Combining Eye Tracking with Optimizations for Lens Astigmatism in modern wide-angle HMDs

Daniel Pohl* Intel Corporation

Xucong Zhang† Max Planck Institute for Informatics

Andreas Bulling‡ Max Planck Institute for Informatics

Abstract—Virtual Reality has hit the consumer market with affordable head-mounted displays. When using these, it quickly becomes apparent that the resolution of the built-in display panels still needs to be highly increased. To overcome the resulting higher performance demands, eye tracking can be used for foveated rendering. However, as there are lens distortions in HMDs, there are more possibilities to increase the performance with smarter rendering approaches. We present a new system using optimizations for rendering considering lens astigmatism and combining this with foveated rendering through eye tracking. Depending on the current eye gaze, this delivers a rendering speed-up of up to 20%.

Index Terms—virtual reality, astigmatism, eye tracking, ray tracing, optics

1 Introduction

Through affordable display panels from smartphones, sensors and inexpensive wide-angle lenses, head-mounted displays (HMDs) have reached into the consumer market [8, 9]. Even though the technology has progressed a lot in recent years, one of the common complaints is that single pixels are visible when using the HMD (screendoor effect). Higher resolution displays, potentially up to 16K, are needed to fix this, but also drive up the rendering requirements. As the human eye only perceives the area sharp where it is focusing at, foveated rendering through eye tracking can be used to lower the computational requirements. But given that all current consumer HMDs have several lens distortions, there is more to exploit in a smart rendering architecture. Our contribution is to combine foaveated rendering techniques through eye tracking with optimizations regarding the HMD's lens astigmatism to improve performance up to 20%.

2 Related Work

Modern consumer HMDs compensate two optical lens distortions in software: radial and chromatic aberrations [16, 5, 1, 11]. While fixing this in software might lead to some blur, it is generally agreed on that to reduce cost and weight in an HMD this is the preferably way for consumer products compared to adding multiple expensive lenses for distortion compensation. Astigmatism is another optical distortion [4], which cannot be removed through software. It leads to the effect that inside the HMD only the center area can be perceived as sharp, while with increasing distance from it the image gets more blurred. Pohl et al [10] proposed to map this behavior into a static sampling map and to adjust the sampling of the renderer to these properties, i.e. to render with higher image quality in the center and with lower one in the blurred outer areas. In a sampling map, an intensity value of 255 represents to use the maximum amount of allowed supersampling, e.g. 8 rays per pixel. A value of 128 would lead to using 4 rays per pixels and so on. The approach of using a static sampling map does not consider eye tracking, which we propose here to merge into a dynamically adjusted sampling map to get even more performance savings.

The idea to make use of the properties of the human eye for rendering dates back quite a while, when Tong and Fisher [14] developed in 1984 a flight simulator, in which a large low quality background image was projected onto a dome and a smaller, high quality image was added into the area where the user was looking. Levoy and Whitaker [7] change the number of rays that are used in a volume ray tracer, depending on the eye gaze. A newer approach from Guenter et al [3] follows the same idea using a rasterizer, which generates three images at different sampling rates, and composites them together. While these are all valid approaches to increase performance, we combine these ideas with optimizations for lens astigmatism in consumer HMDs for an even more efficient rendering architecture.

Eye tracking for professional HMDs has been shown before, like the system from Duchowski et al [2]. Recently, eye tracking has also been added to consumer HMDs by students [13] using cheap components as well as through professional, commercial offers from companies like SensoMotoric Instruments [12]. Our self-constructed eye tracking solution lies in the middle, using a medium-priced camera prototype combined with cheap LEDs.

3 Rendering

For our experimental research renderer, we adapt the concept of sampling maps [10] to change the amount of samples per pixel on a fine granular level. However, we do not limit ourselves to a static sampling map, but we update it every frame depending on the current eye gaze. To be able to quickly do that, we do not create it in the full screen resolution, but in a smaller resolution of 320×180 pixels. The texture could also be expressed as an equation using the distance to the lens center, but we found the look up to be faster. The standard sampling map, just to optimize for lens astigmatism, can be seen in Figure 1 left. During rendering, we use a smoothly interpolated version of it. We create another sampling map for the eye gaze. We use a radius of 0.25 (assuming 0 to 1 for the horizontal image dimensions of one eye). We set the sampling intensity for the foveated region to maximum in that area as shown in Figure 1 center. An enhanced approach could use a fine mapping of the human eyes' acuity instead of using the maximal amount of samples. Next, we create the final sampling map by taking the minimum value between the sampling map for lens astigmatism and the one for the current eye gaze. The result is depicted in Figure 1 right. As a tradeoff between rendering time and image quality, we test the setup with a maximum of 8 rays per pixel for supersampling.

