Fast Stereoscopic Rendering on Mobile Ray Tracing GPU for Virtual Reality Applications

Won-Jong Lee
Samsung Electronics

Seok Joong Hwang
now at SKT

Youngsam Shin
Samsung Electronics

Jeong-Joon Yoo
Samsung Electronics

Soojung Ryu
Samsung Electronics

***Abstract*—In this paper, we present a stereoscopic rendering based on a mobile ray tracing GPU. Adopting an existing algorithm to new mobile GPUs specialized for ray tracing enables two high performance techniques such as reprojection and tile-based rendering. Experimental results show that our implementation can be a versatile solution for future virtual reality applications, as it achieves up to 1.64 times better performance and 20% better energy efficiency compare with state-of-the-art solutions. To the best of our knowledge, our solution is the first to demonstrate mobile virtual reality based on real-time ray tracing.**

I. Introduction

As the recent mobile market is growing continuously, real-time ray tracing has attracted considerable attention for graphics applications. In particular, ray tracing is being considered a potential rendering algorithm that will facilitate a immersive experience for Head Mount Display (HMD)-based virtual reality (VR) applications because it can provide physical light effects such as reflection, refraction, and shadow without distortion [1, 2]. However, current approaches are being realized in PC environments because mobile computing has limited computing power, memory bandwidth, and energy budget. Thus, recently various hardware-based ray tracing solutions such as hardwaresoftware hybrids [3], and GPU IPs [4], are being proposed to resolve these problems.

In this paper, we present the first demonstration of stereoscopic rendering utilizing mobile ray tracing GPU for VR applications. For the sake of real-time rendering in mobile environment, we optimized an existing algorithm for a mobile ray tracing GPU. Our system has two key features. First, we adopt a *reprojection* method that can significantly reduce the computing costs. Unlike the rasterization approach, ray tracing can easily reuse the results of the left (L-) image and apply it for the right (R-) image [5], which can reduce the number of rays fed into hardware accelerator. Second, we utilize a *tile-based rendering* which allows most of the reprojection operations to be performed using the on-chip internal memory without having to access the external memory. The experimental results show that it achieves up to 1.64 times better performance and 20% better energy efficiency, compared with the conventional solution by substantially reducing the computing cost of the dedicated hardware and minimizing the memory operations.

II. Related Work

Due to the performance gap between the requirements and current speed exists, interest in dedicated ray tracing hardware for the mobile domain has naturally been renewed [6]. However, this architecture could not provide a sufficiently high performance (<30 Mrays/s) for real-time ray tracing. SGRT (Samsung GPU based on Ray Tracing) [3] is the first mobile GPU based on ray tracing, which combines the advantages of programmable DSP cores called SRP (Samsung Reconfigurable Processor) [7] and a dedicated hardware called T&I unit. Recently, a commercial mobile GPU featured ray-tracing, called PowerVR GR6500 [4], has been announced.

Fig. 1. Stereoscopic rendering with test scenes: Teapot (15K triangles), Chess (42K), BMW (55K), Chemical Lab. (98K), Music box (106K), and Provence (600K). Except the yellow pixels (indicate *bad pixels*), the most of the pixels (91.54%) in the right image can be shaded with the results of the left image.

Reprojection in ray tracing based stereoscopic rendering has been important topic in past research. Badt [5] firstly proposed the reprojection idea to apply the results of the L-image to the R-image and a solution to fix the error caused by the different visibility between the L- and the R-image. But, this algorithm was targeted only for desktop CPU and could not render in real-time. Recently, the increasing demands for high quality introduces the ray tracing based VR applications [1, 2]. However, these solutions are all implemented in PC GPUs. VR based on mobile ray tracing has not yet been reported.

III. Stereoscopic Rendering on SGRT

In this paper, we present stereoscopic rendering based on mobile ray tracing GPU (Figure 1). The overview of our implementation is shown in Figure 2. As a target GPU, we utilize SGRT [3] which efficiently supports flexible real-time ray tracing by combining the advantages of the hardware and the software. Two key features of our solution are as follows.

