.45×10^{10}
Memory Bandwidth with Cache Size 512 bits	7.04×10^8	3.73×10^8	6.43×10^8	8.59×10^8	4.14×10^9
Memory Bandwidth with Cache Size 2048 bits	7.04×10^8	3.54×10^8	6.43×10^8	8.47×10^8	4.12×10^9
The average transmission rate of DDR-200 is 3.36×10^9 bits/s in our system.					

engine. The first step of architecture analysis is to find the critical computation among these IBR applications. We find that the projection transform in (6) of TRP is the heaviest computation, and the equation can be further reduced as the following equations.

$$x = \frac{m_0x' + m_1y' + m_2}{m_6x' + m_7y' + 1}, \tag{7}$$

where m_i are the product of rotation matrix, translation matrix, and camera intrinsic parameters in (6). x' and y' are the algebras of X/Z and Y/Z, respectively. Note that (7) can also support the perspective transform (5) in TV2 and the computation of viewing angle and slit image's position in CM. Next, (7) should have the capability of square root to support the computation of panorama. The denominator of (7) are reformed as the following equations.

$$x = \frac{m_0x' + m_1y' + m_2}{\sqrt[n]{m_6x' + m_7y' + 1}} + OX, \tag{8}$$

where n equals to 2 for the computation of panorama, and n equals to 1 for the rest applications. In addition, the addition terms of OX is designed to support the computation of (3) and (4) in DIBR.

3.2. Bandwidth Analysis of Memory Access

In each IBR application, a number of images are stored in the memory for the rendering materials, and there needs rapidly memory access to read these images. So, one of the challenges in designing IBR engine is how to design an architecture to support the huge requirement of memory bandwidth. We have the experiments to render a video with frame size 720×480 and 30 fps, and a DDR-200 is employed as the off-chip memory, which can support the average transmission rate 3.36×10^9 bits/s. Table 1 shows the bandwidth analyses of the supporting IBR functions. We find that the CM has the lowest memory bandwidth, 3.93×10^9 bits/s, among these applications, but it is still higher than that DDR-200 can support.

Thus, we use two methods to reduce the bandwidth of memory access: cache mechanism and data compression. The cache mechanism in a hardware architecture is effective for the bandwidth reduction because the repeated data can be read from the cache but not memory. In other words, it can reduce the number of memory access. The larger size of cache can increase the data's hit rate and reduce the number of memory access, however the hardware cost also increases. We have the experiments to observe the bandwidth reduction under various cache sizes. The analysis results in Table 1 show that all the memory bandwidths are obviously decreased with the design of cache. We also find that the effects of bandwidth reduction are unapparent when the cache size is larger than 512 bits. Nevertheless, the memory bandwidth of TRP is still higher than that of DDR-200 can support after using the cache mechanism.

The second method employs the compression technique to reduce the bandwidth by transmitting the data with the minor size. A number of previous works had proposed methods for data compression. For the properties of IBR applications, the compression method needs to have the following characteristics: 1. the minimum compression ratio should be 4.32 times to meet the bandwidth requirement of TRP with cache size 128 bits; 2. the ability of data random access and fixed rate of data compression; 3. low decoding latency to reduce the access time. We find that S3 Texture Compression (S3TC), the compression standard mainly used in graphics, can meet above requirements. The idea of S3TC is to break a texture map into 4×4 pixels as a texel, and each pixel in the texel is represented by 3 bits to achieve the compression rate 4.8 times. However, S3TC is a lossy compression method and the PSNR drop of compression results are discussed in Section 5.

4. DESIGN OF HARDWARE ARCHITECTURE

Fig. 2 shows the proposed architecture of IBR Engine consisted of Projection Engine, Prefetch Engine, and Pixel Engine. Note that the First-In-First-Out (FIFO) buffers between each engine store the processing results to the balance the throughputs among different engines.

4.1. Projection Engine

The function of Projection Engine is to support the computation of image projection in (8). Fig. 3 shows the reconfigurable architecture, consisting of 4 multipliers, 5 adders,

