Foveated Image Refocusing in VR Applications

Yazan Aldehayyat
Stanford University
350 Serra Mall, Stanford, CA
yazana@stanford.edu

Lester Hing Kwok Lao
Stanford University
350 Serra Mall, Stanford, CA
lelao@stanford.edu

Abstract

In this paper we propose a foveated refocusing technique as an alternative to traditional foveated rendering for VR applications. Unlike traditional techniques, we propose to partially render and blur the entire frame then refocus the foveal region of the image locally in the HMD. In addition to reducing the latency requirement of the eye tracking system, our technique also reduces the bandwidth required between the GPU and the HMD since lossless compression can achieve better results with blurred images. Furthermore, similar to previous methods, our technique achieves lower GPU computational load since we partially render each frame.

1. Introduction

Many foveated rendering techniques have been proposed in literature to support higher resolution displays while reducing GPU workload and bandwidth requirements between the HMD and the GPU. However, many of those techniques impose a very stringent latency constraint on the VR system. This is because rendering every frame correctly in a 60Hz display panel requires a motion to photon latency of less than 20ms accord to Michael Abrash [1]. Figure 1 shows a typical data path for traditional foveated rendering illustrating the difficulty in meeting the latency needed.

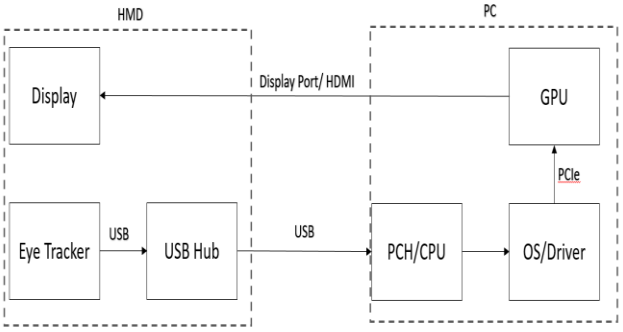

Figure 1: Data path for traditional foveated rendering

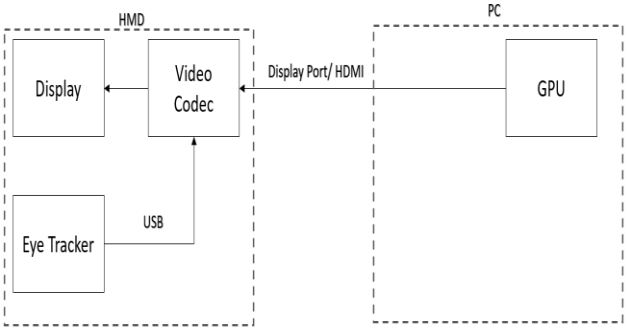

Figure 2: Our proposed data path topology

Furthermore, having the rendering of each frame dependent on eye tracking makes predictive rendering far more difficult since we now need to not only predict the movement of the user but also his gaze. Figure 2 shows our proposed topology where GPU side rendering is completely de-coupled from eye tracking which resolves both of the above constraints.

2. Related Work

Our project builds on 3 main concepts; non-blind image deconvolution, image compression, and foveated rendering. Although all of these individual concepts have been thoroughly researched and explored, optimizing an image pipeline that combines all three poses some unique challenges that as far as we are aware have never been addressed. For example, understanding the choice of blurring kernel on compression ratio after the process of generating custom priors before the image is blurred is a good place to start.

To understand the impact of foveated rendering on the computational load of the GPU and the latency requirements we consulted this paper by Brian Guenter et al. [2]. While this paper provided a great guide on non-blind image convolution and de- blurring, it does not specifically discuss the impact of background blurring on

image compression, but we found this one to be a good resource on the topic. Aside from this, Anjul Patney et al. [3] does hint that decreasing quality in the periphery while maintaining high fidelity in the focal area can increase pixel density and refresh rates, which can be interpreted to decreasing workload and bandwidth requirements on the HMD side.

