Phase-Aligned Foveated Rendering for Virtual Reality Headsets

Eric Turner* Haomiao Jiang Damien Saint-Macary Behnam Bastani

Google Inc.

Abstract

We propose a novel method of foveated rendering for virtual reality, targeting head-mounted displays with large fields of view or high pixel densities. Our foveation method removes motion-induced flicker in the periphery by aligning the rendered pixel grid to the virtual scene content during rasterization and upsampling. This method dramatically reduces detectability of motion artifacts in the periphery without complex interpolation or anti-aliasing algorithms.

Index Terms: H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—Artificial, augmented, and virtual realities

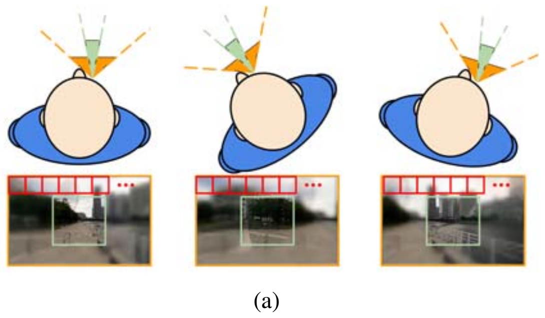
1 Introduction

Immersive VR is trending towards wider fields of view and greater pixel densities. As head-mounted displays move closer to these impressive specifications, the total number of pixels required becomes enormous. The motivation of foveated rendering in VR is to drive displays of these sizes without performing the full render stack on each pixel [4, 7]. Full-resolution content is rendered at the center of the screen or, if eyetracking is available, at the current gaze. The periphery is generated at a sparse resolution, which is then interpolated or back-filled [3, 4, 10]. Computation is saved each frame by reducing the number of pixels calculated by fragment shaders. The amount of foveation must become more aggressive as headsets become more advanced and more immersive. Any existing aliasing or flickering artifacts become more apparent to a user, requiring additional post-processing.

In this paper, we propose foveated rendering that reduces detectability of aliasing in the periphery. Rather than applying complex filtering after rendering, we instead rely on proper angular alignment of the render frustums to minimize frame-to-frame flicker artifacts. Although static aliasing artifacts within each frame are still present, temporal aliasing is greatly reduced. We have found that the temporal flickering is the larger cause of detectability of foveated rendering and its removal allows for more aggressive foveation [5]. Phase-alignment can be performed alongside any upsampling technique. More simplistic upsampling methods, such as nearest neighbor, become viable since their substantial aliasing artifacts are less perceptible in the periphery by not contributing to dynamic flicker under motion [5]. Phase-alignment allows for computationally simpler interpolation techniques that preserve local contrast.

2 Phase-Alignment Method

Under traditional foveation, both the low-acuity and high-acuity regions are updated with head-tracking information [1, 8]. The low-acuity regions are rasterized and upsampled in the latest display coordinate frame, shown in Fig. 1a. Any aliasing artifacts due to upsampling are aligned to display coordinates. Since the display moves in relation to the virtual world content, aliasing artifacts move as well, causing perceivable flickering from frame to frame. The


(a)

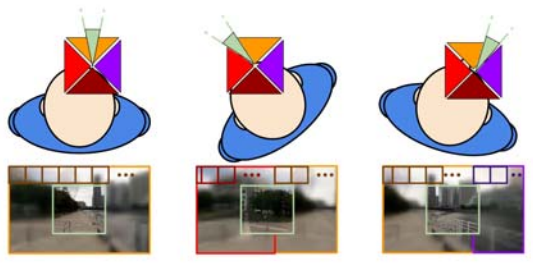

(b)

Figure 1: (a) Traditional foveation with the high-acuity region (green) and wider low-acuity region (orange) rendered aligned to head coordinates; (b) Phase-aligned foveation with the high-acuity region (green) rendered in the display coordinates and low-acuity regions (red, orange, purple, brown) rotationally fixed to world coordinates during rendering.

low-acuity pixels are always upsampled at the same phase-offset on the display, regardless of head rotation, shown by red squares at the top of each image. These squares showcase how each pixel changes as the user rotates, causing the pixel color to shift and flicker.