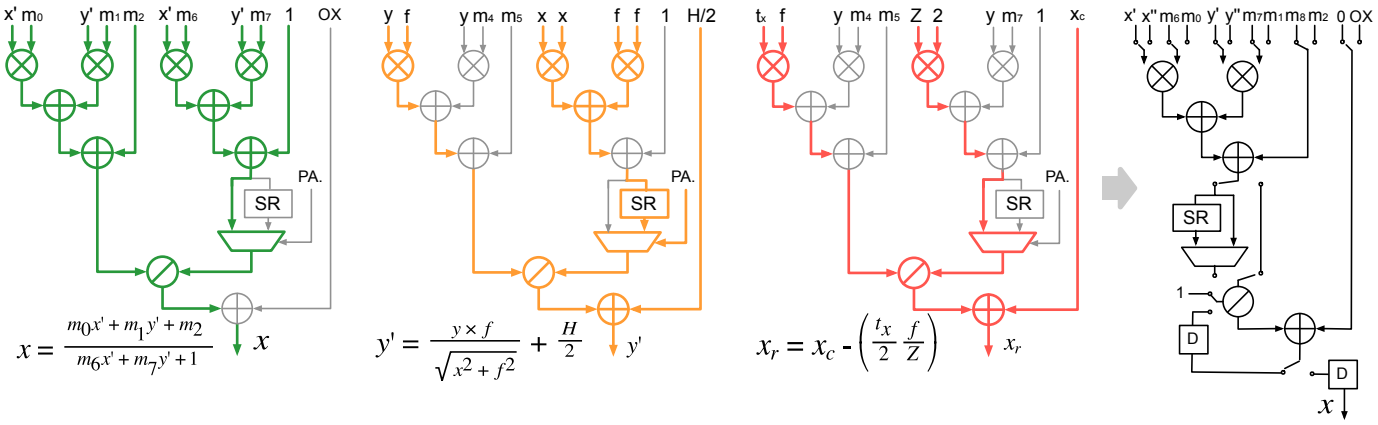

Fig. 3. Reconfigurable structure and folding architecture of Projection Engine.

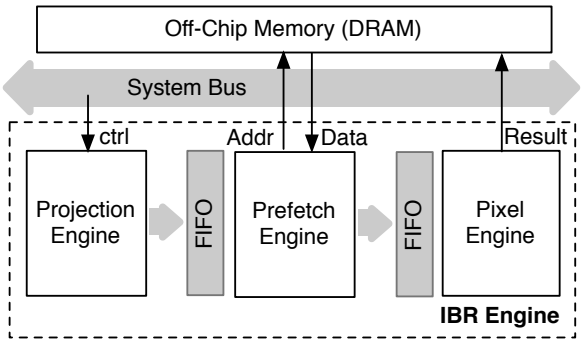

Fig. 2. Proposed architecture of IBR Engine.

1 divider and Look-Up-Tables, to support various processing flows. Note that the trigonometric functions and square root functions are not implemented in the combinational circuits due to high circuit complexity and high power consumption. In the IBR applications, the input values of trigonometric functions and square root functions are on a fixed range, so the Look-Up-Table, like the SR in Fig. 3, is an alternative solution to implement these functions in considering the circuit complexity and power consumption.

To further improve the hardware utilization and decrease hardware cost, we use the folding architecture to reduce the circuit area into a half and also strike a better balance of the throughput between Projection Engine and Prefetch Engine. The folding architecture is shown in the right part of Fig. 3. With the folding architecture, Projection Engine has the reduction in the circuit size but also has a half throughput than the original architecture. The processing results of Projection Engine are sent to the FIFO buffer for further rendering processes. A demo video for the reconfigurable architecture is available on the website[1].

[1]http://media.ee.ntu.edu.tw/larry/vre/

4.2. Prefetch Engine

The function of Prefetch Engine is to read the rendering materials from the memory after receiving the projection coordinates from Projection Engine. As shown in Fig. 2, Prefetch Engine would send an address to the off-chip memory for the data access, and the data would be transferred to IBR Engine after several cycles. The architecture of prefetch Engine includes cache mechanism and compression scheme as the mentions in Section 3.2. From the IBR applications, we find that the earlier the data stored in a cache, the less chance it will be used in the future. Thus, a FIFO mechanism, the earliest data stored in the cache would be first removed when the latest data inputs, is employed in designing the cache. According to the analysis results in Section 3.2, a 128-bits cache is designed in the Prefetch Engine in considering the hardware cost and the bandwidth reduction rate.

For the design of compression scheme, S3TC mentioned in Section 3.2 is adapted to further reduce the memory bandwidth. The primary computation of S3TC is to decode the 16 pixels in a texel, and it can be seen that each pixel value is the interpolation result according to the index between the maximum label and minimum label in a texel. To increase the throughputs of Prefetch Engine, we use 16 decoding units for the parallel computation.

