Display Stream Compression for VR Media

Stephen Lopez
Stanford University
Department of Electrical Engineering
slopez27@stanford.edu

Abstract

To move towards wireless or even network-based connections, HMD's must reduce their bandwidth without sacrificing high fidelity images. By accepting losses in image quality using foveated rendering and perceptually optimized video compression, we can reduce bandwidth consumption while maintaining high resolution images. We find that we can greatly reduce bandwidth consumption at the cost of poor visual fidelity in non-foveated areas. Without a proper eye tracker, it is difficult to tell whether the losses in non-foveated areas are acceptable.

1. Introduction

Modern Virtual Reality HMD's are pushing towards extremely high resolution displays which need powerful computer systems. These powerful computer systems need high-band-width interfaces to send and receive data which forces VR HMD's to be connected through a myriad of wires. Our focus is on reducing the band-width necessary and to move towards wireless or even network-based VR HMDs. Studies that utilize different foveated rendering techniques have shown that opportunities for wireless VR systems are available when using foveated rendering [3].

We plan on tackling bandwidth reduction by exploiting two characteristics of VR HMDs. Since VR HMDs are physically close to one's eyes, we can exploit foveated rendering to only render high quality images at the fovea's gaze. After we render our image, we use H.264 compression that takes into account the fovea position and the warped image the lens creates to further reduce bandwidth. By utilizing foveated rendering and H.264 compression, we will lower resolution in areas that don't require the same high visual fidelity.

1.1. Foveated Shading

The fovea is a small depression in the eye that has the highest density of cones and thus captures significant amount of detail. The areas surrounding the fovea capture color and motion but do not have the same visual acuity as the fovea. Foveated shading takes advantage of this biological phenomena by rendering the image in accordance to the detail needed. The center, or where the fovea is gazing, should be rendered as highly detailed as possible while the surrounding regions can sacrifice visual fidelity. By dividing our image into two regions, we end up with a foveated region and a non-foveated region. Since we want to achieve extremely aggressive compression to gain substantial bandwidth reduction, we further divide our non-foveated region into two rings, where the outer ring will be more aggressively compressed.

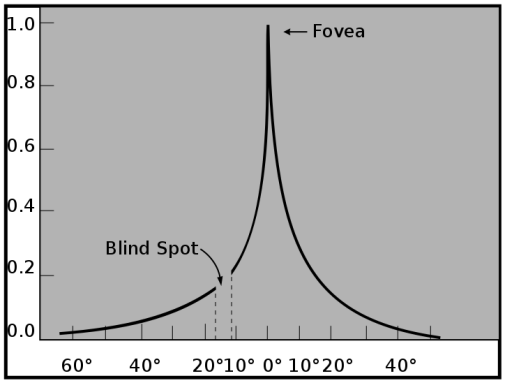

Figure 1. The following graph shows the correlation between visual acuity/cone distribution and distance from fovea measured in degrees. We can see that our best visual acuity is centered and around the fovea.

1.2. H.264 Compression

Our VR HMD requires a lens between the display screen and our eyes. We realized that these lenses warp at the edges and cause us to lose visual fidelity in the surrounding regions. By applying the same fundamentals in foveated rendering, we decided that we could use our lens warp to further optimize our compression at the edges of the lens.

This compression must happen at the end of our rendering pipeline, where we already have a fully shaded and warped image. We apply H.264 compression that will be optimized to take into account our fovea position and the effects of lens warping. We divide our image into multiple subsections, where we can individually assign compression factors, or "error sensitivity". These compression factors allow us to sacrifice visual fidelity in warped/non-foveal regions while maintaining high resolution images in the center/foveal regions.

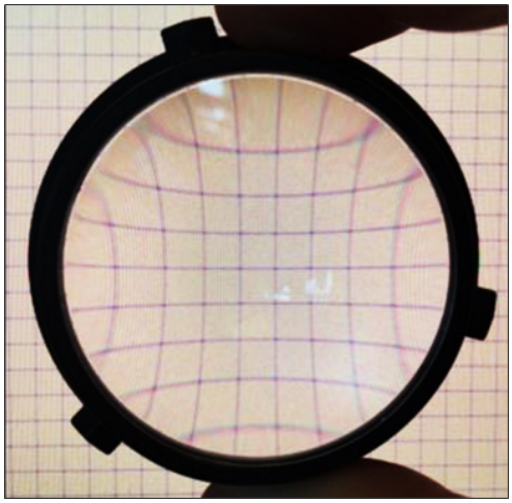

Figure 2. The image shows us the warping effects that a lens can create. We notice that at the edges of the lens our image are severely warped while the center is practically untouched. These differences can allow us to further optimize our compression algorithms.

