Aalto University

OPEN ACCESS

This is an electronic reprint of the original article.
This reprint may differ from the original in pagination and typographic detail.

Illahi, Gazi Karam; Siekkinen, Matti; Kamarainen, Teemu; Yla-Jaaski, Antti

Foveated streaming of real-Time graphics

Published in:
MMSys 2021 - Proceedings of the 2021 Multimedia Systems Conference

DOI:
10.1145/3458305.3463383

Published: 15/07/2021

Document Version
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

Please cite the original version:
Illahi, G. K., Siekkinen, M., Kamarainen, T., & Yla-Jaaski, A. (2021). Foveated streaming of real-Time graphics. In *MMSys 2021 - Proceedings of the 2021 Multimedia Systems Conference* (pp. 215-226). ACM. https://doi.org/10.1145/3458305.3463383

This material is protected by copyright and other intellectual property rights, and duplication or sale of all or part of any of the repository collections is not permitted, except that material may be duplicated by you for your research use or educational purposes in electronic or print form. You must obtain permission for any other use. Electronic or print copies may not be offered, whether for sale or otherwise to anyone who is not an authorised user.

Foveated Streaming of Real-time Graphics

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

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

Teemu Kämäräinen
University of Helsinki
Helsinki, Finland
teemu.kamarainen@helsinki.fi

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

ABSTRACT

Remote rendering systems comprise powerful servers that render graphics on behalf of low-end client devices and stream the graphics as compressed video, enabling high end gaming and Virtual Reality on those devices. One key challenge with them is the amount of bandwidth required for streaming high quality video. Humans have spatially non-uniform visual acuity: We have sharp central vision but our ability to discern details rapidly decreases with angular distance from the point of gaze. This phenomenon called *foveation* can be taken advantage of to reduce the need for bandwidth. In this paper, we study three different methods to produce a foveated video stream of real-time rendered graphics in a remote rendered system: 1) foveated shading as part of the rendering pipeline, 2) foveation as post processing step after rendering and before video encoding, 3) foveated video encoding. We report results from a number of experiments with these methods. They suggest that foveated rendering alone does not help save bandwidth. Instead, the two other methods decrease the resulting video bitrate significantly but they also have different quality per bit and latency profiles, which makes them desirable solutions in slightly different situations.

CCS CONCEPTS

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

KEYWORDS

Virtual Reality, Foveated Rendering, Cloud Rendering, Foveated Streaming

ACM Reference Format:
Gazi Karam Illahi, Matti Siekkinen, Teemu Kämäräinen, and Antti Ylä-Jääski. 2021. Foveated Streaming of Real-time Graphics. In *12th ACM Multimedia Systems Conference (MMSys '21) (MMSys 21), September 28-October 1, 2021, Istanbul, Turkey.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3458305.3463383

MMSys 21, September 28-October 1, 2021, Istanbul, Turkey
© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in *12th ACM Multimedia Systems Conference (MMSys '21) (MMSys 21), September 28-October 1, 2021, Istanbul, Turkey*, https://doi.org/10.1145/3458305.3463383.

1 INTRODUCTION

High quality visual experience with real-time graphics requires powerful hardware for graphics processing. Therefore, AAA video games and Virtual Reality (VR) applications typically rely on a PC that has a dedicated graphics card for rendering, which is expensive, constrains mobility, and can be laborious to setup.

Remote rendering offloads the (heavy) graphics computing work to a remote server that streams the rendered graphics as encoded video to the client device in real time. In this way, a high quality experience can be provided for resource constrained (mobile) devices [14, 41]. However, this approach has its own challenges. Besides the need for ultra-low latency, remote rendering requires substantial amount of bandwidth for high quality video streaming from server to client. Fortunately, the specifics of human visual system can be leveraged to mitigate this challenge.

It is well known that human visual system has spatially non-uniform acuity [45]: sharp vision centered at the point of gaze and exponentially decreasing acuity with angular eccentricity from the point of gaze. The phenomenon is also called *foveation.*

If gaze tracking information is available, remote rendering can take advantage of foveation to reduce bandwidth requirements. The idea is to decrease the level of detail within a video frame from regions that are spatially distant from the point of gaze. In this way, the overall amount of information per frame is reduced, which potentially leads to lower bitrate of the encoded video. As a side effect, the amount of computational work in rendering can be reduced also by considering foveation already in the graphics rendering phase.

In this paper, we compare three different methods to stream foveated video in a remote rendering system. We consider foveation in graphics rendering (foveated rendering), in post processing right after the rendering (foveated warping), and in the video encoding phase (foveated video encoding). The foveated warping is, to the best of our knowledge, original in the context of foveated streaming of real-time graphics. Interestingly, our results reveal that foveated rendering alone does not help reduce the resulting video bitrate, mainly due to its harmful effect on inter-frame compression. Foveated warping effectively reduces the number of pixels per frame, whereas foveated encoding reduces the number of bits per pixel within a frame. Both of these methods lead to substantial bitrate savings but differ in terms of quality and other benefits. While foveated warping appears to fall slightly behind foveated encoding from quality per bit point of view, it can potentially offer notably lower latency and/or higher resolution visual experience due to reduced video encoding and decoding workload.

In summary, we make the following contributions:

- We present three different ways to stream foveated real-time rendered content including a novel approach based on 2d image warp.
- We implement these techniques in a real system and run a number of experiments to evaluate their performance, including a small scale test with human subjects.
- We show that foveated rendering only is a poor choice and that among the two other methods, there is no clear winner because they have different tradeoffs between quality per bit and video coding latency and resolution limits.

The rest of the paper is organized as follows: We first introduce the background and related work in Section 2 regarding foveation both for video encoding and rendering. We then introduce three different methods to include foveation in a remote rendering system in Section 3. In Section 4, we present the remote rendering system, the experiment setup and the metrics used in our evaluation (Section 5). Lastly, we discuss the results in the context of choosing the right method for a foveated remote rendering system in Section 6 before concluding the paper.

2 BACKGROUND AND RELATED WORK

In this section, we introduce the background and related work related to foveation, both in the context of foveated rendering and foveated video coding. Before this, we also present how the foveation phenomenon is an in-built property of the Human Visual System (HVS) and introduce the concept of streamed remote rendering.

2.1 Streaming Remote Rendered Graphics

Remote rendering leverages cloud computing in the context of graphics computing. In a remote rendering system, a server with sufficient compute and rendering resources executes a typically complex graphics application to render the graphics, compresses the rendered frames and transmits the compressed frames to (typically) a thin client which decompresses and displays the frames. Interactive remote rendered applications allow the thin client to send control inputs to the rendering server [42]. The frames rendered in a remote rendering system may be compressed using e.g geometry compression [12] or video encoding [15].

Streaming remote rendered graphics is an actively researched area, owing to its use in applications like remote desktop, cloud gaming, and cloud-rendered VR and XR [42]. Currently, there is a heightened commercial focus on cloud gaming with the launch of commercial services like GeForce Now [3] and Google Stadia [19]. Since mobile devices have limited compute, render and energy resources, streaming remote rendered graphics is under intense research focus as an enabling technology to deliver untethered interactive XR experiences to mobile devices [4].

Previous research on remote rendered XR has shown that the computing resources can be decoupled from the standalone device either by a dedicated wireless link [28] or by cloud-enabled rendering with a traditional WiFi or mobile network connection [21]. The stringent latency requirements of remote rendering have been a particular focus for XR and traditional cloud gaming. Lai et al. [23] utilized pre-rendered frames to address the latency issues in cloud-rendered VR. On the other end of the pipe-line, Li et al. [26] proposed the use of 3D image warping to reuse received pixels between multiple frames on the client side. Both of these techniques can be used together with the foveated optimization methods introduced in this paper.