Reprojection: we aggressively utilize the reprojection approach [5] to reduce the computing cost by adopting existing algorithm. Reprojection is an efficient way to accelerate stereoscopic rendering by exploiting the coherence between the L- and the R-images. During the ray tracing of the L-image, the intermediate data of ray-object intersection points are reprojected into the space of the R-image. Then, this information can be directly reused for shading in the R-image, which can eventually avoid the full ray tracing. Due to the disparity of the viewpoints of the L- and the R-image, the reprojection errors called *bad pixels* might be occurred, but these are appropriately detected with a simple test and are processed by normal ray tracing. With the SGRTs full programmability, we could implement the additional kernels (*reprojection*, *error detection*, and *reusing*) with the inspector-executor model to reduce the side-effects of branch divergence and maximize

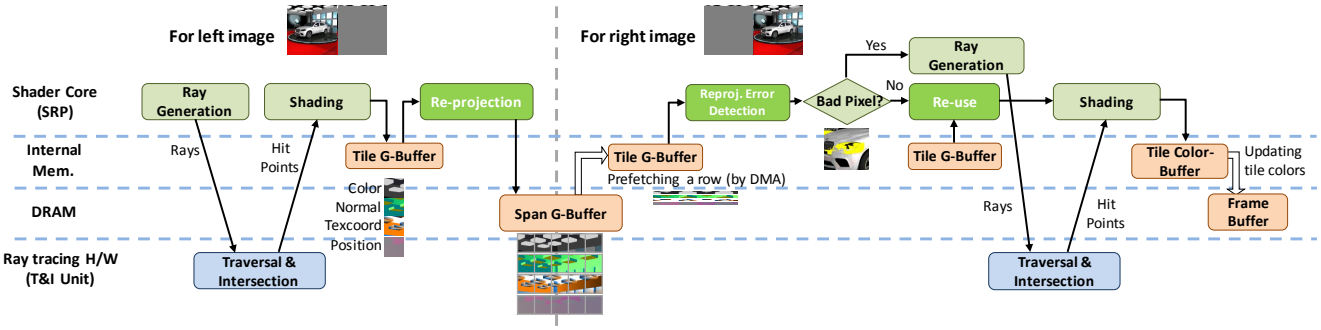

Fig. 2. Overview of our stereoscopic rendering on mobile ray tracing GPU.

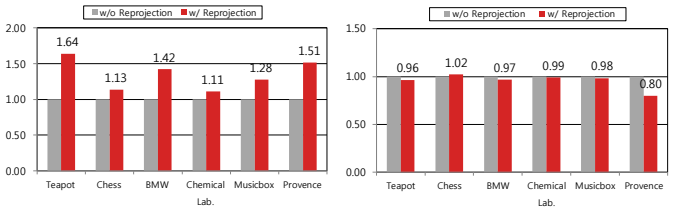

Fig. 3. Relative performance (left) and energy consumption (right).

the efficiency of stream processing. Consequently, with the minimal overheads (∼10%) in software, we could significantly reduce the computing cost for the hardware T&I units (∼68%).

Tile-based ray tracing: Unlike the conventional GPU ray tracers, we employed the tile-based approach in order to reduce the memory bandwidth. Our rendering is similar to *deferred rendering* frequently used in hybrid rasterizer-ray tracing [4]. In the first pass, the rasterizer stores the geometry information (color, normal, texture coordinates, and position) for the primary rays to G-buffer. In the second pass, the ray tracer reads the G-buffer, generates the secondary rays and traces them. The G-buffer should be located in the external DRAM due to its excessive size, which can cause to bandwidth problem. Adopting the reprojection algorithm makes the same problem because the G-buffer for the intersection points of the L-image should be used for the R-image. We resolve this problem by utilizing tile-based approach widely utilized in modern mobile GPUs. By conducting ray tracing per-tile basis rather than per-screen, the G-buffer can be fit into the internal memory, which allows the kernels (reprojection and reusing) to be performed using the on-chip internal SRAM without having to access the external DRAM. Exceptionally, the error detection kernel accesses the DRAM because this should test the whole pixels in the same row of the current pixel. Before entering this kernel, the G-buffer data for a row of the current pixel is prefetched by DMA (Direct Memory Access) function, which can effectively hide the latency of the DRAM access.