1.3. Eye Tracking

The foundation of our project relies heavily on the ability to track the fovea's position. We believe it is an interesting problem to solve, but we will not incorporate it for this project. We will assume we always know the user's fovea position, which in this case is the center of the image, and render images accordingly. We realize that this makes it difficult to actually test our foveated rendering effectiveness, so we will instead focus on the bandwidth gains and hope to test the effectiveness of our foveation in future iterations of our project.

2. Methods

The implementation of foveated shading and H.264 compression both try to reduce bandwidth by lowering our visual fidelity in non-key regions. Foveated shading happens during the shading portion of the rendering pipeline, while H.264 compression is a post-processing algorithm. Both functions seek to optimize compression strength by using natural phenomena, fovea position and lens warping, to decide which areas of our image need high visual fidelity. While both functions were tested, I primarily focused on foveated shading while my partner Neil focused on H.264 compression. As such I will leave the finer details of H.264 compression to my partner Neil's report.

2.1. Foveated Shading

The first step of foveated shading is to parse the image into the foveal region and non-foveal region. Since we wanted more aggressive compression without creating a jarring shift between the foveal and non-foveal region, we divided the non-foveal region into two different sections that will have varied compression strength. With our three sections in place, we needed to decide how large each section should be. We found that on average the fovea's angular span from the center was 7.76 degrees and that a more inclusive angle was 11.39 degrees [4]. We found that our foveal size is highly dependent on our eye tracking and latency, so we decided to have two different tests that used these measurements as our inner radii [1]. Our outer radii didn't have strict guidelines, so we settled on using double the size of our inner radii to differentiate between our non-foveal regions.

Next we had to decide how we wanted to differentiate shaders between our three segments. We settled on utilizing Blinn-Phong shading for each segment and controlling our compression strength through loop perforation. The result resembles a block Blinn-Phong shader where we can control the size of our block. Through the first pass, our shader computes a single value in each block skipping the other values in our segment. The second pass computes which fragment is the key frament in the segment and copies the already computed fragment color. By changing the block size, we control how aggressively we compress in that region. The final iteration of our project has a pixelated effect in the non-foveal regions that are accaptable because these regions do not need the same visual acuity as the foveal region. Our foveal region never has any compression during the shading process since we want to maintain the full visual acuity in this region. We found that right outside our foveal region, we lose 75 percent of visual acuity [6]. With that in mind, we compressed the middle region by a factor of 4 and the outside region by a factor of 9. We also wanted to try out aggressive foveation, we so compressed our middle segment by a factor of 9 and the outer segment by a factor of 16.

Finally, we then must decide how we want to quantize our results. We decide to measure image quality using PSNR, but we later find that PSNR does not necessarily capture the data we wish to know..To measure bandwidth consumption, we just compute the ratio of "important" pix-

els, the pixels that dictate the color of the block in our loop perforation, to our original image, where every pixel is an important pixel.

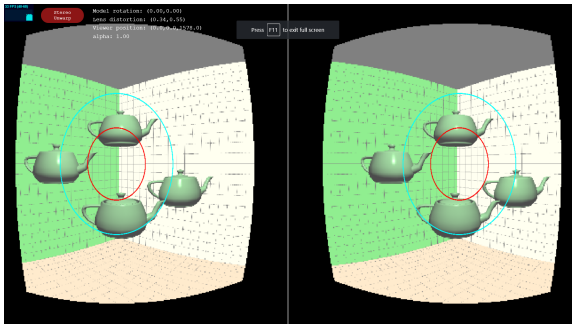

Figure 3. We assume the fovea is centered and proceed to distinguish between our three regions. The center is our foveal region, the middle ring is our mildly compressed region while the outer ring is our most aggressively compressed region.