2.2 Foveation in Human Visual System

Foveation, the phenomenon of non-uniform visual acuity of the human eye, is well known [45]. Fovea centralis, a small region on the retina directly behind the lens has the highest density of cone cells and it is responsible for our sharp central vision. The regions surrounding the fovea centralis have progressively lower cone density resulting in proportionally lower resolution sampling of a visual scene. This non-uniformity of cone distribution and cortical magnification lead to foveation [6, 37]. The highest visual acuity of the human eye is estimated to be within an spatial angle of 2^o of the human visual field, falling sharply with increasing eccentricity [6]. This is also reflected in the variation of the so-called cutoff frequency with eccentricity. Cutoff frequency, at a given viewing distance, is the maximum spatial frequency perceived by the human eye and varies with eccentricity from the fovea centralis. Figure 1 shows the cutoff frequency of the human eye with respect to eccentricity from the fovea centralis.

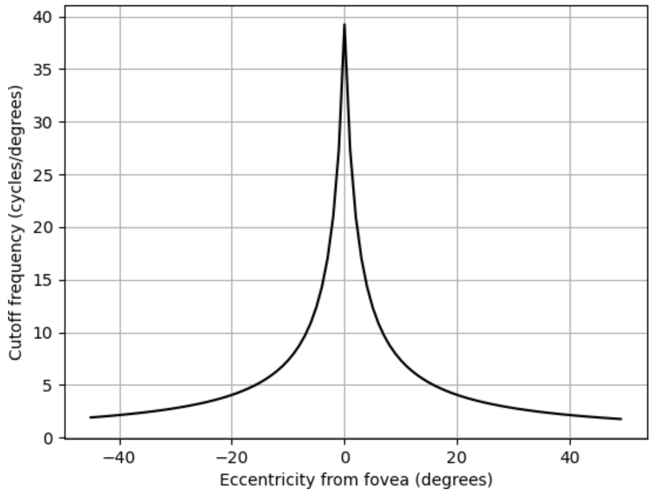


Figure 1: Cutoff frequency as a function of eccentricity from the fovea.

2.3 Foveated Video Encoding

Exploiting foveation in video coding has been explored for a long time as *foveated video coding* [20, 46, 47]. The main principle is to spatially vary the encoding quality of each video frame such that the quality is highest at viewers' gaze location and lower elsewhere. The viewer gaze is either predicted or reported in real time by an eye-tracker. However, deployment of foveated video coding into commercial space has been slow due to gaze tracking or prediction requirements. More recently, with the availability of consumer grade eye trackers, it is seeing a renewed interest, especially for real-time planar and 360^o video streaming applications.

Lungaro et al. [29] proposed a tile-based gaze-aware streaming solution to minimize the bandwidth usage in 360^o video. Similarly, Romero-Rondón et al. [39] demoed a prototype of a system with foveated video streaming for VR. In this paper, we extend the approach to real-time rendered graphics and an evaluation of three different foveation methods. Gaze-based foveation systems have been also proposed for cloud gaming [16] and traditional video streaming [40]. In this paper, we evaluate further how foveated video encoding interplays with foveated rendering.

2.4 Foveated Rendering

Development of consumer grade eye-tracking solutions has stimulated research into *foveated rendering*, which encompasses those foveation methods that are applied within the graphics rendering pipeline. Foveated rendering uses a spatial quality profile that attempts to match the visual acuity of the eye [11, 22, 30, 33, 34, 43]. This entails dividing the scene into multiple regions and rendering each region according to its angular eccentricity from the point of gaze: the central region with highest quality, the most peripheral regions with lowest quality, and the regions in between with a transitional quality.

The different quality levels with correspondingly reduced computing can be achieved, for instance, with variable rate shading (VRS) [5, 34]. As the name implies, VRS enables dynamically setting different shading rates for different regions within a frame, allowing per frame compute resource provisioning. It can be used to improve render quality where the viewer is focusing, likely to focus or where details are more visible due to scene geometry and lighting and lower the render quality in regions of the frame where the viewer is not focusing or where details are less visible, e.g in dark corners. This approach improves performance while either not impacting perceived quality or even improving perceived quality.

2.5 Other Related Work

The third method that we consider, namely Foveated Warping, is based on variable resolution sampling, which is the underlying principle for many foveated rendering and foveated video coding methods discussed in the above sections. For a discussion, we refer the reader to [24]. A method similar to our approach, except that it targets video conferencing systems, is presented in [2]. In contrast to our approach, the resolution scaling is not based on the cutoff frequency at a pixel location. Furthermore, the method uses lookup tables to implement the variable resolution scaling and does not reduce the pixel dimensions of the image. Projections similar to foveated warping have been used in the context of remote rendering, although not for the purpose of foveation and, therefore, also not based on properties of the HVS, such as the custom wide angle projection presented in [36]. Fish eye lenses have also been presented as an approach to implement variable resolution transforms

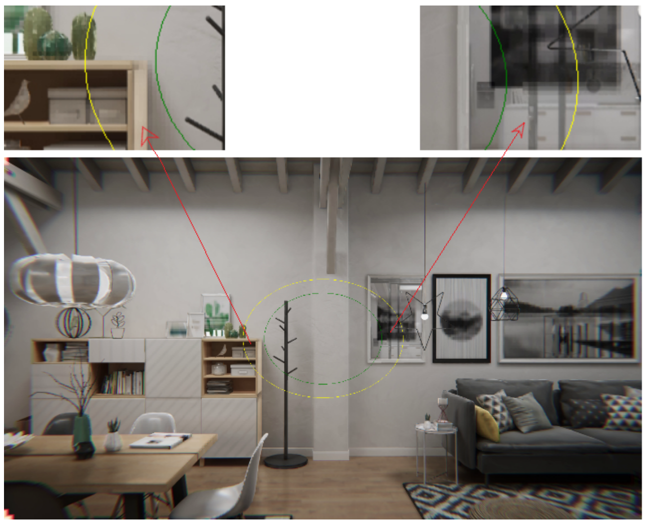


Figure 2: Frame rendered with three shading rates: one shading pass per pixel (inside green ellipse), per 2x2 pixels (between green and yellow ellipses) and per 4x4 pixels (rest).

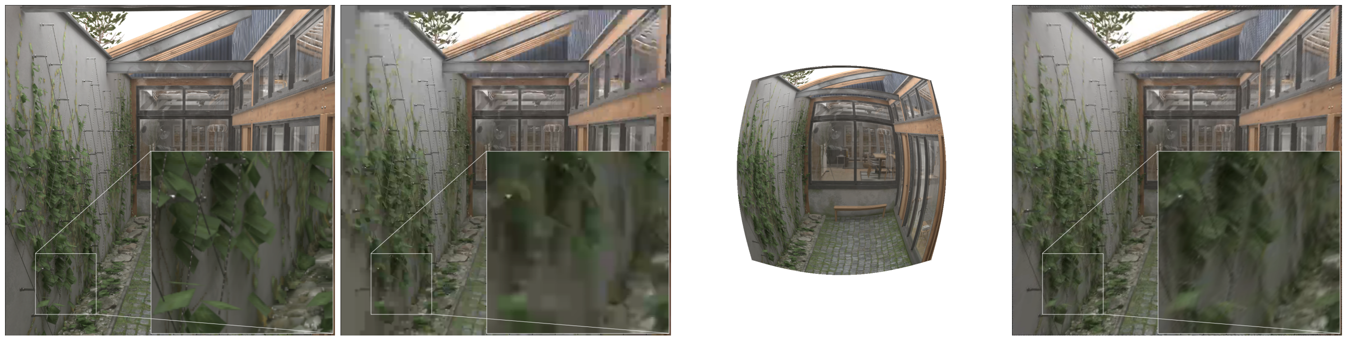
to better simulate the HVS sampling function for foveated imaging applications for example in [7]. Fisheye lenses produce an image with a high resolution in the center and low resolution away from the center. Fisheye lenses are difficult to model numerically, especially in real time rendering applications. Our foveated warping approach could be conceived as a modelling approximation of a custom fisheye lens with an orthographic mapping function which produces an undistorted image in a pre-defined central area and compresses the image in the periphery. This paper develops on our first look into delivering remote rendered foveated rendered graphics efficiently [18] where we investigated interplay of naive (non-anti aliased) foveated rendering and foveated video encoding.

