On the Interplay of Foveated Rendering and Video Encoding

Gazi Karam Illahi
Aalto University, Finland
gazi.illahi@aalto.fi

Matti Siekkinen
Aalto University, Finland
matti.siekkinen@aalto.fi

Teemu Kämäräinen
Aalto University, Finland
teemu.kamarainen@aalto.fi

Antti Ylä-Jääski
Aalto University, Finland
antti.yla-jaaski@aalto.fi

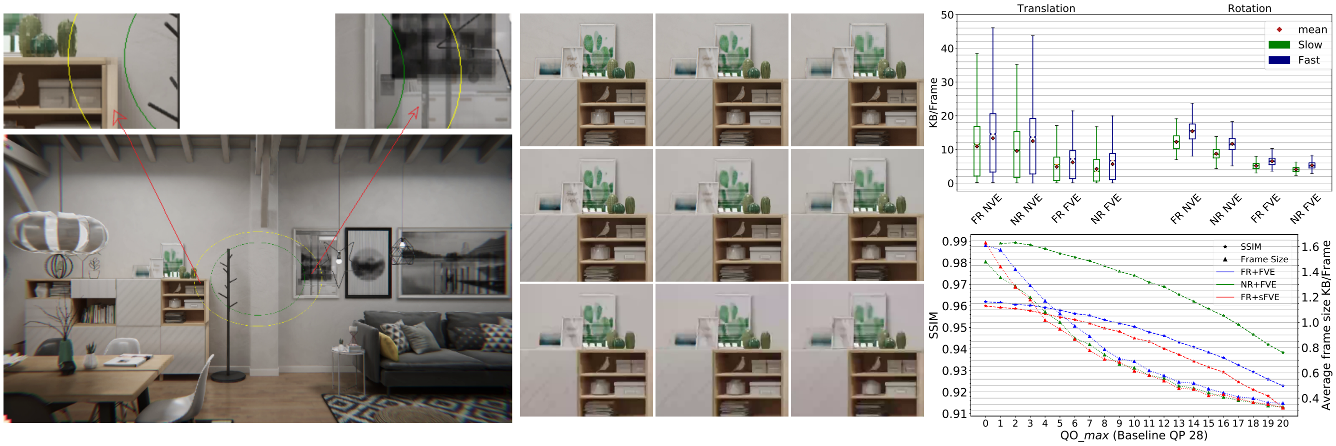

Figure 1: Left to right: (1) An overview of a Foveally Rendered scene with three shading rates and (2) a detailed sample from a frame rendered with different shading rates and QP values. The shading rates used in (1) are 1 shading pass per 1 pixel (within green ellipse), per 2x2 pixels (between green and yellow ellipses), and per 4x4 samples (beyond yellow ellipse). In (2) the same shading rates are used from right to left and encoded with (top to bottom) QPs: 0, 28, 38. QP=0 corresponds to lossless encoding, QP 28 is a typical value used for encoding and QP=38 corresponds to the QP of the lowest quality region in our FVE scheme.

ABSTRACT

Humans have sharp central vision but low peripheral visual acuity. Prior work has taken advantage of this phenomenon in two ways: *foveated rendering* (FR) reduces the computational workload of rendering by producing lower visual quality for peripheral regions and *foveated video encoding* (FVE) reduces the bitrate of streamed video through heavier compression of peripheral regions. Remote rendering systems require both rendering and video encoding and the two techniques can be combined to reduce both computing and bandwidth consumption. We report early results from such a combination with remote VR rendering. The results highlight that FR causes large bitrate overhead when combined with normal video encoding but combining it with FVE can mitigate it.

CCS CONCEPTS

• **Computing methodologies** → **Virtual reality**; *Rendering*; Non-photorealistic rendering; • **Computer systems organization** → *Real-time system architecture*; • **Networks** → *Cloud computing*.