2.2. H.264 Compression

We utilized H.264 compression after our image was rendered to further reduce bandwidth. Our H.264 compression is perceptually optimized to take into account lens warping and foveal gaze. We parse our image into a 4x4 grid and individually assign each grid an error sensitivity which coincides with their place in relation to fovea and lens. Each of the 16 parts is then compressed using H.264 compression algorithm while taking into account their error sensitivity. We make sure to limit the error to an empirically tuned bound. We calculate PSNR to measure our image quality.

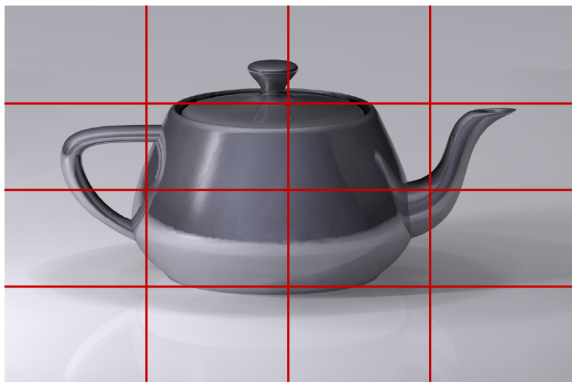

Figure 4. We can see how we parse the image into a 4x4 grid. Each one of these sections has its own compression value defined which leads to different compression strengths for each section.

3. Results/Analysis

By using foveated shading and H.264 compression, we hoped to lower bandwidth consumption without sacrificing image quality. We made sure to leave as much visual fidelity in our key regions while aggressively compressing around the edges. We found that our foveated shading had substantial bandwidth savings at the cost of poor visual fidelity in our non-foveated regions. Our H.264 compression did not yield results that we expected and will need further tuning before we begin to see significant bandwidth savings.

3.1. Foveated Shading

For our analysis on Foveated shading, we found significant bandwidth reduction for the cost of visual fidelity. We tested using two distinct inner radii sizes, and their respective outer radii, and we also tested both with two different forms of compression. While we found the savings we expected, it is difficult to evaluate our success based solely on numbers. It is impossible to test our foveation without a dedicated fovea tracker since at the end of the day, what matters most is if the user is capable of telling if their is significant loss in visual fidelity. We will further evaluate our foveal shading in future iterations of our work.

We knew we were aggressively compressing our image, we we decided to measure how much visual fidelity we were sacrificing through PSNR. We took a video of our teapot and found that the clip had a PSNR of 35.2. We then compared images of our evaluated foveated shading to our original image, and found that the PSNR of every one of our foveated rendered teapots was around 14.9. Regardless of foveal size or compression strength our images showed minuscule variance in PSNR. This variance suggested that regardless of foveal size or compression strength, we will have the same drop in visual fidelity. While this is an interesting find, I believe significant testing is still required before accepting such a bold statement. I believe a unforeseen issue we might have faced is the image we decided to use. We have significant compression in the outer regions yet the surrounding areas in our image are considerably sparse. We end up not seeing the effects of our aggressive foveation. Even with a more dense image, I also found that PSNR does not capture the problem we were trying to solve. Foveated region accepts the losses in the surrounding areas and renderes the image center perfectly. With that in mind, the only tests to capture our foveation is to gather opinions as to whether the foveation is noticeable/nauseating/distracting. PSNR values give an interesting look as to the visual fidelity the entire image sacrificed, but we are interested as to whether we notice the losses when staring at the foveated region.

Through our analysis we found that we reached the significant bandwidth reductions that we were expecting. Our

original plan of using a smaller foveal regeion with regular compression achieved a compression factor of around 4.3x. The smaller foveal region should have a larger non-foveal region which should lead to more significant bandwidth reductions. We found that the smaller foveal region had larger bandwidth reduction than the larger foveal region by a a factor of 1.7x. We found that when we aggressively compressed, used 4x and 9x instead of 9x and 16x compression factors for the middle and outer ring respectively, we achieved a compression factor of about 1.5x. Without proper foveated rendering testing that requires eye tracking, it is difficult to tell whether these bandwidth savings can be possibly implemented or if the visual acuity is just too low.