4.3. Pixel Engine

The function of Pixel Engine is to calculate the pixel values on the rendering frame. For the algorithm using backward warping scheme like PA, CM, TV2 and TRP, each pixel value on the rendering frame should be interpolated with 4 pixel values on the projection positions. The computation of pixel interpolation adopts the bilinear interpolation. For forward warping schemes, such as DIBR, the reference pixel will be mapped to a pixel position according to the pixel's disparity. However, some holes are on the rendering frame because of the differ-

ent sampling resolution between the input and output images. To perform occlusion detection, we put a buffer with 64 bits in Pixel Engine. Because pixels only move in horizontal direction and the new coming pixel will always be at the right side of current pixel in left-eye view in DIBR, we can easily check whether current rendering block contains holes or not. If there are holes, we can simply use neighbor's pixel value to interpolate the holes. For multi-layer-based rendering, such as TRP, a weighting computation is also included in Pixel Engine to support the visual effects of alpha blending.

5. EXPERIMENTAL RESULTS

5.1. Specification and FPGA Implementation

Table 2 shows the specification and implementation details of IBR Engine. Here we use TSMC 0.18μm process technology for circuit synthesis and implement the design on the FPGA, XilinX Virtex5, for the function verification. Note that the size of non-combinational circuit is larger than the combinational circuit due to the Look-Up-Tables in Projection Engine, cache in Prefetch Engine and FIFO buffers between engines. The design runs at 100 MHz and renders 30 fps with video resolution 720×480 to achieve real-time requirement.

Table 2. Specification and implementation of IBR Engine.

	Specification	Design on FPGA
Technology	TSMC18	XilinX Virtex5
Clock Rate	100 MHz	63.2 MHz
Gate Count	89.7 K Gates	12.7 K Slices
Memory	499.7 K Gates	47.0 K Slices
Power	61.29 mW	Not Available
Rendering Size	720×480	720×480
Frame Rate	30.3 fps	21.9 fps

5.2. Analysis of Hardware Utilization

Hardware utilization is one of the indexes to reveal the design quality. The analysis results show that Prefetch Engine and Pixel Engine have very high utilization rates in all supporting IBR applications, but Projection Engine has the relative lower rate. The utilization rates of IBR Engine in each application are listed in Table 3, and the average rate is 74.9%. In order to increase the hardware utilization, we employ the folding architecture proposed in Section 4.1. The circuit size of Projection Engine is reduced from 49920 gate counts to 33024 gate counts, and the average hardware utilization of IBR Engine increases to 81.2%. Note that the throughput of Projection Engine reduces to a half by using folding architecture, but the system throughput has no reduction due to the critical throughput in Prefetch Engine. With the design of FIFO buffers in Fig. 2, the bubble cycles from the variant throughputs of Prefetch Engine can be removed, and system throughputs are increased 27.4%

5.3. Analysis of Bandwidth Reduction

The bandwidth of memory access is effectively reduced by using a 128-bits cache. Table 3 shows the reduction rates of each application and the average rate is 60.6%. As the mentions in Section 3.2, the bandwidth of TRP is still higher than the maximum bandwidth that DDR-200 can support. To further reduce the bandwidth, S3TC is used to compress the data and achieve much lower bandwidth, and the average reduction rate is 91.8%. However, S3TC is a lossy encoder and the video quality would be decreased. Table 3 reveals that the average PSNR of compressed video is 36.8. In our opinion, it is a trade-off between memory bandwidth and video quality in using data compression. However, the data compression can be removed from the IBR Engine if using an off-chip memory with supporting higher bandwidth like DDR2-200.

Table 4. Rendering abilities (fps) of CPU and IBR Engine.

	PA	CM	DIBR	TV2	TRP	Av.
CPU	4.0	3.4	2.4	2.8	2.1	2.9
IBR Engine	39.0	49.0	30.6	34.6	30.3	36.7
Ratio	9.8	14.4	12.8	12.4	14.4	12.7

5.4. Comparison

Here we use Intel ® Core™ 2 Duo, running at 2.83 GHz, as the CPU platform to perform IBR applications with rendering size 720×480. Table 4 shows the rendering results and performances between CPU and IBR Engine. The average rendering frame rate of CPU is only 2.9 fps because IBR applications need complex computation in the projection transform and heavy bandwidth requirement of memory access. Comparing to the CPU architecture, IBR Engine is specific for the computation of projection transform and the architecture of memory access, and therefore it can achieve the average rendering rate 36.7 fps. In other words, the computational ability of IBR Engine is 12.7 times than that of CPU in running these IBR applications.