KEYWORDS

virtual reality, foveated rendering, video encoding, cloud rendering

ACM Reference Format:
Gazi Karam Illahi, Matti Siekkinen, Teemu Kämäräinen, and Antti Ylä-Jääski. 2020. On the Interplay of Foveated Rendering and Video Encoding. In *26th ACM Symposium on Virtual Reality Software and Technology (VRST '20), November 1–4, 2020, Virtual Event, Canada.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3385956.3422126

1 INTRODUCTION

Interactive applications aiming for high quality visual experience, such as AAA video games and Virtual Reality (VR) applications, require powerful graphics hardware for real time rendering. To avoid the need for local PC having a dedicated graphics card to render graphics for these applications, remote rendering systems for gaming [2, 7, 9, 22] and VR [3, 11, 13, 14] have emerged. They offload (most) rendering tasks from client device to a remote server and stream the rendered graphics in real time as encoded video to

---

Matti Siekkinen and Teemu Kämäräinen are also with University of Helsinki, Finland.

VRST '20, November 1–4, 2020, Virtual Event, Canada

ACM ISBN 978-1-4503-7619-8/20/11.
https://doi.org/10.1145/3385956.3422126

the client device. However, high quality video streaming in such a system requires substantial amount of bandwidth.

Human visual system has spatially non-uniform acuity [24]: sharp vision at the point of gaze and exponentially decreasing acuity with angular eccentricity from that point. The phenomenon is also called *foveation*. It can be leveraged to reduce both graphics computing as well as bandwidth requirements in remote rendering.

Foveated rendering (FR) uses a spatial quality profile that aims to match the visual acuity of the eye [5, 12, 16, 18, 19, 23]. The scene is divided into multiple regions each of which is rendered according to its angular eccentricity from the gaze point using, e.g., variable rate shading [19]. *Foveated video encoding* (FVE) applies the same principle to encode video frames with a spatially varying quality profile similarly aimed to match our visual acuity [1, 8, 10, 15, 20, 21, 25]. This can be done by, e.g., varying the quantization of individual coding units (macroblocks in h.264) according to their distance from the gaze point, reducing the resulting video bitrate accordingly. We call this technique *foveated quantization*. With both FR and FVE, the viewer gaze can be either predicted, reported in real time by an eye-tracker, or a combination of the two.

In this paper, we present early results from combining FR and FVE in a remote rendering system. While both methods have been independently researched previously, we believe to be the first to closely examine their interplay.

2 EXPERIMENTS

We ran experiments with different combinations of the techniques: FR combined with normal video encoding (NVE), normal rendering (NR) with FVE, FR with FVE, and as a baseline NR with NVE.

We simulated client gaze and controls programmatically and instead of streaming the rendered video, saved it to disk at the server. As user head motion, we simulated translation only and rotation only motion and, in addition, replay a real user trace combining both types of motion. In translation only mode, the player (and hence the camera) was programmed to perform continuous rectilinear translation. In rotation only mode, the player was programmed to rotate along the vertical axis (yaw) continuously. Slow (5 m/s for translation and 10 deg/s for rotation) and fast (10 m/s and 20 deg/s) motion were simulated in both modes. The point of gaze was fixed to the center of the rendered frame in all experiments where player moved. Additionally, we simulated situations where only the gaze was shifting and scene was otherwise static and gaze fixation duration was either *short* ($500ms$), *long* ($5secs$), or *fixed* (at center) cases. The gaze locations were intuitively modelled as a bi-variate Gaussian distribution centered around the center of the scene. In each experiment, 3 minutes worth of video data was recorded.

We used a Unity-based remote rendering system in our experiments with gaze data captured from the client and a server application that allows configuring FR and/or FVE parameters. The scene used was "ArchVizPRO Interior Vol.6" available in Unity Asset Store containing 3D photo-realistic user explorable house (Fig 1).

FR was implemented using the variable rate shading feature of Nvidia VRworks suite of APIs [17] and the Vive plugin [6]. It allows setting different shading rates for different regions within a frame and varying them across frames. It can improve performance without reducing perceived quality. Samples with different shading rates are shown in Figure 1. We used three level shading where high quality (1 shading pass per pixel) region was fixed to a radius of 1/8th of frame width, the transition quality (1 pass per 4 pixels) region to a radius of 1/6th of frame width and the rest was set to low quality (1 pass per 16 pixels).

To capture and encode rendered frames into video, we use a modification of the 360 Capture SDK [4]. Framerate was set to 60 fps. FVE in the form of foveated quantization is implemented by adjusting the underlying h.264 encoder's quantization offset QO for each macroblock of a video frame. The method is the same as in [8]. We set the standard deviation of the 2-D Gaussian used to calculate the QO to 1/8th of the frame width, and the maximum possible QO (QO_{max}) to 10. In all experiments, the encoding scheme used was Nvidia's low latency preset and the rate control mode was constant QP with $QP = 28$ as the baseline unless otherwise mentioned.