In phase-aligned foveated rendering, we enforce low-acuity regions to be rotationally world-aligned, which are then reprojected and resampled onto the final display surface. The phase offset between the low-acuity pixels and the native-resolution pixels is dynamic from frame-to-frame, ensuring each low-acuity pixel is aligned with the virtual world rather than the display.

Fig. 1b shows our phase-aligned method. The high-acuity region matches the rotational movement of the head, but the low-acuity regions are rotationally fixed to world coordinates. Multiple low-acuity regions are now necessary. As one low-acuity region moves out of the display frame, another fills the area. The low-acuity pixels, shown as squares at the top of the rendering, are always phase-aligned with the world content rather than the display. As a result, no flickering artifacts are introduced due to head rotation.

3 Implementation and Results

In a 3D system, a total of six low-acuity regions are allocated to cover all faces of a cube, but only a subset are needed each frame. As shown in Fig. 2, prior to rendering we compute which regions are visible at the display's current orientation. Typically only two

*email:elturner@google.com

2018 IEEE Conference on Virtual Reality and 3D User Interfaces
18-22 March, Reutlingen, Germany
978-1-5386-3365-6/18/$31.00 ©2018 IEEE

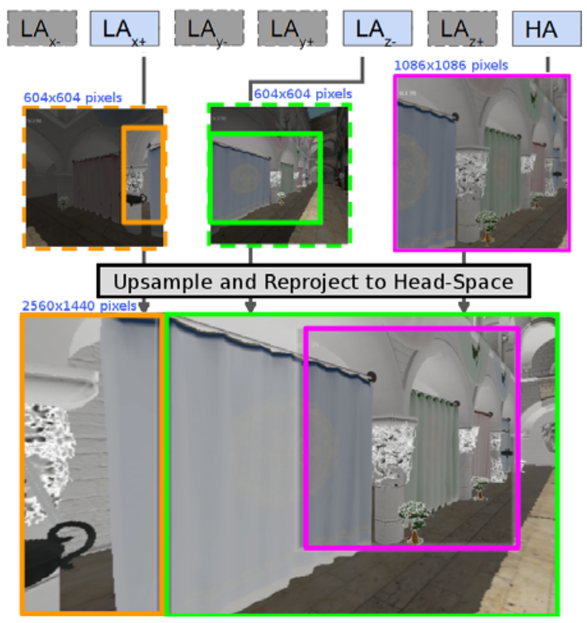


Figure 2: Six Low-Acuity (LA) regions and one High-Acuity (HA) regions are preallocated per eye. Only a subset are selected each frame (colored in blue). Each selected viewing frustum is computed (dashed outline) and its content is rendered to the subimage that is visible on the final display (solid outline). The result is reprojected and upsampled into the final image for each eye.

or three low-acuity regions are used on any given frame, depending on the output field of view of the headset. Fig. 2 shows an example frame using two low-acuity regions. For each surface used, we find the portion overlapping the output display and only render to that area. The remaining area is masked out via depth-culling, short-circuiting the fragment shader and reduces draw cost. As a result, the net number of pixels used in the low-acuity regions is approximately equal for phase-aligned foveation and traditional foveation. In Fig. 2, the low-acuity buffers are shown as 604×604 pixels, but only a fraction of these pixels are used.

After rasterization, these surfaces are upsampled, reprojected, and composited onto a final full-resolution buffer. This buffer is sent through lens distortion correction and presented to the display. This process is repeated for both eyes each frame. By rendering this scene in a foveated manner, a total of 1.5 million pixels are computed per eye across three surfaces in the shown example, whereas a full resolution rendering would require 3.7 million pixels on a single render surface for each eye.

Increasing the number of required framebuffer objects adds an overhead cost. Each object in the scene duplicates its draw calls for each framebuffer used, increasing computation on both the CPU and GPU. Fig. 3 shows how much GPU time is spent per frame per eye in rendering and composition of a typical VR scene of a 3D mesh [2]. These measurements are for a VR headset with a resolution of 2560×1440 pixels per eye, a horizontal binocular field of view of 140 degrees, and full positional tracking, driven by a desktop PC using a GTX-980 GPU. These measurements use an upsampling factor of 10×10 in the periphery with a high-acuity region covering $\pm 25°$ at the optical center.