As the comparison of previous works, Yang et al. had used graphics hardwares, two NVIDIA GPU: Quadro2 Pro and GeForce3, to process the scene construction [7], which is one of IBR applications. To compare the performance, IBR Engine processes the computation of 3D warping, the same experiments in [7]. The rendering abilities of Yang's work and IBR Engine under different rendering sizes are shown in Table 5. We find that IBR Engine has the obviously advantage of rendering ability in the lower rendering size, and it still has 2.2 times performance than Quadro2 Pro under frame size

Table 3. Analysis of hardware utilization, bandwidth reduction and quality of rendered video.

	PA	CM	DIBR	TV2	TRP	Av.
Hardware Utilization (%)	73.4	68.1	59.7	86.2	87.3	74.9
Hardware Utilization with Folding Architecture (%)	84.5	79.2	73.3	84.0	85.0	81.2
Bandwidth Reduction with Cache 128 Bits (%)	68.7	57.8	61.2	49.1	66.0	60.6
Bandwidth Reduction with Cache 128 Bits and S3TC (%)	93.5	91.2	91.9	89.4	92.9	91.8
Quality of Rendered Video after Compression (PSNR)	37.6	37.7	37.8	35.8	35.3	36.8

Table 5. Rendering abilities (fps) of GPU and IBR Engine.

Rendering Size	128×128	256×256	512×512
Quadro2 Pro	62.5	32.3	12.2
IBR Engine	434.8	108.7	27.2
Ratio	7.0	3.4	2.2
GeForce3	25.0	18.2	6.4
IBR Engine	434.8	108.7	27.2
Ratio	17.4	6.0	4.3

512×512. As the comparison of circuit size, the gate counts of Quadro2 Pro and GeForce3 are 3.13M and 7.13M, respectively. However, IBR Engine has only 0.59M gate counts. It shows that the proposed IBR Engine has higher computation ability and smaller circuit size than Quadro2 Pro and GeForce3 in processing the IBR applications.

6. CONCLUSIONS AND EXTENSIONS

To the best of our knowledge, we design the first hardware architecture specific for the computation of IBR and implement the circuit on the FPGA. Because the computations of IBR are different from various applications, we employ the reconfigurable architecture to perform various rendering schemes. Next, a hardware sharing is introduced to increase the hardware utilization. To solve the heavy requirement of memory bandwidth, we insert cache mechanism to reduce the number of memory access and use data compression to decrease the size of data transmission. The experiments show that the proposed IBR Engine can support various IBR applications with outstanding rendering abilities and low hardware cost.

Although only five IBR applications are introduced in this paper, the IBR Engine is able to support the 3D rendering model and more applications can be performed on it, like photo tourism [3] and video texture [8]. However, there are still some limitations of the proposed IBR Engine, which can not support the application with computation complexity higher than 3D rendering model, such as 4D light field rendering [9]. The possible extensions of IBR Engine are to support more IBR applications and improve the rendering resolution to high definition with 60 fps.

7. REFERENCES

[1] H.-Y Shum, S.-C. Chan, and S. B. Kang, *Image-Based Rendering*, Springer, 2007.

[2] J.-H. Lai, C.-L. Chen, C.-C. Kao, and S.-Y. Chien, "Tennis video 2.0: A new presentation of sports videos with content separation and rendering," *Journal of Visual Communication and Image Representation*, vol. 22, no. 3, pp. 271–283, 2011.

[3] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism: Exploring photo collections in 3d," *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 835–846, 2006.

[4] J.-H. Lai, P.-C Wu, C.-L. Chen, C.C. Kao, and S.-Y. Chien, "Tennis real play," in *IEEE International Conference on Consumer Electronics*, 2011, pp. 275–276.

[5] H.-Y. Shum, K.-T. Ng, and S.-C. Chan, "A virtual reality system using the concentric mosaic: Construction, rendering, and data compression," *IEEE Transactions on Multimedia*, vol. 7, no. 1, pp. 85–95, 2005.

[6] C. Fehn, "Depth-image-based rendering (dibr), compression and transmission for a new approach on 3d-tv," in *Proceedings of SPIE Stereoscopic Displays and Virtual Reality Systems XI*, 2004, pp. 93–104.

[7] R. Yang, G. Welch, and G. Bishop, "Real-time consensus-based scene reconstruction using commodity graphics hardware," *Computer Graphics Forum*, vol. 22, no. 2, pp. 207–216, 2003.

[8] A. Schodl, R. Szeliski, D. H. Salesin, and I. Essa, "Video textures," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 2000, pp. 489–498.

[9] M. Levoy and P. Hanrahan, "Light field rendering," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996, pp. 31–42.