3. Method and Steps

In this project we split our work into two sections; GPU side rendering and HMD side rendering. On the GPU side we took a computer-generated image from Pamela Torres [4] and simulated the process of partial rendering by deleting pixels based on several criteria. The image was chosen, first, for the relative intricacy in the image. The combination of detailed and plain characters in the image gives different types of images for an eye to focus on. Second, the video game theme was very analogous to many VR applications. After that, we blurred the image with various blurring kernels at different parameters in an attempt to understand how those parameters impact the compression ratio and the ability to recover that image. In addition to improving the compression process, applying a blurring kernel on the partially rendered image acts as an anti-aliasing filter which is needed for certain partial rendering techniques.

On the HMD side we applied several methods for deblurring the image and compared the results versus run time. Weiner filtering was the only method that provided a real time implementation and as such we used it in our eye-tracking demo.

3.1 GPU Side Rendering

Since rendering our own images was outside of the scope we used a pre-rendered image to simulate the workload done on the GPU side. We selected the following image as our sample since it had multiple sections of low and high frequencies as well as a large gamut of colors. Figure 3 shows the original image we started with.

Figure 3: Original image

3.1.1 Partial Rendering

To simulate partial rendering, we took the original image and removed all the pixels that are similar to their neighboring pixels within a given threshold. A larger threshold meant that the GPU had to render fewer pixels at the expense of losing detail in the image while a lower threshold increased computational workload and preserved more detail. This technique preserves the edges in the image at the expense of the details which is often the better compromise to make since humans are more sensitive to edges. This similarity parameter can be chosen based on the performance of the GPU and can be adjusted in real time to guarantee a consistent frame rate at the expense of degradation in image quality. Figure 4 shows our image after it has been subjected to our simulated partial rendering technique.

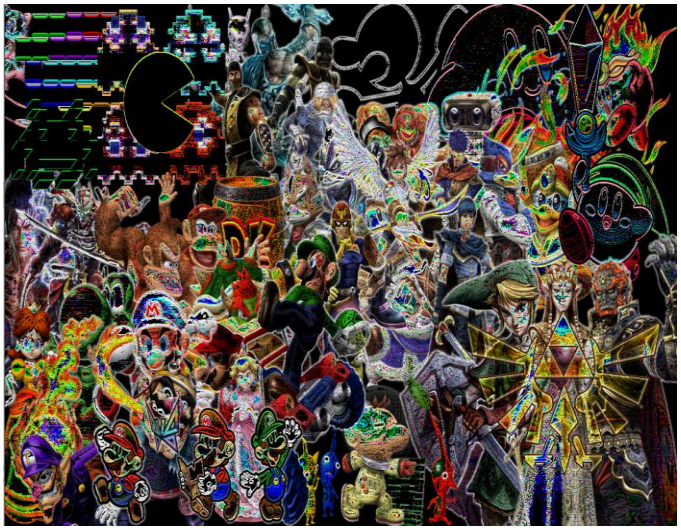

Figure 4: Simulated partial rendering

3.1.2 Blurring

After the image has been rendered we apply a blurring kernel. We experimented with multiple different kinds of blurring kernels such as Gaussian, bilateral, motion and disk filter. One important feature that our kernel must have is that it is mathematically invertible, otherwise we would not be able to unblur it in the later steps. In addition, we were interested in the blurring kernel that gives the best compression ratio while still maintaining image quality after deblurring. Figure 5 compares the compression ratio of different blurring kernels while holding the PSNR of the image constant. Despite equal PSNR the Gaussian provided the best perceptual image quality while the disk filter left noticeable artifacts in the image. Figure 6 shows these artifacts.

	Gaussian	Disk	Motion
Picture			
Compression	70%	65%	79%
PSNR	25.10db	25.63db	25.47db

Figure 5: Blurring kernels and compression ratios

Figure 6: 'Disk' artifacts after de-blurring

3.1.3 Image Compression

After blurring the image, we simulate the compression process by simply writing the image to disk. This uses Matlab's compression engine. We compared the size of the original image versus the size of the blurred image compressed with both JPEG and PNG at different quality levels. Figure 5 summarizes these results of image compression ratio vs compression technique. While Figure 7 shows the impact that the standard deviation of the Gaussian blurring kernel has on compressed image size for PNG compression. For the remainder of the project we used PNG compression since JPEG compression severely degrades the quality of the recovered image.

