Efficient Peripheral Flicker Reduction for Foveated Rendering in Mobile VR Systems

Haomiao Jiang* Facebook AR/VR
Tianxin Ning† Facebook AR/VR
Behnam Bastani‡ Facebook AR/VR

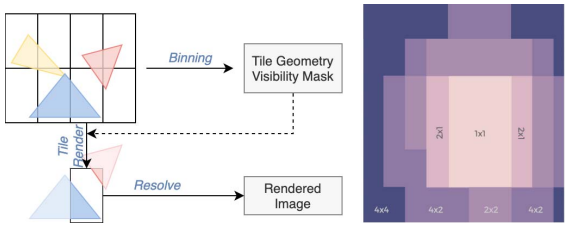

Figure 1: (Left) Tile-based rendering framework. (Right) The varying shading rate across tiles allows downsampling periphery.

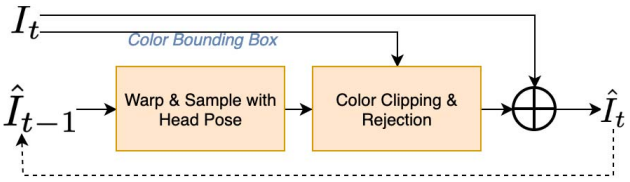

Figure 2: Flowchart for the proposed efficient flicker control algorithm.

Abstract

We propose an efficient algorithm to largely mitigate the flickering artifacts induced by tile based foveated rendering, which can be executed in sub-milliseconds on mobile VR platform. The algorithm can be easily integrated to VR applications without intrusively affecting the development process. We demonstrate the speed and visual quality of the proposed algorithm on the Oculus Quest.

Index Terms: Computing methodologies—Computer graphics—Graphics systems and interfaces—Virtual reality

1 Introduction

In virtual reality system, high resolution, high field of view and high refresh rate rendering is desired to deliver an immersive experience. However, due to the limited compute power, it is challenging for mobile VR systems to achieve high image fidelity while meets the requirements for resolution and refresh rate.

To achieve good image quality with limited compute resources, foveated rendering techniques are developed to take advantage of acuity difference between fovea and periphery vision [10] and spend most rendering power on the important regions [6] [1]. One of the most widely used foveated rendering techniques for mobile VR system is tile-based foveated rendering, in which the shading rate is configured to be at different levels to each render tile. In this way, GPU render time and power can be saved by using lower sampling frequency in periphery regions (Figure 1).

Lower sampling and shading rate can induce aliasing artifacts, which usually gets viewed as jagged edges [7]. With head motion and tracking noise, the aliasing artifacts become flickering, which is very noticeable in both fovea and periphery region [9].

2 Related Work

A handful of algorithms have been proposed to reduce aliasing artifacts in rendered content. Spatial anti-aliasing methods, e.g. multi-sample anti-aliasing (MSAA), generates more samples in the aliasing region to avoid under-sampling issue. Other spatial / screen space anti-aliasing methods, e.g. AXAA [5], SMAA [3] etc., are developed to perform edge detection and anisophical filtering, which can be expensive in mobile VR systems. On the other hand, temporal anti-aliasing methods have been developed to use the correlations from previous frames for anti-aliasing [4]. In most cases, accurate motion vectors are used to determine the correspondence between current and previous frame.

Besides anti-aliasing, there are also algorithms developed to prevent spatial aliasing artifact to induce flickering with motion. For example, [8] propose to quantize the head rotation so that aliasing pattern does not change with head rotation.

3 Proposed Method

The proposed flicker control algorithm uses the information in corresponding pixels in previously reconstructed frames to stabilize the current frame (Figure 2), i.e.

$$\hat{I}_t[x,y] = I_t[x,y] + (1-\alpha[x,y])(\hat{I}_{t-1}[x',y'] - I_t[x,y])f(I_t,\hat{I}_t-1)[x,y]$$

Here (x',y') indicates the corresponding pixel position of (x,y) in previous frame and f represents the rejection logic to test if $It[x,y]$ and $It-1[x',y']$ are actually corresponding points from the same object. Computing the corresponding pixel position x', y' in previous frame usually requires accurate motion vectors. However, computing and storing motion vectors for all pixels and objects involve extra compute and integration cost for most applications. Instead of requiring very accurate motion vectors, we use head pose information to compute the sampling position in previous frame, i.e.

$$[x',y',z'] = LP_{t-1}P_t^{-1}L^{-1}[x,y,1]$$

L is the transform matrix derived by lens intrinsic and converts tangents of viewing angles to the projected texture coordinates. P represents the rotation matrix. For simplicity and generality, we use the rotation part of the head poses so that the corresponding pixel position can be computed without knowing the exact depth of each point. Besides, we don't require screen-space jitter as we observe the head pose jitter from human motion and sensor noise in current VR headset is enough for anti-aliasing.

The blending factor (α) between the current and previous frame is spatially varying to provide a smooth transition between regions. The value of blending factor depends on foveated rendering shading rate, eccentricity, tile size, etc. α drops from one to a small fractional number in far periphery. The smooth fall off also helps slightly reduce the visibility of the boundary between tiles with different shading rate.

*e-mail: haomiao.jiang@oculus.com
†e-mail: tianxin.ning@oculus.com
‡email: behnam.bastani@oculus.com

978-1-7281-6532-5/20/$31.00 ©2020 IEEE
DOI 10.1109/VRW50115.2020.00-23

Figure 3: (left) Without rejection (right) With rejection logic.