3 RESULTS

The top-right plot in Fig 1 shows the resulting frame sizes that directly reflect the bandwidth demand. Interestingly, FR with NVE produces up to 30% larger frames on average compared to NR and NVE. The reason is that, even though there is less total visual information in a frame rendered with FR, the dissimilarity between two successive frames rendered with FR increases compared to NR. This means that inter-frame compression is less effective and the size of the predicted frames increases. We obtain similar but more pronounced results with experiments where only gaze shifts within a static scene with FR nearly doubling the bandwidth demand compared to normal rendering when FVE is not applied. Applying FVE together with FR remedies the situation. However, in all experiments we observe that NR+FVE yields smaller frames than FR+FVE, which hints that it may be possible to design a scheme that produces even smaller frames than the two independent methods combined.

The bottom-right plot in Fig 1 shows how structural similarity index measure (SSIM) and frame size behave with different parameter settings. The results were calculated using a single static scene rendered using NR, FR and encoded using FVE, NR with NVE being the reference. As expected, SSIM decreases with increasing QO_{max}. However, comparing the shapes of the SSIM and frame size curves reveals an interesting tradeoff between them: most of the savings in frame size can be obtained with small sacrifice in quality (e.g., using $QO_{max} = 10$). The results agree with those reported in [8] where the authors used a similar FVE scheme.

4 CONCLUSION

This paper reports early results from combining foveated rendering and video encoding in remote rendering systems. The results suggest that foveated rendering combined with normal video encoding may dramatically increase bandwidth consumption but applying foveation also in video encoding mitigates the problem. As future work, we plan to conduct user studies to better understand the impact of the two techniques on visual experience and to explore whether frame sizes can be further optimized with a tailored combination of FR and FVE.

ACKNOWLEDGMENTS

This work is supported by the Academy of Finland (grant 332306).

REFERENCES

[1] Z. Chen and C. Guillemot. 2010. Perceptually-Friendly H.264/AVC Video Coding Based on Foveated Just-Noticeable-Distortion Model. *IEEE Transactions on Circuits and Systems for Video Technology* 20, 6 (2010), 806–819.

[2] Nvidia Corporation. [n.d.]. *GeForce Now.* Retrieved June 29, 2020 from https://www.nvidia.com/en-us/geforce-now/

[3] Nvidia Corporation. [n.d.]. *NVIDIA CloudXR SDK.* Retrieved June 29, 2020 from https://developer.nvidia.com/nvidia-cloudxr-sdk

[4] Facebook, Inc. 2020. *360 Capture SDK.* Facebook, Inc., Menlo Park, California. https://github.com/facebookarchive/360-Capture-SDK

[5] Brian Guenter, Mark Finch, Steven Drucker, Desney Tan, and John Snyder. 2012. Foveated 3D graphics. *ACM Transactions on Graphics (TOG)* 31, 6 (2012), 1–10.

[6] HTC Corporation. 2020. *Vive Foveated Rendering for Unity.* HTC corporation, Xindian, New Taipei , Taiwan. https://github.com/ViveSoftware/ViveFoveatedRendering

[7] Chun-Ying Huang, Cheng-Hsin Hsu, Yu-Chun Chang, and Kuan-Ta Chen. 2013. GamingAnywhere: An Open Cloud Gaming System. In *Proceedings of the 4th ACM Multimedia Systems Conference* (Oslo, Norway) *(MMSys '13)*. ACM, New York, NY, USA, 36–47. https://doi.org/10.1145/2483977.2483981

[8] Gazi Karam Illahi, Thomas Van Gemert, Matti Siekkinen, Enrico Masala, Antti Oulasvirta, and Antti Ylä-Jääski. 2020. Cloud Gaming with Foveated Video Encoding. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 16, 1 (2020), 1–24.