Sigma	1	2	3	5	10
Picture					
Compression	84%	72%	65%	56%	46%

Figure 7: Gaussian sigma vs compression

3.2 HMD Side Refocusing

In this part of the project we sample the blurred image and compared various deblurring methods. This is traditionally an inversing problem as said by Ruxin Wang *et al.* [5]. Therefore, we mainly focused on Wiener deconvolution for a simple inversing solution and ADMM with various priors as a more powerful one. The Wiener deconvolution method is very computationally efficient as we have a known kernel with virtually no noise. This is why we used it in our demo, since it can be implemented in real time quite easily and efficiently. On the other hand, ADMM is a more powerful technique that can handle noise in the image better.

Figure 8 show a comparison of the various de-blurring techniques and resulting images. Wiener deconvolution performs quite well, likely because the image contains very little noise. The various ADMM priors all perform quite similarly but the runtime is vastly different.

3.2.1 Wiener Deconvolution

The biggest benefit to Wiener filtering is its simplicity. We simply deconvolve the image by multiplying by the inverse of the blurring kernel with the addition of weighting factor in the Fourier domain. The weighting factor is used to control noise and prevent the product from becoming undefined. In our case we added another weighting factor that acts as high boost filter. The weighting factor boost the high frequencies, but it also adds noise to the image. Finding the optimal value requires some optimization. You can see Figure 9 and

Figure 10 for localized de-blurring of our blurred image which were obtained from the eye-tracking code.

Figure 8: Comparison of various de-blurring techniques

Figure 9: Localized de-blurring (Dr. Mario)

Figure 10: Localized de-blurring (Zelda)

4. ADMM filtering

ADMM is a powerful optimization technique that can be used in many different applications. ADMM uses a multi-step iterative process, utilizing a prior to iterate to optimize the solution. The deconvolution of an image is an ill-posed problem with infinitely many solutions; therefore, the prior helps us narrow down those options to the ones we want. In this project we used several different priors including one that we devised ourselves. For example, we used BM3D which is a denoising technique that works in this application because it limits the high frequency noise. Another example is the total variation (TV) prior which ensures that only sparse gradients are present in the image, which is a feature of natural images. On the other hand, we also used our own prior based on information we gained from the image before we blur. We called this prior, "Gradient weighted prior" and is discussed in the following section

4.1. Gradient weighted prior.

The idea is behind the gradient weighted prior (GWP) method is that since we have information about the image before we blur it, we can use that information to also deblur it. In this case we derive the gradient of the image before we blur it, calling it the true gradient. Then, we soft threshold the gradient of the blurred image to the true gradient. This is similar to the TV prior except we are not thresholding to zero in this case. In this example we only use the magnitude of the gradient of the intensity channel. However, this technique can also be adapted to soft threshold both x and y gradients of each color channel separately. We need to make sure that the gradient information will not be too large since that would increase the bandwidth required between HMD and PC.

5. Demo

To demonstrate real-time refocusing we had setup that uses a Tobii Eye Tracker and a Matlab script that automatically de-blurs sections of the image based on a user's gaze. Currently, the demo is hard-coded for a specific resolution of screen 3000x2000(267 DPI) and the character.png image which has a resolution of 900x1600. The demo and instruction on how to run it can be found in the accompanied zip file. Since an eye tracker is not normally available, the demo supports simulating the eye tracker using your mouse cursor. Simply hover over any section of the image and the Matlab script will de-blur and focus that section of the image. Please see an image of our set-up in Figure 11 and 12. Aside from our eye tracker demo, we also included an ADMM demo for static image recovery. Using reference ADMM code from Stanly Chan *et al.* [6], we were also able to implement our own gradient comparison prior to the reference code and compared it with the reference as seen above in Figure 8.