3 FOVEATION METHODS FOR REMOTE RENDERING

The main advantage of foveation in remote rendering systems is the ability to save network bandwidth. It can also reduce the computing workload in the rendering phase, hence enabling potentially higher graphics quality settings for the application or more clients per GPU. In this section, we introduce three different methods to include foveation into a remote rendering system that we call and abbreviate as follows: Foveated Rendering (FR), Foveated Video Encoding (FVE), and Foveated Warping (FW).

3.1 Variable Rate Shading Based Foveated Rendering

FR is applied in the graphics rendering phase, as the name suggests, and the output is a sequence of foveated images delivered as such to an unmodified video encoder. We use the variable rate shading technique to configure shading rates for predefined circular regions specified for different distances from the gaze fixation point. More specifically, we define three regions: a high quality region corresponding to the foveal area, the transition region as a disc



(a) Original Frame (b) Foveated Encoded Frame (c) Foveated Warped Frame (d) Foveated Unwarped Frame

Figure 3: Foveation Samples with zoomed-in regions.

surrounding the foveal area, and the rest of the frame as low quality peripheral region. The high quality region radius is 1/8 of the frame width while as the transition annular region has an inner radius of 1/8 of the frame width and an outer radius of 1/6 of the frame width. Figure 2 shows an overview of a Foveally Rendered scene with three shading rates with decreasing sample rates moving out from the center of the image (gaze location).

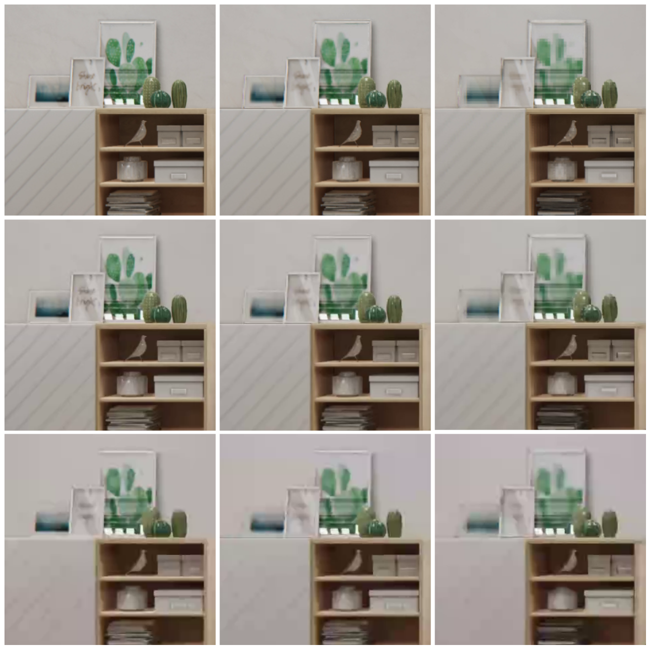


Figure 4: Samples from frames rendered with different shading rates and QP values. Shading rates: one pass per pixel, per 2x2 pixels, per 4x4 pixels (left to right). QPs: 0, 28, 38 (top to bottom).

Figure 4 further illustrates how variable rate shading affects visual quality compared to and combined with quantization performed by a video encoder.

3.2 Foveated Encoding Using Quantization Offsets

We implement FVE by varying the quantization parameter per macroblock within each frame according to a predefined function. We configure the underlying Nvidia hardware h.264 encoder to accept a quantization offset QO for each macroblock of a video frame on top of the quantization coefficient calculated by the rate control algorithm. The quantization offsets are based on the distance of a macroblock from the viewer gaze location. This method allows us to control the quality profile of encoding with fine grained control. Consider a macroblock with normalized coordinates x,y and let x_g and y_g be the normalized coordinates of the gaze fixation. The normalized pixel distance between the macroblock and gaze is then $r = \sqrt{(x - x_g)^2 + (y - y_g)^2}$. We calculate the QO for the macroblock at x, y as :

$$QO_{x,y} = 0, \quad r <= 0.125$$
$$QO_{x,y} = QO_{max}\left(a + b * r^2\right), \quad r > 0.125$$

For an input image of size $1088p \times 1088p$, we use $a = 0.112$ and $b = 2.2063$. Figure 5 illustrates QOs in terms of QO_{max} with respect to normalized distance from gaze fixation. The method is similar to that presented in [17], however there the QO calculation is based on a Gaussian surface centered around the gaze. We use the above parameterization to match the resolution variation implemented in foveated rendering 3.1 and in foveated warping (see Section 3.3)

Figures 3a and 3b show a uniform quality frame and an FVE frame with gaze at the center of the frame.

3.3 Foveated Warping

The idea of Foveated Warping (FW) is to reduce the number of pixels within each frame in a foveated manner, which yields a lower bitrate when encoded to a video. In a sense, we spatially compress the image in proportion of the distance from the gaze fixation point. The name comes from the fact that the operation corresponds to a 2d image warp and we perform it as a post processing step after rendering. Figures 3c and 3d show a sample foveated warped frame with gaze at the center produced at the server side and the same frame unwarped at the client side, respectively.

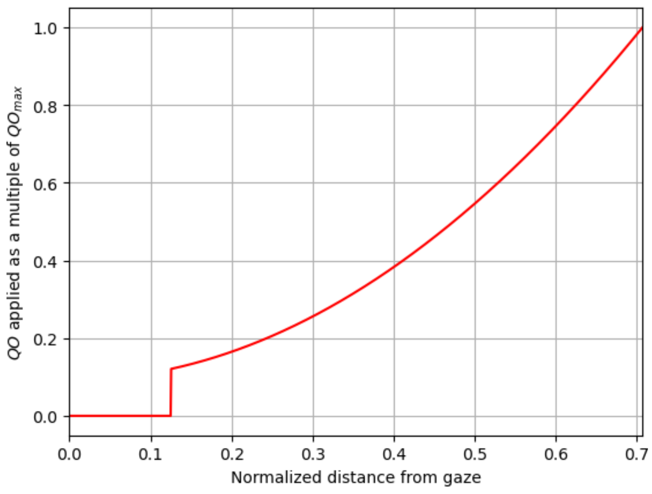


Figure 5: Visualization of quantization offsets QOs with respect to pixel distance from the frame center (gaze) in normalized coordinates

We define FW as a mapping function $FW(I)$ which takes an input image I to produce a foveated warped image O such that each pixel P in O is mapped from n number of pixels in I, n being approximately inversely proportional to the normalized cutoff frequency f_{cn} at pixel P, calculated in a gaze centered coordinate system. Cutoff frequency f_c is the maximum visible spatial frequency at a given pixel and depends on the eccentricity of the pixel from the gaze point and the viewing distance. Cutoff frequency at a pixel can be calculated using the so-called Contrast Sensitivity Function CSF and its inverse Contrast Threshold Function CTF. CSF quantifies the sensitivity of the human eye to contrast in a stimulus and is obtained empirically by pyschophysical experiments [10]. We calculate cutoff-frequency at a pixel using the CSF model provided in [10] and used commonly in video encoding work [46, 47, 49] given by:

$$\frac{1}{CS(f,e)} = CT(f,e) = CT_0 \exp\left(\alpha f \frac{e+e_2}{e_2}\right)$$

where f is spatial frequency in cycles/degree, e is retinal eccentricity in degrees, CT_0 is minimal contrast threshold, α is spatial frequency decay constant, e_2 is half-resolution eccentricity constant, $CS(f,e)$and $CT(f,e)$ are respectively the CSF and CTF for a given f and e. In [10] $\alpha = 0.106$, $e_2 = 2.3$ and $CT_0 = 0.015625$ are reported as the best fitting parameters. The cutoff frequency f_c is then calculated as:

$$f_c = \min\left(\frac{e_2 \ln(\frac{1}{CT_0})}{\alpha(e+e_2)}, \frac{\pi N v}{360}\right) \tag{1}$$