[9] Google Inc. [n.d.]. *Stadia.* Retrieved June 29, 2020 from https://stadia.google.com/

[10] Laurent Itti. 2004. Automatic foveation for video compression using a neurobiological model of visual attention. *IEEE Transactions on Image Processing* 13, 10 (Oct 2004), 1304–1318. https://doi.org/10.1109/TIP.2004.834657

[11] Teemu Kämäräinen, Matti Siekkinen, Jukka Eerikäinen, and Antti Ylä-Jääski. 2018. CloudVR: Cloud accelerated interactive mobile virtual reality. In *Proceedings of the 26th ACM international conference on Multimedia.* 1181–1189.

[12] Anton S Kaplanyan, Anton Sochenov, Thomas Leimkühler, Mikhail Okunev, Todd Goodall, and Gizem Rufo. 2019. DeepFovea: Neural reconstruction for foveated rendering and video compression using learned statistics of natural videos. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–13.

[13] Zeqi Lai, Y Charlie Hu, Yong Cui, Linhui Sun, Ningwei Dai, and Hung-Sheng Lee. 2019. Furion: Engineering high-quality immersive virtual reality on today's mobile devices. *IEEE Transactions on Mobile Computing* (2019).

[14] Luyang Liu, Ruiguang Zhong, Wuyang Zhang, Yunxin Liu, Jiansong Zhang, Lintao Zhang, and Marco Gruteser. 2018. Cutting the Cord: Designing a High-Quality Untethered VR System with Low Latency Remote Rendering. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '18).* Association for Computing Machinery, New York, NY, USA, 68–80. https://doi.org/10.1145/3210240.3210313

[15] Pietro Lungaro, Rickard Sjöberg, Alfredo J.F. Valero, Ashutosh Mittal, and Konrad Tollmar. 2018. Gaze-Aware Streaming Solutions for the Next Generation of Mobile VR Experiences. *IEEE Transactions on Visualization and Computer Graphics* 24, 4 (April 2018), 1535–1544. https://doi.org/10.1109/TVCG.2018.2794119

[16] Xiaoxu Meng, Ruofei Du, Matthias Zwicker, and Amitabh Varshney. 2018. Kernel foveated rendering. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1, 1 (2018), 1–20.

[17] Nvidia Corporation. 2020. *Nvidia VRWorks SDK.* Nvidia Corporation, Santa Clara, USA. https://developer.nvidia.com/vrworks

[18] C O'Sullivan, J Dingliana, and S Howlett. 2002. Gaze-contingent algorithms for interactive graphics. *The Mind's Eyes: Cognitive and Applied Aspects of Eye Movement Research, J. Hyönä, R. Radach, and H. Deubel, Eds. Elsevier Science, Oxford* (2002).

[19] Anjul Patney, Marco Salvi, Joohwan Kim, Anton Kaplanyan, Chris Wyman, Nir Benty, David Luebke, and Aaron Lefohn. 2016. Towards foveated rendering for gaze-tracked virtual reality. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 179.

[20] Miguel Fabian Romero-Rondón, Lucile Sassatelli, Frédéric Precioso, and Ramon Aparicio-Pardo. 2018. Foveated streaming of virtual reality videos. In *Proceedings of the 9th ACM Multimedia Systems Conference.* 494–497.

[21] Jihoon Ryoo, Kiwon Yun, Dimitris Samaras, Samir R. Das, and Gregory Zelinsky. 2016. Design and Evaluation of a Foveated Video Streaming Service for Commodity Client Devices. In *Proceedings of the 7th International Conference on Multimedia Systems* (Klagenfurt, Austria) *(MMSys '16).* ACM, New York, NY, USA, Article 6, 11 pages.

[22] Ryan Shea, Jiangchuan Liu, Edith C-H Ngai, and Yong Cui. 2013. Cloud gaming: architecture and performance. *IEEE Network* 27, 4 (2013), 16–21.

[23] Nicholas T Swafford, José A Iglesias-Guitian, Charalampos Koniaris, Bochang Moon, Darren Cosker, and Kenny Mitchell. 2016. User, metric, and computational evaluation of foveated rendering methods. In *Proceedings of the ACM Symposium on Applied Perception.* 7–14.

[24] Brian A Wandell. 1995. *Foundations of vision.* Vol. 8. sinauer Associates Sunderland, MA.

[25] Zhou Wang, Ligang Lu, and Alan. C. Bovik. 2003. Foveation scalable video coding with automatic fixation selection. *IEEE Transactions on Image Processing* 12, 2 (Feb 2003), 243–254. https://doi.org/10.1109/TIP.2003.809015