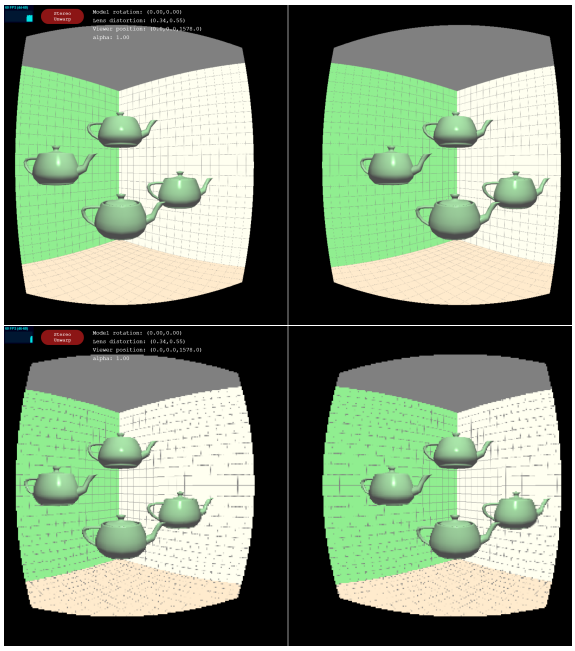

Figure 5. A comparison between the image being normally rendered, and rendering using foveated shading. The image of the top has no foveation while the image in the bottom uses the large fovea and intense compression.

Foveation	PSNR	Bandwidth Compression
Normal	35.2	1
Small	14.95	4.3264
Small Intense	14.87	6.1065
Large	14.99	2.8473
Large Intense	14.91	3.6472

Table 1. The graph compares the PSNR and Bandwidth Compression of our normal image to our various foveated images.

Even though out PSNR isn't very telling of what's going on in our images, we can clearly see that there were significant bandwidth reductions. With the proper implementation of eye tracking, we can hope to conclude our foveated shading results.

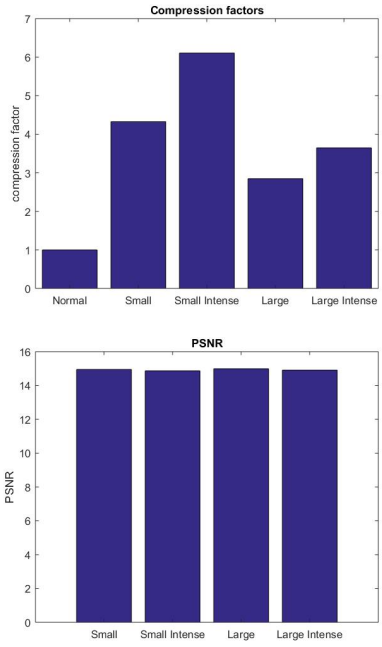

Figure 6. The top image shows a comparison between the different compression factors while the bottom image is a comparison between the PSNR values. For our compression we found significant bandwidth reductions that varied according to fovea size and compression intensity. Our PSNR values on the other hand did not show any variance regardless of fovea size and compression intensity. These findings were surprising and need further investigating.

3.2. H.264 Compression

There were many difficulties when testing out H.264 compression and we did not see any substantial difference. The PSNR of our compressed image was over 35.2 so there was very little difference in visual acuity or bandwidth reduction. We hope to continue testing using different compression algorithms to acquire more substantial differences. My partner Neil will talk more about H.264 Compression in his paper.

4. Discussion

I found that we made some interesting strides towards lowering bandwidth consumption. I would say using perforated loops and foveated shading had its substantial gains but it is impossible to measure the success without fully acknowledging the fact that we need to test it to see if the foveated shading works with eye tracking. The H.264 compression was an interesting experiment but it was difficult to

place inside the normal pipeline and we didn't find interesting observations. We will need to continue testing different iterations of compression algorithms before calling it a failure.

4.1. Testing Foveated Rendering

When starting this project we assumed that we know the fovea's position was at the center and rendered our compression algorithms with that foveal zone. Now that we have found the visual acuity and bandwidth reduction relationship, it is necessary to implement foveal tracking. We can't fully evaluate the success of our foveated rendering unless we know that our algorithms are unnoticeable to the user.

First and foremost, the most important implementation to truly analyze if our foveated rendering is visibly acceptable is to implement some sort of eye tracking hardware/software. Once we have an eye tracker, it also becomes important to understand the necessary latency requirements to make it acceptable.