The second term in equation 1 is the display Nyquist frequency in terms of the width of the image N and viewing distance v. When N is measured in pixels and v in picture widths, it corresponds to the maximum spatial frequency that can be displayed by an N pixel wide display without aliasing at a viewing distance of v display widths [49]. Note that the cutoff frequency f_c depends on the retinal eccentricity which depends upon the euclidean distance of a pixel from the gaze fixation and can be calculated as:

$$e(v,x,y) = \arctan\frac{\sqrt{(x-x_g)^2+(y-y_g)^2}}{Nv}$$

where x_g and y_g are coordinates of the gaze fixation point. To illustrate FW consider an image I and gaze fixation point at x_g and y_g. A foveated warped output image O is obtained by transforming the input image I to a polar coordinate system with the gaze point x_g and y_g as the pole, scaling by normalized cutoff frequency f_c and then transforming back to Cartesian coordinates. A pixel with coordinates in the output image O_{x_o,y_o} is mapped from pixels in the input image I_{x_i,y_i} as follows. The target pixel is represented in polar coordinates as r and ϕ given by:

$$\begin{aligned} r &= \sqrt{(x_o-x_g)^2+(y_o-y_g)^2} \\ \phi &= \arctan\frac{(y_o-y_g)}{(x_o-x_g)} \end{aligned} \tag{2}$$

Polar coordinates r and ϕ can easily be converted back to Cartesian coordinates as:

$$\begin{aligned} x_o &= r\cos(\phi) \\ y_o &= r\sin(\phi) \end{aligned}$$

For FW, we scale the radius r according to the normalized cutoff frequency f_{cn} at the output pixel. A pixel in the output image O_{x_o,y_o} is thus obtained as a pixel I_{x_i,y_i} from the input image, where:

$$\begin{aligned} x_i &= \frac{r}{f_{cn_o}}\cos(\phi) \\ y_i &= \frac{r}{f_{cn_o}}\sin(\phi) \end{aligned} \tag{3}$$

where f_{cn_o} is the normalized cutoff frequency at a pixel with coordinates x_o and y_o. While the mapping above corresponds to the cutoff frequency and thus matches the spatial pixel resolution to the acuity of HVS, it is hard to invert i.e obtain I_{x_i,y_i} from O_{x_o,y_o} because of the f_{cn_o} term in Equation 3, which makes it impractical to use in real-time rendering and streaming applications. We model the scaling term r/f_{cn_o} in equation 3 as a parabolic function $p(r) = a + b \times r^2$ so Equation 3 becomes

$$\begin{aligned} x_i &= p(r)\cos(\phi) \\ y_i &= p(r)\sin(\phi) \end{aligned} \tag{4}$$

For the setup described in Section 4.1, we parameterize $p(r)$ so that it roughly matches the quality profile of the FR (see Section 4 4.2. This translates to pixels within the foveal region (i,e $r <= 1/8$ of frame width) in the input image being mapped one to one to the foveal region in the warped image while the rest of the pixels in the input image are scaled. For a frame size of $1088p \times 1088p$, and a viewing distance v of 3.5, we use $a = 0.112$ and $b = 2.2063$, for $r > 1/8$ of the frame width. Figure 6 visualizes how the distance of a pixel from the gaze fixation point decreases as a result of the warp (red curve). We next present the experiment setup used in our measurements in more detail.

4 EXPERIMENT SETUP

In this section we describe the the remote rendering system we use to evaluate the different foveation methods, the experiment setup, and metrics used in the evaluation.

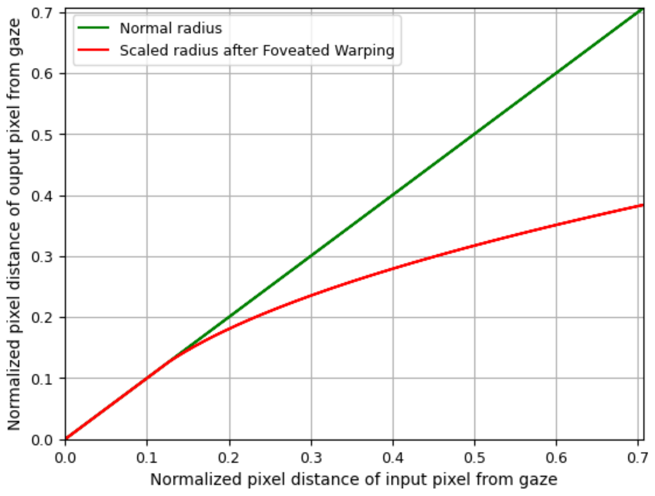


Figure 6: The distance of pixel from gaze fixation point changes with foveated warping (red curve).

4.1 System

We use a Unity game engine -based remote rendering system in our experiments which utilizes gaze data captured from the client. We have developed a server application that is configurable to enable FR and/or FVE with user set parameters at launch time. Although, we used a particular Unity scene with photo-realistic assets to provide a user explorable 3D virtual house, any Unity scene can be deployed in the application without affecting the operation of FR or FVE. An overview of the developed system is presented in Figure 7.

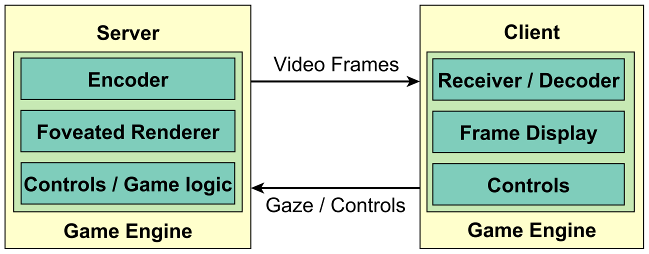


Figure 7: System overview.

The FR module, which performs the foveation is implemented by using the variable rate shading feature of Nvidia VRworks suite of APIs [31] using the Vive Foveated Rendering Plugin for Unity [13]. The Vive Foveated Rendering Plugin allows use of Variable rate shading within Unity as well as provides integration with eye-tracking features of the HTC Vive Pro Eye[1] HMDs. Table 1 lists the shading rate options available. Samples with different shading rates are shown in Figures 2 and 4.

To capture and encode the rendered frames, we use a modified version of the 360 Capture SDK [8]. FVE is implemented by configuring the underlying h.264 hardware encoder by Nvidia with so called QP delta map that specifies the quantization offset QO for each macroblock of a video frame. In the experiments, we use constant QP rate control and these offsets are applied on top of this QP value.