4 Experimental Results

4.1 System Description

We create side-by-side stereo images in a total resolution of 1920×1080 pixels using an in-house ray tracer, partly accelerated by Intel Embree [15]. We use the "island" level from *Enemy Territory: Quake Wars*[1]. We use a Dual-CPU (Intel® Xeon®

*e-mail: daniel.pohl@intel.com

†e-mail: xczhang@mpi-inf.mpg.de

‡e-mail: bulling@mpi-inf.mpg.de

Manuscript received 12 September 2013; accepted 10 January 2014; posted online 29 March 2014; mailed on 1 May 2014.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

[1] id Software and Splash Damage

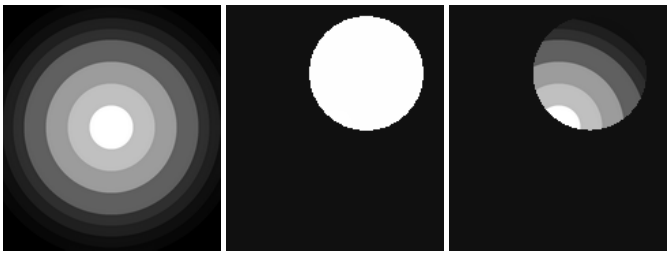

Fig. 1: *The intensity in the sampling map determines the number of samples per pixel used. Left: sampling map considering the HMD's lens astigmatism. Center: sampling map for the current eye gaze. Right: combined sampling map, taking the minimum values of the other two sampling maps in the foveated region.*

E5-2699 v3, 2.3 GHz, 18 physical cores) PC with a NVidia® GeForce® 970 GTX. As HMD we utilize the Oculus Rift DK2 [8]. We modified it by adding infrared lights and a small infrared eye camera to record the eye movement. The camera is the Pupil Pro head-mounted eye tracker that can record videos of the eye at 120 Hz in a resolution of 640×480 pixels [6]. To enable the camera to get a good look at the eyes, a hot mirror was added at a steep angle in front of the display. A photo of this prototype can be seen in Figure 2. The eye gaze is calibrated for every user by showing a calibration pattern with a total of 12 points, of which one is shown at a time and the user has to focus on it. The full 12 points can be seen in Figure 3 (left). A captured image with pupil detection is shown in Figure 3 (right).

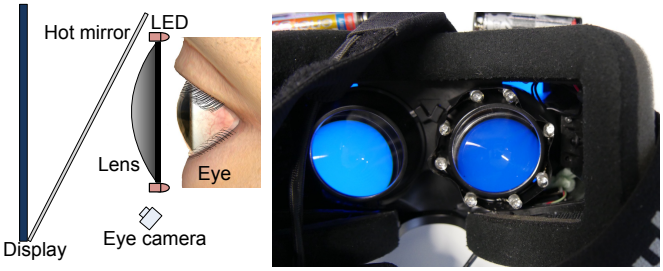

Fig. 2: *Eye tracking system. Left: illustration of how the eye tracking works. Right: Exterior of the modified Oculus Rift DK2.*

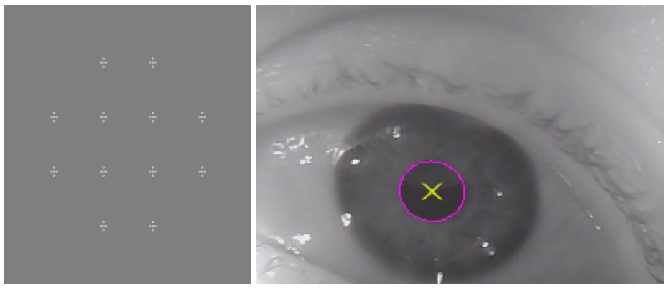

Fig. 3: *Left: visualization of all calibration points. Only one is shown at the same time. The procedure starts with the left top point. The eye camera takes a picture while the user looks at the point. Afterwards, the point smoothly moves to the position of the next neighbor. Gray is used as background color to ensure an average brightness, affecting the opening size of the pupil. Right: video captured image of the eye with pupil detection.*

4.2 Rendering Performance

We compare the achieved frame rates of our approach with the different methods for rendering the image with a maximum of 8 rays per pixel. The speed-up depends on where the eye gaze is. In Table 1, we show the values for the eye gaze as shown in Figure 1. Our combined approach is 18% faster there. Looking at points further away, the speed-up is 20%. When the eye gaze is in the center, there are fewer savings possible as the sampling map for astigmatism indicates high amount of supersampling there, but the combined methods still gives a speed-up of 5%. When allowing even higher supersampling than 8 rays per pixel, more performance improvements would be possible.