Figure 11: Demo set-up with Tobii

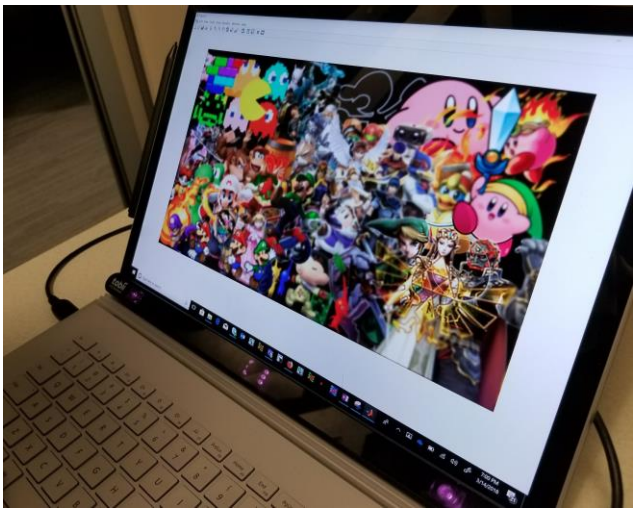

Figure 12: Set-up with Tobii focus on bottom right (Zelda)

6. Discussion

Our results and demo show that the overall concept is feasible; however, many areas of the project can be optimized further. For example, we think that it might be possible to implement an invertible bilateral filter that produces a blurred image with intact edges but also an entropy map that can be used reconstruct the sharp image. Another area of improvement is investigating other ADMM priors to find one that is better suited for this application. For example, we could use the depth map of the image instead of the gradient map as a prior. This could also allow us to incorporate depth data in the blurring kernel to add depth cues to our reconstructed image. Finally, our technique at the moment only support still images. The methods we used will have to be adapted to different video encoding techniques to actually work on a moving image. The demo can also be optimized to work with any screen resolutions and with any image pixel density.

7. Conclusion

Our results show that an HMD side foveated refocusing algorithm could be a low latency alternative to traditional foveated rendering techniques. Despite its simplicity, Wiener deconvolution performs quite well in this application since noise is low. Also, Wiener deconvolution is simple enough to be implemented in real-time, making it the most readily available option for implementation. Conversely ADMM has the potential to improve the quality of the recovered image. New priors are most likely required to improve both runtime and perceived image quality.

References

[1] Michael Abrash. ”Why Virtual Reality is Hard” GDC 2013.Available: http://media.steampowered.com/apps/abrashblog/MAbrash%20GDC2013.pdf [March 14 2018]

[2] Brian Guenter & Mark Finch & Steven Drucker & Desney Tan,& John Snyder. "Foveated 3D Graphics. ACM Transactions on Graphics (TOG)." 31. 10.1145/2366145.2366183.Available: https://www.microsoft.com/en-us/research/wp-content/uploads/2012/11/foveated_final15.pdf [March 9, 2018]

[3] Anjul Patney & Marco Salvi & Joohwan Kim & Anton Kaplanyan & Chris Wyman & Nir Benty & David Luebke & Aaron Lefohn. "Towards foveated rendering for gaze-tracked virtual reality." ACM Transactions on Graphics. 35. 1-12. 10.1145/2980179.2980246.Available: http://cwyman.org/papers/siga16_gazeTrackedFoveatedRendering.pdf [March 9, 2018]

[4] Pamela Torres, Video-Game-Wallpaper, 2015.

[5] Ruxin Wang, and Tao Dacheng . "Recent progress in image deblurring." arXiv preprint arXiv:1409.6838 (2014).Available: https://arxiv.org/pdf/1409.6838.pdf [March 9, 2018]

[6] S. H. Chan, X. Wang and O. A. Elgendy, "Plug-and-Play ADMM for image restoration: Fixed point convergence and applications," IEEE Transactions on Computational Imaging, Nov. 2016.