[1]https://www.vive.com/eu/product/vive-pro-eye/overview/

Table 1: Shading Rate Options

Shading Option	Explanation
CULL	no shading
X16 PER PIXEL	16x supersampling
X8 PER PIXEL	8x supersampling
X4 PER PIXEL	4x supersampling
X2 PER PIXEL	2x supersampling
X1 PER PIXEL	1 shading pass / 1 pixel
X1 PER 2X1 PIXELS	1 shading pass / 2 pixels
X1 PER 1X2 PIXELS	1 shading pass / 2 pixels
X1 PER 2X2 PIXELS	1 shading pass / 4 pixels
X1 PER 4X2 PIXELS	1 shading pass / 8 pixels
X1 PER 2X4 PIXELS	1 shading pass / 8 pixels
X1 PER 4X4 PIXELS	1 shading pass / 16 pixels

FW is implemented as a post-processing step for the rendered frame at the server according to the discussion in 3.3. It consists of a blit operation with a single pass fragment shader. The warped frame is then passed on to the video encoder. At the client side, after a warped frame is decoded, we do an inverse transformation with the shader that samples the incoming frame for displaying. Since FW is modelled as a pixel wise invertible function, it can take advantage of parallel processing capacity of GPU of the device. Hence, at server side, the warp operation adds one blit with a fragment shader, which is extremely fast, and at client side it does not result in any additional operations because the input frames need to be sampled anyway in order to be drawn on display. We used both a Windows-build and an Android-build client for our experiments.

For the client-side delay measurements, we utilized an Android-based client and timestamped it to report the decoder delays using two mobile devices: a Pixel 5 mobile phone and an Oculus Quest 2 standalone VR headset.

4.2 Experiments

We simulated client controls programmatically and measured the resulting frame sizes. Unless otherwise stated, the scene camera followed a pre-defined one-minute-long trajectory, simulating a player exploring the scene, in each experiment. As for the point of gaze, it was fixed to the center of the rendered frame in the above described experiments. We used a high-quality architectural visualization scene from Oneiros [32] called *The Scandinavian House demo (AVP Vol.6)* available in the Unity Asset Store [44]. The scene is a fully navigable interior of a scandinavian house that includes more than 200 objects and 4K textures. The target framerate was set to 30 fps in each experiment.

For FR we fix the high quality region to a radius of 1/8th of frame width, the transition region a radius of 1/6th of frame width and the rest as low quality peripheral region. The shading rates for the high, transition and the low quality regions was set to x1 per pixel, x1 per 2x2 pixels and x1 per 4x4 pixels, respectively. Further, we use Multi-Sample Anti Aliasing to ameliorate possible aliasing caused by FR.

For FVE we varied the maximum possible QO (QO_{max}) between 0 to 30 and calculated QO offsets for each macroblock as discussed in Section 3.2: a central circular region region of width 0.25× of frame width is applied with no QO offsets while the macroblocks outside the region are applied with increasing QO offsets, the rate of increase being parabolic with respect to the distance of the macroblock from the frame center, mimicking the variable resolution applied in Foveated Warping. In all experiments, the encoding scheme used was Nvidia's low latency preset[2] and the rate control mode was constant QP.

For FW, we parametrize the warping function as discussed in Section 3.3: pixels within a circular region of width 0.25× of frame width are mapped one to one from the rendered frame to the warped frame while the rest of the pixels are mapped according proportional to the cutoff frequency at the pixel location in the output.

It should be noted that while there is no direct way of making the parameterization of FVE and FW equal, we use the same function to calculate the QO offsets in FVE and strength of warping in FW in order to make the experiments comparable. Further, it should be noted that when we discuss FR or FW in the context of video characteristics and metrics, normal spatially uniform video encoding is implied.

4.3 Metrics

Image and video quality is very subjective and is hence best measured subjectively. Subjective testing, however, is time consuming, expensive and not always feasible. Objective image and video quality metrics attempt to measure quality algorithmically, by e.g finding the difference between an encoded image or video and its original version or by using statistical or other information without using the original version. However, commonly used objective quality metrics like PSNR and MSE do not account for the HVS in general and foveation in particular. While metrics like MSSIM, VMAF [27] try to incorporate properties of the HVS in quality evaluation, they do not account for foveation and hence are not directly usable to evaluate foveated images or video. Various metrics for evaluation of foveated have been developed [25, 38, 46] with varying successes reported by the authors. To evaluate the impact of the studied foveation methods, we use Foveated Structural Similarity Index (FSSIM) [9] based on Multi Scale Structural Similarity Index (MSSIM) [48]. FSSIM attempts to take foveation into account, in addition to being perceptually motivated. We compute FSSIM using only the luminance component (FSSIM-Y) of the encoded images because it is defined over luminance originally and because HVS is more sensitive luminance than color [45]. To complement the results with FSSIM, we also used another objective quality evaluation metric for foveated images based on PSNR, namely FPSNR [9]

Further, to validate the results of our objective evaluation we perform a limited user study to evaluate the quality with FVE and FW. The pilot study is an "Absolute Category Rating: Hidden Reference" study, which provides a Differential Mean Opinion Score (DMOS) of the sequences shown to the subject. In the next section, we utilize the metrics presented here and the experiment setup introduced earlier to measure the impact of the different foveation methods on bitrate, quality and latency.

[2]http://developer.download.nvidia.com/compute/nvenc/v4.0/NVENC_AppNote.pdf

5 RESULTS

In this section, we study the impact of the different foveation methods on video bitrate, objective quality and finally on the latency of the remote rendering system. We also validate the objective quality measurements with a small scale study on subjective quality.

5.1 Impact of Foveated Rendering on Video Bitrate

To investigate how FR impacts video bitrate we ran experiments where we produced foveated frames using FR and the same experiments without foveation, i.e. using normal rendering (NR) with one shading pass per pixel throughout the frame, and encoded the frames to video using different QPs (10,20,25,30).

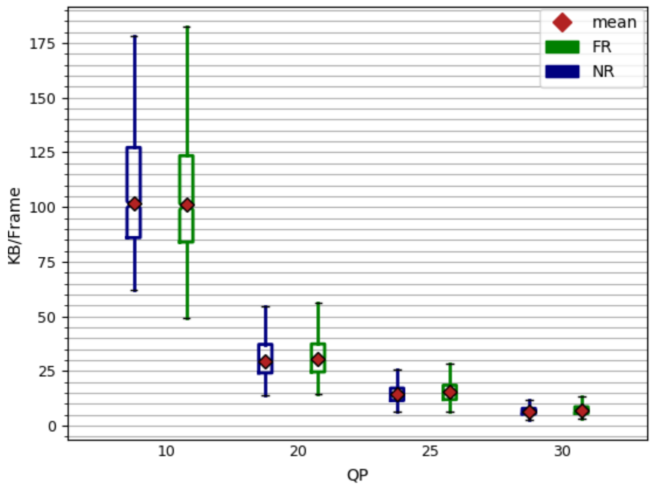


Figure 8: Comparison of frame sizes with foveated rendering (FR) and normal rendering (NR) with different QP values.

The results are plotted in Figure 8. In each case, the average frame size with FR is either larger or equal to the average frame size without foveation. The result is counter-intuitive given that the amount of visual information carried by FR frames is smaller than that carried by NR frames. The reason is that, even though there is less visual information in a frame available with FR, the content of the high quality region changes between successive frames, for example due to camera motion. This increases the dissimilarity between two successive frames rendered with FR when compared to two successive frames rendered with NR. Increase in dissimilarity means that inter-frame compression is less effective and the size of the predicted frames increases, which leads to larger than expected frame sizes with FR. We verify this by redoing some experiments with compressed using only Intra-coding (I-frames). The results reveal the resulting NR frames to be clearly larger than FR frames on average, which confirms our suspicion that the predicted frames are to blame for the larger than expected frames with FR. These results strongly suggest that FR alone is not a good choice to implement foveation when bandwidth is scarce.