Default rendering spends the majority of processing on rasterization, with the remaining GPU time allocated for software lens distortion correction, totaling 3.4 ms per eye. Traditional foveation with two regions reduces the total GPU time per frame to 1.9 ms. Our phase-alignment method requires 2.0 ms on average per eye per frame for the same level of upsampling.

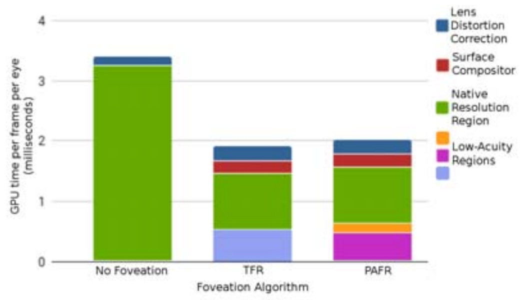


Figure 3: GPU performance of Phase-Aligned Foveated Rendering (PAFR) algorithm and Traditional Foveated Rendering (TFR). Each column represents the breakdown of GPU time per frame per eye.

Phase-alignment costs slightly more than traditional foveation, due to additional render targets, but the visual quality improvements are dramatic [5]. More aggressive foveation can be used while preserving the same perceived quality, yielding net savings. Further, as future head-mounted displays use wider fields of view, multiple frustums are required regardless to generate renderings that exceed 180°, in which case phase-alignment is optimal.

4 Limitations and Conclusion

Phase-alignment is most useful for VR headsets with a wide field of view or high pixel density display. These systems require the most pixels to be rendered and are bottlenecked by the fragment shader, maximizing the benefit of foveation. Although phase-alignment adds some overhead compared to traditional foveated rendering for the same pixel count, due to additional render targets and composition, it allows these headsets to foveate more aggressively [5, 9].

Phase-alignment is limited to providing flicker removal for rotational movement only. If a user moves translationally, the world-aligned cube is recentered on their eye position. Flickering is reduced, but not completely removed in this case. Flickering caused by animation in the scene is still unaffected, so we recommend applying additional smoothing techniques such as Temporal Anti-Aliasing (TAA) to compensate for these effects [6].

References

[1] B. Bastani, E. Turner, C. Vieri, H. Jiang, B. Funt, and N. Balram. Foveated pipeline for ar/vr head-mounted displays. *Information Display*, 33(6), Nov. 2017.

[2] Crytek. Sponza sample scene. `https://www.cryengine.com/marketplace/product/sponza-sample-scene`, 2016. [Online; accessed 18-Oct-2017].

[3] M. Fujita and T. Harada. Foveated real-time ray tracing for virtual reality headset. *Light Transport Entertainment Research*, 2014.

[4] B. Guenter, M. Finch, S. Drucker, D. Tan, and J. Snyder. Foveated 3d graphics. *ACM Transactions on Graphics*, 36(6), 2012.

[5] D. Hoffman, Z. Meraz, and E. Turner. Sensitivity to peripheral artifacts in vr display systems. *SID Symposium Digest Technical Paper*, 2018.

[6] B. Karis. High-quality temporal supersampling. *ACM SIGGRAPH 2014 Courses*, (10), 2014.

[7] Y. S. Pai, B. Tag, B. Outram, N. Vontin, K. Sugiura, and K. Kunze. Gazesim: Simulating foveated rendering using depth in eye gaze for vr. *SIGGRAPH '16*, July 2016.

[8] A. Patney, M. Salvi, J. Kim, A. Kaplanyan, C. Wyman, N. Benty, D. Luebke, and A. Lefohn. Towards foveated rendering for gaze-tracked virtual reality. *ACM Transactions on Graphics*, 35(179), 2016.

[9] H. Strasburger, I. Rentschler, and M. Juttner. Peripheral vision and pattern recognition: A review. *Journal of Vision*, 11, May 2011.

[10] M. Weier, T. Roth, E. Kruijff, A. Hinkenjann, A. Perard-Gayor, P. Slusallek, and Y. Li. Foveated real-time ray tracing for head-mounted displays. *Eurographics*, 35(7), 2016.