3.1 Rejection Logic

To avoid color bleeding and ghosting artifacts, pixels on moving objects need to be detected and rejected for blending. (Figure 3).

We compare the pixel color from sampled previous reconstructed frame to the min-max color bounding box with its the four direct neighbor. If any of the colors of the previous pixel falls outside, the color is tone mapped to the region boundary along the line connecting the two colors, i.e.

$$\hat{c}_t = c_t + (1-\alpha)\hat{c}_{t-1}$$

$$\alpha = \max_{rgb}(\min(\frac{c_{min} - \hat{c}_{t-1}}{c_t - \hat{c}_{t-1}}, \frac{c_{max} - \hat{c}_{t-1}}{c_t - \hat{c}_{t-1}}))$$

where c_t and $\hat{c}_{t-1}$ represents the color pixel color and corresponding reconstructed previous frame pixel values respectively. c_{max} and c_{min} represents the max / min RGB values of the four foveation-awared direct rendered neighbors and the pixel itself. In low shading rate tiles, we sample to the shaded neighbors instead of direct spatial neighbors in eye buffer. The color bounding box is normalized by the sampling distance / shading rate.

3.2 Avoiding Blur

As we blend recusively with previous frames, the method can introduce blurriness and loss of contrast if no special treatment is involved. Loss of contrast in large area of periphery region can result in tunnel vision artifact [2] and needs to be addressed. The main source of the blurriness comes from the interpolation / resampling of previous frame, which gets amplified by the inherent recurrency of the algorithm. Without considering rejection logic, the reconstructed value is defined by

$$\hat{I}_t[x,y] = \alpha I_t[x,y] + (1-\alpha)(g \circ \hat{I}_{t-1})[x,y]$$

Here g is the resampling / warping function. With texture sampling, g is handled with bilinear interpolation.

To reduce the blurriness, we reduce the low pass filtering effect in g by moving the sampling location towards pixel center. In extreme cases, the sampling location is moved exactly at pixel center, and bilinear resampling falls back to nearest neighbor resampling. However, the incontinuity of nearest resampling causes warbling artifacts. Experimentally, we observed quadratic scaling towards the center gives a good result for sharpness without warbling. Additionally, we avoid blurriness by pre-sharpening with the neighbor values used for rejection (Figure 4).

3.3 Performance Optimization

On mobile GPUs, tile-based rendering reduces the bandwidth of accessing off-chip memory. Performing a separate renderpass is costly as it requires data going back and forth between high-speed GPU local cache and system memory. To keep the optimal performance, Vulkan subpass is introduced to keep all the color content within the tile local memory. More optimally, custom resolve shader in graphics driver can be used as a part of the resolving stage to perform temporal anti-aliasing. With foveated rendering, we only perform temporal anti-aliasing in the downsampled region, before resolving to full resolution in system memory while pixels in full-resolution region are untouched.

Figure 4: (Left) foveated rendered image with 1/16 shading rate (Mid / Right) flicker controlled without / with sharpening

Table 1: Speed of Proposed Flicker Control Method

Foveated Rendering Level	Pixel Density	Time (ms)
Low	47.26%	0.78
Medium	38.49%	0.91
High	31.30%	0.98

4 Results

We measured the GPU render time of proposed algorithm using a synthesized scene. The result includes extra render time for both eyes at 1440 x 1536 resolution. GPU clock is fixed to ensure consistent profiling results. With custom resolve shader enabled, the proposed algorithm takes less than 1 ms. We evaluate the perceptual quality via a user study with two VR scenes running on Oculus Quest. In 31 out of 41 trials, subject prefers flicker controlled visual quality. The user test validate the visual improvement brought by the proposed fast flicker reduction method.

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):14–35, 2017.

[2] A. H. Chan and A. J. Courtney. Foveal acuity, peripheral acuity and search performance: A review. *International Journal of Industrial Ergonomics*, 18(2-3):113–119, 1996.

[3] J. Jimenez, J. I. Echevarria, T. Sousa, and D. Gutierrez. Smaa: enhanced subpixel morphological antialiasing. In *Computer Graphics Forum*, vol. 31, pp. 355–364. Wiley Online Library, 2012.

[4] J. Korein and N. Badler. Temporal anti-aliasing in computer generated animation. In *ACM SIGGRAPH Computer Graphics*, vol. 17, pp. 377–388. ACM, 1983.

[5] J.-H. Nah, S. Ki, Y. Lim, J. Park, and C. Shin. Axaa: Adaptive approximate anti-aliasing. In *ACM SIGGRAPH 2016 Posters*, p. 51. ACM, 2016.

[6] 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 (TOG)*, 35(6):179, 2016.

[7] L. R. Rabiner and B. Gold. Theory and application of digital signal processing. *Englewood Cliffs, NJ, Prentice-Hall, Inc., 1975. 777 p.*, 1975.

[8] E. Turner, H. Jiang, D. Saint-Macary, and B. Bastani. Phase-aligned foveated rendering for virtual reality headsets. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 1–2. IEEE, 2018.

[9] C. W. Tyler. Analysis of visual modulation sensitivity. iii. meridional variations in peripheral flicker sensitivity. *JOSA A*, 4(8):1612–1619, 1987.

[10] K. Xiao, G. Liktor, and K. Vaidyanathan. Coarse pixel shading with temporal supersampling. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, p. 1. ACM, 2018.