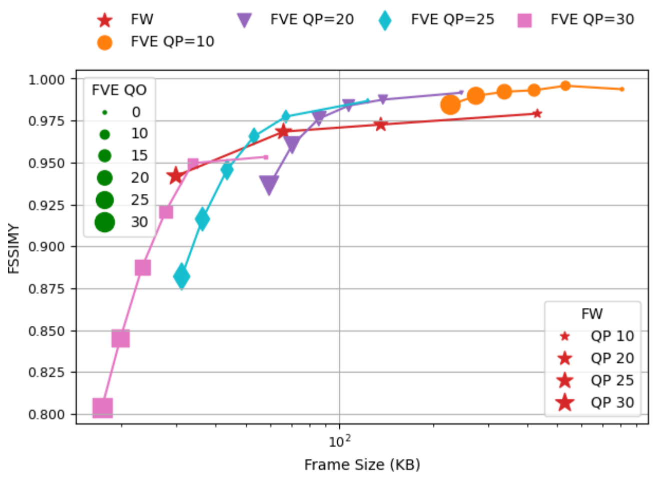


Figure 9: FSSIM-Y and Average Frame Sizes for different Encoding schemes. Note that the condition FVE with QO_{max} = 0 is equivalent to normal encoding.

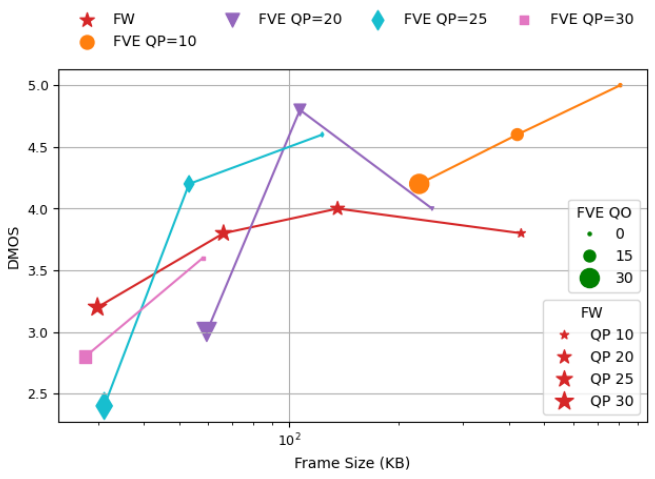


Figure 10: DMOS and frame sizes for different Encoding schemes. Note that the condition FVE with QO_{max} = 0 is equivalent to normal encoding.

5.2 Bitrate vs. Quality

To explore the effect of FVE and FW on video bitrate as well as the objective quality, we conducted multiple experiments to record frame-sizes with different encoding parameters shown in Table 2. For FVE we set the QP values shown in the table permuted with the listed QO_{max} values, while for FW we set the listed QP values. A rendered frame and its corresponding encoded version were periodically saved to disk for objective quality measurements. Note that the case with $QO_{max} = 0$ corresponds to normal non-foveated video encoding.

Table 2: FVE and FW evaluation

Foveation method	FVE,FW
QP	10,20,25,30
QO_{max}	10,15,20,25,30

The results are plotted in Figure 9. Unsurprisingly, the less quantization is applied, the better the quality with both methods. Furthermore, all the configurations of these two foveation methods reduce the resulting bitrate compared to no foveation. However, the main takeaway from these experiments is that, except for the setup having most aggressive quantization, the quality per bit is slightly higher with FVE than with FW across the board. In other words, FVE appears to deliver higher quality with the same bitrate compared to FW. We also calculated FPSNR for these experiments but the results are somewhat mixed. Nevertheless, they also suggest that the quality-per-bit ratio is higher with FVE than with FW.

5.3 Subjective Quality

To supplement the objective evaluation results, we also conducted a limited user study to gain further understanding of the impact of the two methods, FW and FVE, on perceived video quality. Five participants with 20/20 vision or corrected to 20/20 vision participated in the study.

5.3.1 Setup. We installed the server and client described in Section 4.1 on a powerful workstation equipped with Intel Xeon W-2133 (3.2Ghz) CPU, Nvidia GeForce RTX 2060 GPU, 32 GB of Memory and a solid state drive. This allowed us to eliminate impact of any network related distortions. The server rendered the same scene as in the other experiments with foveated rendering enabled. Multi-sample Anti-aliasing (8x Multisampling) was used on the server to ameliorate possible aliasing artefacts introduced by FR. The client was configured to receive video frames without buffering from the server. Video was encoded at 30fps at the server and displayed on a Dell U2715H display.

5.3.2 Procedure. We followed the ITU-T recommendations [35] in the user study design. Specifically, we used Absolute Category Rating-Hidden Reference (ACR-HR) as the rating method because our goal was to compare the impact of FW and FVE on subjective video quality. The test sequence was a 30 second exploration of the scene along a fixed trajectory. An encoding profile of $QP = 10$, $QO_{max} = 0$ was used as the reference condition. At $QP = 10$ encoding is generally considered to be visually lossless. The various test conditions, shown in table 3 were presented in random order to each subject to minimize carryover effects. The subjects were placed at a distance of 3.5 times the frame width and the test sequence ran for 30 seconds. They were instructed to focus their gaze on the center of the frame while viewing the test sequences. To help them do this, a grey frame with a red dot in the center was displayed for 5 seconds before displaying a test sequence. At the end of each test, the subject was asked to the rate the quality of the video on a scale of 1 to 5, with a slider on the screen.

5.3.3 Results. The results of the user study are shown in Table 3 and plotted with frame sizes of the respective test cases Figure 10. The DMOS broadly agrees with FSSIMY for the respective conditions. The DMOS values at a particular QP for FVE are higher than the corresponding case of FW when QO_{max} <= 15. Surprisingly, the FVE case having $QP = 20$ yields higher DMOS with

Table 3: User Study

Condition			DMOS
Foveation	QP	QO_{max}	
FW	10	0	3.8
FW	20	0	4
FW	25	0	3.8
FW	30	0	3.2
FVE (Reference)	10	0	5
FVE	10	15	4.6
FVE	10	30	4.2
FVE	20	0	4
FVE	20	15	4.8
FVE	20	30	3
FVE	25	0	4.6
FVE	25	15	4.2
FVE	25	30	2.4
FVE	30	0	3.6
FVE	30	15	2.8

Table 4: Parameters varied in the encoding / decoding measurements.

Foveation method	Foveated Encoding / Foveated Warping
Client device	Google Pixel 5 / Oculus Quest 2
Render resolution	1088x1088 / 2176x2176 / 3264x3264
Server GPU	Nvidia RTX 2060 / RTX 2080 Ti / Titan V

$QO_{max} = 15$ than with $QO_{max} = 0$. The same applies for FW with $QP = 10$ vs. $QP = 20$. These discrepancies are likely to be caused by temporal distortions which are not captured by image quality metrics, such as FSSIMY. Looking at Figure 10, the FVE condition $QP = 20, QO_{max} = 15$ is the best performing condition when considering both quality and bitrate.

5.4 Latency

FW allows the system to pack the rendered frames into a smaller resolution video size compared to FVE without compromising the quality perceived by the client too much as observed in the previous sections. This has the potential to reduce the end-to-end latency of a remote rendering system by offering lower delay on both the server during video encoding and the client during video decoding. We utilized the same ArchVizPro Vol 6 scene in our latency measurements as we did for the quality measurements and averaged the results over three runs for each test case both in video encoding and decoding.

We tested the encoding speeds of three GPUs (Nvidia RTX 2060, Nvidia RTX 2080 Ti and Nvidia Titan V) with three render resolution scenarios (1088x1088, 2176x2176 and 3264x3264) both with foveated warping and foveated encoding. For foveated encoding, the sent video resolution matches the render resolution. However, for foveated warping the resulting video sizes are 704x704, 1408x1408 and 2112x2112. The encoding delay results are presented in Figure 11 and the parameters varied for the encoding and decoding experiments are summarized in Table 4.

All the three GPUs show a similar growth in delay when the render resolution is increased. The results show that the encoding delay can quickly double with each resolution tier with foveated encoding. The foveated warping method has the advantage of resulting in lower video resolutions which translates to lower video encoding times. Overall, the foveated warping method introduces 31-67% lower encoding times than foveated encoding with matching render resolutions.