IV. Experimental Results

To verify and evaluate how our implementation can reduce the computing cost of the T&I unit, we utilized the cycle accurate simulator of the SGRT integrated with the energy model. We used the energy and power model of [8] which utilized a custom model based on the database built with the power values per component from Synopsys PrimTime PX with SAMSUNG 14nm LPP process technology. The hardware configuration of the T&I is the same as the T&I unit 2.0 [9]. As the test scenes, we used five datasets (Figure 1): Teapot (15K triangles), Chess (42K), BMW (55K), Chemical Lab. (98K), Music box (106K), and Provence (600K). Test scenes were all rendered at 2048×1024 resolution with enough secondary ray effects. We compared with the standard reference; ray tracing without reprojection in the same hardware platform. Figure 1 shows the results of the stereoscopic rendering for six test scenes. The pixels, marked as yellow, in the R-image indicates the bad pixels. We could find that most of the pixels (91.54% in average) in L-image could be reused as shown in the figure.

Figure 3 shows the relative performance and energy consumption for the T&I unit. Overall, it achieved up to 1.64 times better performance compared with the reference platform. This is because it can substantially reduce the computing cost of the T&I unit. In terms of the absolute performance, we could obtain 131.3, 14.4, 18.2, 8.6, 28.9 and 44.5 fps for each test scene, respectively. We believe that this can be real-time or interactive throughputs though there are some variants among the test scenes. Regarding energy consumption, our implementation could reduce up to 20% because it could cut the workloads in the hardware. Obviously, this is another proof of the superiority of our implementation.

V. Conclusion

In this paper, we present a solution to realize ray tracing based stereoscopic rendering utilizing a mobile ray tracing GPU. With the combination of the reprojection and tile-based ray tracing, our approach could be a versatile solution for future VR applications, as it achieves up to 1.64 times better performance and 20% better energy efficiency, compared with the state-of-the-art solution.

References

[1] T. Harada, "Foveated ray tracing for VR on multiple GPUs," *ACM SIGGRAPH Asia 2014, Course – GPU Compute for Graphics*, 2014.

[2] P. Miller, "NVIDIA brings interactive photorealism to VR with Iray," https://blogs.nvidia.com/blog/2016/04/05/vr-with-iray, 2016.

[3] W.-J. Lee, Y. Shin, J. Lee, J.-W. Kim, J.-H. Nah, S.-Y. Jung, S.-H. Lee, H.-S. Park, and T.-D. Han, "SGRT: A mobile GPU architecture for real-time ray tracing," *ACM High Performance Graphics (HPG)* , pp. 109–119, 2013.

[4] J. McCombe, "New techniques made possible by PowerVR ray tracing hardware," *Game Developer Conference (GDC), technical talk*, 2014.

[5] J. S. Badt, "Two algorithms for taking advantage of temporal coherence in ray tracing," *The Visual Computer*, vol. 4, no. 3, pp. 123–132, 1988.

[6] H.-Y. Kim, Y.-J. Kim, J.-H. Oh, and L.-S. Kim, "A reconfigurable SIMT processor for mobile ray tracing with contention reduction in shared memory," *IEEE Transactions on Circuits and Systems (TCS)*, vol. 1, no. 99, pp. 1-13, 2012.

[7] W.-J. Lee, S.-O. Woo, K.-T. Kwon, S.-J. Son, K.-J. Min, C.-H. Lee, K.-J. Jang, C.-M. Park, S.-Y. Jung, and S.-H. Lee, "A scalable GPU architecture based on dynamically embedded reconfigurable processor," *ACM High Performance Graphics (HPG), poster*, 2011.

[8] W.-J. Lee, Y. Shin, S. J. Hwang, S. Kang, J.-J. Yoo, and S. Ryu, "Reorder Buffer: An energy-efficient multithreading architecture for hardware MIMD ray traversal," *ACM High Performance Graphics (HPG)*, pp. 21-32, 2015.

[9] J. Lee, W.-J. Lee, Y. Shin, S. J. Hwang, S. Ryu, and J. Kim, "Two-AABB traversal for mobile real-time ray tracing," *ACM SIGGRAPH Asia 2014, Simposium on Mobile Graphics and Interactive Applications (MGIA)*, article no. 14, 2014.