Method	Frames per second
Brute-force supersampling	14 fps
Lens astigmatism	35 fps
Foveated rendering	39 fps
Foveated + lens astigmatism (our approach)	46 fps

Table 1: Performance comparison at the eye gaze of Figure 1

5 Limitations

It should be noted that the general concept of using a sampling map, either static or dynamic, requires a rendering architecture in which the amount of supersampling can be chosen on a per-pixel level. This applies potentially to custom-written renderers like ray tracers, voxel ray caster or software-based rasterization engines that could be extended with these features. With the latest versions of the APIs for hardware rasterization on GPUs, OpenGL 4.5 and DirectX 12, it is not possible to freely choose the amount of supersampling per pixel. However, the sampling map concept could be used there in a more broadly sense, where a certain level of detail, e.g. in tessellation or shader complexity, could be applied on a per-pixel level.

6 Conclusion

In this paper, we have shown that by combining optimizations for lens astigmatism with foveated rendering, that speed-ups of up to 20 percent can be achieved. This lightens the performance requirements for future HMDs, especially considering the expected increase in screen and temporal resolution.

References

[1] M. Antonov, N. Mitchell, A. Reisse, L. Cooper, and S. LaValle. Oculus SDK overview, 2013. `https://developer.oculusvr.com/`.

[2] A. T. Duchowski, E. Medlin, A. Gramopadhye, B. Melloy, and S. Nair. Binocular eye tracking in VR for visual inspection training. In *VRST*, 2001.

[3] B. Guenter, M. Finch, S. Drucker, D. Tan, and J. Snyder. Foveated 3D graphics. *TOC (Proc. SIGGRAPH Asia)*, 31(6), 2012.

[4] E. Hecht. *Optics*. Addison-Wesley, 2002.

[5] K. Inoguchi, M. Matsunaga, and S. Yamazaki. The development of a high-resolution HMD with a wide FOV using the shuttle optical system. In *SPIE*, volume 6955, 2008.

[6] M. Kassner, W. Patera, and A. Bulling. Pupil: an open source platform for pervasive eye tracking and mobile gaze-based interaction. In *UbiComp*, pages 1151–1160, 2014.

[7] M. Levoy and R. Whitaker. Gaze-directed volume rendering. In *I3D*, pages 217–223, 1990.

[8] Oculus VR. Oculus Rift, 2014. `http://www.oculusvr.com/`.

[9] J. Olson, D. Krum, E. Suma, and M. Bolas. A design for a smartphone-based head mounted display. In *VR*, pages 233–234, 03 2011.

[10] D. Pohl, T. Bolkart, S. Nickels, and O. Grau. Using astigmatism in wide angle HMDs to improve rendering. In *VR*, pages 263–264, 2015.

[11] D. Pohl, G. S. Johnson, and T. Bolkart. Improved pre-warping for wide angle, head mounted displays. In *VRST*, 2013.

[12] SensoMotoric Instruments. Eye tracking HMD upgrade package for Oculus Rift DK2, 2015. `http://www.smivision.com/en/gaze-and-eye-tracking-systems/products/eye-tracking-hmd-upgrade.html`.

[13] M. Stengel, S. Grogorick, M. Eisemann, E. Eisemann, and M. Magnor. An affordable solution for binocular eye tracking and calibration in head-mounted displays. In *Proc. Multimedia 2015*, pages 15–24, Oct. 2015.

[14] H. Tong and R. Fisher. *Progress Report on an Eye-Slaved Area-of-Interest Visual Display*. Defense Technical Information Center, 1984.

[15] I. Wald, S. Woop, C. Benthin, G. S. Johnson, and M. Ernst. Embree: A kernel framework for efficient CPU ray tracing. *TOG (Proc. SIGGRAPH)*, 33(4):143:1–143:8, 2014.

[16] B. Watson and L. Hodges. Using texture maps to correct for optical distortion in head-mounted displays. In *VR*, pages 172–178, 1995.