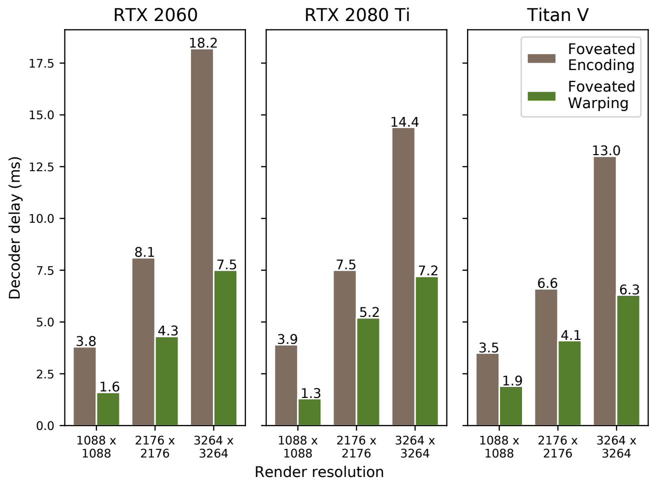


Figure 11: Encoder delay using Foveated Encoding or Foveated Warping with different server render resolutions and GPUs.

Higher resolution video frames also increase the decode time on the client. We verified this presumption by measuring the decode times on two different resource-constrained client devices using both the FW method and FVE. The results using matching video resolutions for both methods on a Pixel 5 mobile phone and an Oculus Quest 2 standalone VR headset are presented in Figure 12.

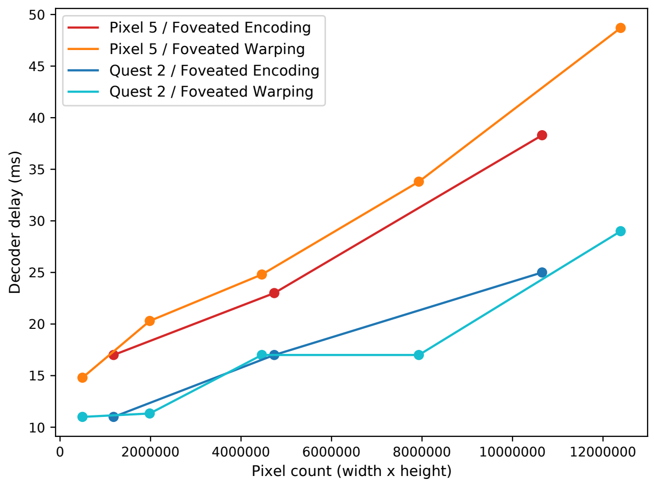


Figure 12: Decoder delay when increasing pixel count using Foveated Encoding or Foveated Warping.

The results show that both methods follow a fairly linear increase in decoding delay when the resolution (pixel count) increases. The more powerful SoC (Qualcomm Snapdragon XR2) of the Oculus Quest 2 shows a less steep curve when the resolution increases compared to the Qualcomm Snapdragon 765G in the Pixel 5. The difference in decoding delay comparing the two methods is within a few milliseconds. However, FW can pack the video frames in lower overall resolution. This results to lower decoding speeds on the client-side for a given render resolution. Figure 13 shows the decoding delay benefits with three different render resolution scenarios.

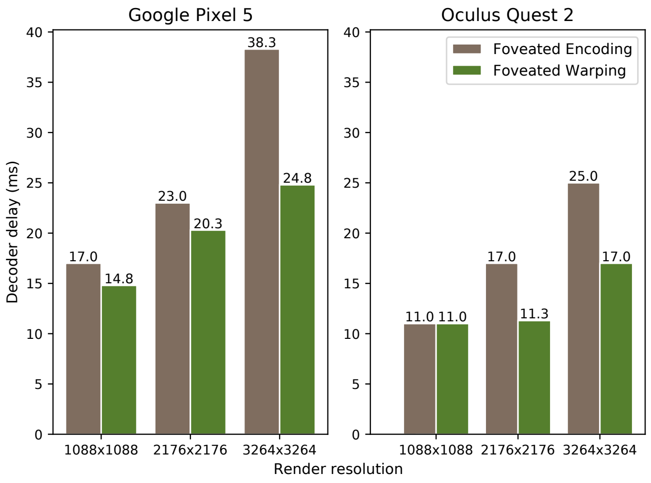


Figure 13: Decoder delay using Foveated Encoding or Foveated Warping with different server render resolutions.

The results show that the end device can have a big impact on the client-side latency in remote rendering systems. The standalone VR headset Oculus Quest 2 yields latencies that are 25-44% shorter than with the Google Pixel 5. In addition, foveated warping leads to up to 35% decrease in latency thanks to the lowered frame resolution. In the next section, we discuss the results in the context of choosing the right method for foveated remote rendering in distributed systems.

6 DISCUSSION

It is clear that FR alone is not a solution to the bandwidth consumption problem in remote rendering. While the two other methods, FW and FVE, do mitigate this problem, it is not straightforward to decide which of them is better. The results shown in previous sections suggest that FVE provides higher quality per bit than FW. Therefore, it would seem that FVE should be favoured. However, FW provides significantly lower end-to-end latency by reducing the video encoding and decoding delay, which is also a critical performance metric in remote rendering. Furthermore, FW enables delivering foveated frames with higher target resolution than FVE because of the limitations of video codecs. The benefits for both foveated encoding and warping are roughly visualized in Figure 14.

Balancing between different constraints in remote rendering systems is difficult as the service provider needs to be aware of the entire pipeline from the server to the client. The encoding scheme

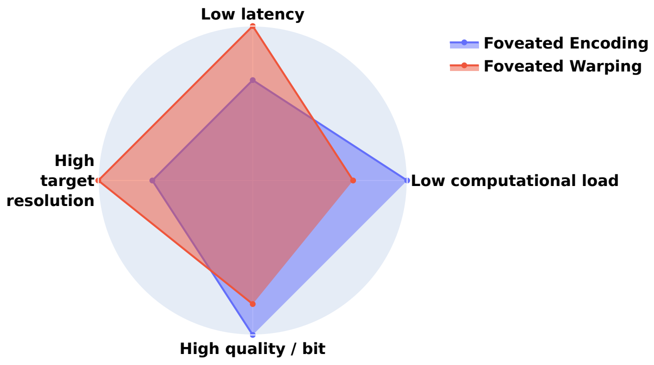


Figure 14: Comparison of the benefits of Foveated Encoding and Foveated Warping.

is one of the many decisions needed to be made in such systems. The targeted end-user device might have computational limitations regarding for example the resolutions that its decoder can handle. On the other hand, the end-user device could be for example a standalone VR headset with a large native eye resolution requiring high target resolution also from the remote rendering system. These two scenarios favor the FW method for its capability to target high render resolutions with low latency as seen in our delay measurements. However, a remote rendering system provider might also have to optimize the quality / bit ratio to minimize the costs both for the provider and the end-user. An optimization decision favoring FVE over FW would be also to minimize the computational load on the server by not targeting as high resolutions as would be possible with FW.

In this paper, we analyzed the objective quality of different methods of foveation in remote rendering. We also conducted a small-scale user study to confirm the objective measurements as the metrics can often be different in user studies. Although the initial user study seems to confirm the objective measurements, deep understanding of how the different factors affect user perceived quality with foveated video requires additional experiments. Some factors that we did not yet consider in this work include the impact of resolution vs. encoding quality (quantization) on perceived quality, which needs to be fully understood before anything conclusive can be stated on which scheme is better in a particular situation. We also need to examine how effective different anti-aliasing methods are with both methods, particularly temporal vs. spatial AA. An extensive user study is planned as part of our follow-up work.