On the other hand, assuming we have perfect eye tracking, it then becomes necessary find a way to measure the effectiveness of foveated rendering. I found that measuring using PSNR was not helpful and I'm not sure other visual fidelity measurements would help. We would have to test our machine and create surveys to see the effectiveness of our foveated shading. Particularly we wish to know if it is distracting, nauseating, or noticeable. It is a difficult process to quantize but we would begin to understand what are acceptable boundaries.

4.2. Foveated Shading

The process of foveated shading itself could go through many more iterations. Currently we can only apply loop perforation through integers, so a possible extension would be to allow non-integer loop perforation. Non-integer loop perforation would allow to create smoother transitions and avoid jarring transitions. This could allow our aggressive foveation as long as the transitions aren't noticeable. To further smooth our image, we can find ways to interpolate our color. We currently just look at the dominant pixel and shade in the entire block with the same color, but instead we could implement some more interpolation to get smoother transitions without having to actually go through the shader calculations for each individual pixel. This could increase our PSNR without significantly increasing computational power or bandwidth consumption.

Currently we're working with Blinn-Phong shading since we had difficulty with trying to implement various shaders throughout each region, for example Blinn-Phong in the center, Gourand in the middle ring, and Flat for the outer ring. I would like to continue trying out various shaders since I believe it could offer significant improvements. Finally, I also believe it would be interesting to try to implement more complicated shading techniques, such as ray tracing. I would want to observe how foveated shading affects more complicated shaders.

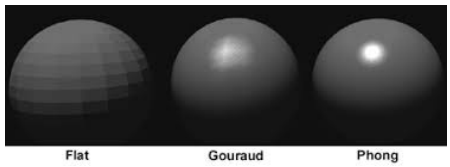

Figure 7. The image shows three different shaders which all have different levels of visual fidelity. It would be interesting to try to implement these shaders all in conjunction. Something like Blinn-Phong in the center, Gourand in the middle ring and Flat shading in the outer ring would be interesting.

There are a lot of interesting techniques and algorithms that could be incorporated to make the foveated rendering more bearable. We are worried that even with decent eye tracking and non-aggressive foveation, it is difficult to create a VR HMD that isn't nauseating or disorienting because of artifacts in the non-foveal zone. NVIDIA's research found that the peripheral vision is capable of capturing contrast, not just color and motion. Traditional foveated rendering induced a form of tunnel vision bacause of the lack of contrast in the blurred regions. By applying a post-processing contrast enhancement, they were able to use a smaller foveal zone and increase the blur region [5]. Google has found that foveated rendering can cause pereptual artifacts when motion is induced. To combat these artifacts, Google has included Phase-Aligned Rendering and Conformal Rendering which create smoother scenes that allow more aggressive foveation. [2] We would look into incorporating these functions into future implementations.

4.3. Post-Processing Compression

We ended up running out of time before fully exploring H.264 compression. Not only are we going to continue tuning our parameters but we will continue exploring functions to decide our compression factors. We will look into existing saliency predictors which use data collected from VR simulations to understand how people explore virtual environments [7].

References

[1] R. Albert, A. Patney, D. Luebke, and J. Kim. Latency requirements for foveated rendering in virtual reality. *ACM Trans. Appl. Percept.*, 14(4):25:1–25:13, Sept. 2017.

[2] E. T. Behnam Bastani. Introducing a new foveation pipeline for virtual/mixed reality, 2017.

[3] G. Illahi, M. Siekkinen, and E. Masala. Foveated video streaming for cloud gaming. *CoRR*, abs/1706.04804, 2017.

[4] R. A. Jonas, Y. X. Wang, H. Yang, J. J. Li, L. Xu, S. Panda-Jonas, and J. B. Jonas. Optic disc - fovea angle: The beijing eye study 2011. 10(11):1–10.

[5] 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 Trans. Graph.*, 35(6):179:1–179:12, Nov. 2016.

[6] F. D. Purves D, Ausgustine GJ. Anatomical distribution of rods and cones. *Neuroscience 2nd Edition*, 2001.

[7] V. Sitzmann, A. Serrano, A. Pavel, M. Agrawala, D. Gutierrez, B. Masia, and G. Wetzstein. How do people explore virtual environments? *IEEE Transactions on Visualization and Computer Graphics*, 2017.