Another dimension that we did not touch in this paper is the type of user interface, especially VR vs. "normal" display. Foveation is potentially more beneficial with VR because of the relatively short viewing distance, which allows more aggressive foveation. However, adapting the presented methods for VR needs to take the VR display specifics into account, such as the impact of lens distortion and its compensation, virtual image distance and pixel density. This is also part of our future work.

The impact of delay in foveated rendering/streaming is a topic that will need to be studied separately, as e.g. done in [1]. We did not evaluate rendering delay at all because that is affected by so many different factors. There decoding and encoding delays have a baseline value purely dependant on the underlying hardware as we use hardware encoder and decoder on server and client devices, respectively. The client is a mobile device (Pixel 5 and Oculus Quest) and we apply nontrivial video resolutions in these tests, which explains why the decoder may struggle with high resolutions. It is possible to somewhat reduce the encoding delay by splitting frames and running several parallel encoders [28]. However, with consumer grade Nvidia cards, the number of parallel streams is limited, and it may not help the decoding operation that is run by the much more constrained client device.

The current implementation of the Foveated Warping method is a post processing step. Although this step only incurs a small additional computation cost on the server after rendering, it is most likely possible to implement it also within the rendering pipeline in the similar fashion than the authors propose to do their custom projection in [36]. This would further optimize the rendering part of the pipeline at the server, which would make the Foveated Warping method even more attractive for foveated streaming of real-time graphics.

7 CONCLUSION

This paper reports results from a study of different methods to stream foveated real-time remote rendered graphics. The results suggest that foveated rendering alone does not decrease bandwidth consumption when encoded to a video, it even increases the video bitrate in certain cases. However, foveated video encoding (FVE) as well as a novel method we call foveated warping (FW) do. Their comparison suggests that FVE delivers higher quality per bit consumed, whereas FW may be a more attractive solution for resource constrained clients because it foveates frames by reducing the number of pixels, instead of bits encoded per pixel as FVE does, which yields lower video coding resolution and latency for a particular target render resolution and enables higher target resolution when the supported video encoding/decoding resolution on device hardware is limited. For future work, we intend to study alternative methods of warping and conduct an extensive user study including VR setup with real time gaze tracking based foveation.

ACKNOWLEDGMENTS

We would like to thank Enrico Masala for valuable insights into video compression. This work has been financially supported by the Academy of Finland (grant number 332306) and the Nokia Center for Advanced Research .

REFERENCES

[1] Rachel Albert, Anjul Patney, David Luebke, and Joohwan Kim. 2017. Latency Requirements for Foveated Rendering in Virtual Reality. *ACM Trans. Appl. Percept.* 14, 4, Article 25 (Sept. 2017), 13 pages. https://doi.org/10.1145/3127589

[2] Anup Basu, Allan Sullivan, and Kevin Wiebe. 1993. Variable resolution teleconferencing. In *Proceedings of IEEE Systems Man and Cybernetics Conference-SMC*, Vol. 4. IEEE, 170–175.

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

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

[5] Nvidia Corporation. [n.d.]. *VRWorks - Variable Rate Shading (VRS)*. Retrieved March 8, 2021 from https://developer.nvidia.com/vrworks/graphics/variablerateshading

[6] A Cowey and ET Rolls. 1974. Human cortical magnification factor and its relation to visual acuity. *Experimental Brain Research* 21, 5 (1974), 447–454.

[7] George Curatu and James E Harvey. 2008. Lens design and system optimization for foveated imaging. In *Current Developments in Lens Design and Optical Engineering IX*, Vol. 7060. International Society for Optics and Photonics, 70600P.

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

[9] Andrew Floren and Alan C Bovik. 2014. Foveated Image and Video Processing and Search. In *Academic Press Library in Signal Processing*. Vol. 4. Elsevier, 349–401.

[10] Wilson S Geisler and Jeffrey S Perry. 1998. Real-time foveated multiresolution system for low-bandwidth video communication. In *Human vision and electronic imaging III*, Vol. 3299. International Society for Optics and Photonics, 294–305.

[11] 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.

[12] G. Hesina and D. Schmalstieg. 1998. A network architecture for remote rendering. In *Proceedings. 2nd International Workshop on Distributed Interactive Simulation and Real-Time Applications (Cat. No.98EX191)*. 88–91. https://doi.org/10.1109/DISRTA.1998.694570

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

[14] 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

[15] 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

[16] Gazi Illahi, Matti Siekkinen, and Enrico Masala. 2017. Foveated Video streaming for cloud gaming. In *2017 IEEE 19th International Workshop on Multimedia Signal Processing (MMSP)*. 1–6. https://doi.org/10.1109/MMSP.2017.8122235

[17] 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.

[18] 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*. 1–3.

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

[20] 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

[21] 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.

[22] 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.

[23] 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).

[24] Sanghoon Lee and Alan C Bovik. 2003. Fast algorithms for foveated video processing. *IEEE Transactions on Circuits and Systems for Video Technology* 13, 2 (2003), 149–162.

[25] Sanghoon Lee, Marios S Pattichis, and Alan C Bovik. 2002. Foveated video quality assessment. *IEEE Transactions on Multimedia* 4, 1 (2002), 129–132.

[26] Yong Li and Wei Gao. 2019. DeltaVR: Achieving high-performance mobile VR dynamics through pixel reuse. In *Proceedings of the 18th International Conference on Information Processing in Sensor Networks*. 13–24.

[27] Zhi Li, Anne Aaron, Ioannis Katsavounidis, Anush Moorthy, and Megha Manohara. 2016. Toward a practical perceptual video quality metric. *The Netflix Tech Blog* 6, 2 (2016).

[28] 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

[29] 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

[30] 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.

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

[32] OneirosVR. 2019. ArchVizPRO Interior Vol.6. https://assetstore.unity.com/packages/3d/environments/urban/archvizpro-interior-vol-6-120489https://oneirosvr.com/portfolio/archvizpro/

[33] 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).

[34] 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.

[35] ITUT Recommendation. 2008. P. 910,"Subjective video quality assessment methods for multimedia applications,". *International Telecommunication Union, Tech. Rep* (2008).

[36] Bernhard Reinert, Johannes Kopf, Tobias Ritschel, Eduardo Cuervo, David Chu, and Hans-Peter Seidel. 2016. Proxy-guided Image-based Rendering for Mobile Devices. *Computer Graphics Forum* (2016).

[37] Lee Ann Remington and Denise Goodwin. 2011. *Clinical anatomy of the visual system E-Book*. Elsevier Health Sciences.

[38] Snježana Rimac-Drlje, Mario Vranješ, and Drago Žagar. 2010. Foveated mean squared error—a novel video quality metric. *Multimedia tools and applications* 49, 3 (2010), 425–445.

[39] 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.

[40] 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.

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

[42] Shu Shi and Cheng-Hsin Hsu. 2015. A Survey of Interactive Remote Rendering Systems. *ACM Comput. Surv.* 47, 4, Article 57 (May 2015), 29 pages. https://doi.org/10.1145/2719921

[43] 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.

[44] Unity. 2021. Unity Asset Store - ArchVizPRO Interior Vol.6. https://assetstore.unity.com/packages/3d/environments/urban/archvizpro-interior-vol-6-120489https://oneirosvr.com/portfolio/archvizpro/

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

[46] Zhou Wang and Alan C Bovik. 2006. Foveated image and video coding. In *Digital Video, Image Quality and Perceptual Coding*, Hong Ren Wu and Kamisetty Ramamohan Rao (Eds.). CRC Press, Boca Raton, 431–457.

[47] 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

[48] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. 2003. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, Vol. 2. Ieee, 1398–1402.

[49] Zhou Wang and A. C. Bovik. 2001. Embedded foveation image coding. *IEEE Transactions on Image Processing* 10, 10 (2001), 1397–1410. https://doi.org/10.1109/83.951527