Power, Performance, and Image Quality Tradeoffs in Foveated Rendering

Rahul Singh *1, Muhammad Huzaifa †1, Jeffrey Liu ‡1, Anjul Patney §2,
Hashim Sharif ¶1, Yifan Zhao ‖1, and Sarita Adve **1

1University of Illinois Urbana-Champaign
2NVIDIA

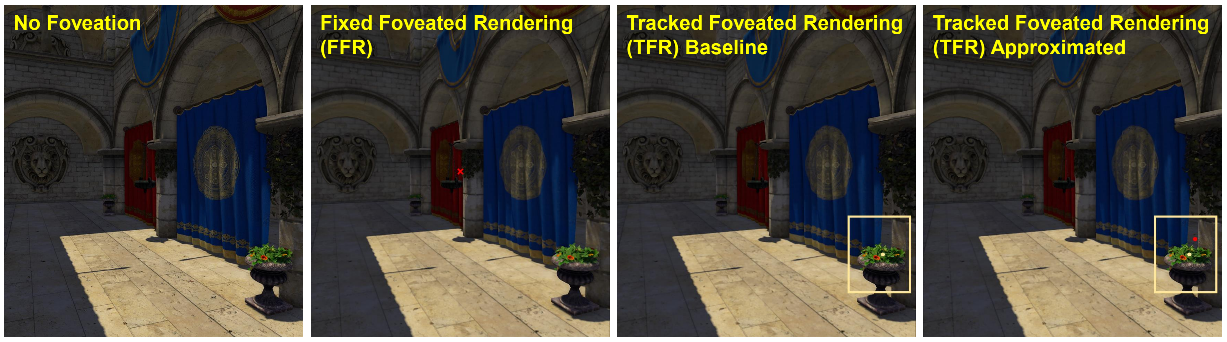

No Foveation
Fixed Foveated Rendering (FFR)
Tracked Foveated Rendering (TFR) Baseline
Tracked Foveated Rendering (TFR) Approximated

Figure 1: From left to right: Sponza rendered with non-foveated rendering, Fixed Foveated Rendering (FFR) (the user's gaze is always assumed to be in the center at the cross), Gaze-Tracked Foveated Rendering (TFR) using a baseline gaze-tracking model, and Gaze-Tracked Foveated Rendering using our approximated gaze tracking model. With FFR, a user gazing at the plant cannot perceive many details of the plant since the actual gaze is far from the gaze assumed by the system. On the other hand, with baseline TFR, the system correctly identifies the user's gaze on the plant (the yellow dot), and the plant is now in full detail. With our approximated gaze tracking model, we achieve a 20× improvement in energy efficiency and almost a 9× speedup of the gaze-tracker. More importantly, we maintain the quality by choosing an appropriate foveation level. Hence, our technique operates at much lower energy while maintaining acceptable image fidelity.

Abstract

Extended reality (XR) devices, including augmented, virtual, and mixed reality, provide a deeply immersive experience. However, practical limitations like weight, heat, and comfort put extreme constraints on the performance, power consumption, and image quality of such systems. In this paper, we study how these constraints form the tradeoff between Fixed Foveated Rendering (FFR), Gaze-Tracked Foveated Rendering (TFR), and conventional, non-foveated rendering. While existing papers have often studied these methods, we provide the first comprehensive study of their relative feasibility in practical systems with limited battery life and computational budget.

We show that TFR with the added cost of the gaze-tracker can often be more expensive than FFR. Thus, we co-design a gaze-tracked foveated renderer considering its benefits in computation, power efficiency, and tradeoffs in image quality. We describe principled approximations for eye tracking which provide up to a 9× speedup in runtime performance with approximately a 20× improvement in energy efficiency when run on a mobile GPU. In isolation, these approximations appear to significantly degrade the gaze quality, but appropriate compensation in the visual pipeline can mitigate the loss. Overall, we show that with a highly optimized gaze-tracker, TFR is feasible compared to FFR, resulting in up to 1.25× faster frame times while also reducing total energy consumption by over 40%.

*e-mail: rahuls10@illinois.edu
†e-mail: huzaifa2@illinois.edu
‡e-mail: jliu179@illinois.edu
§e-mail: apatney@nvidia.com
¶e-mail: hsharif3@illinois.edu
‖e-mail: yifanz16@illinois.edu
**e-mail: sadve@illinois.edu

Index Terms: Human-centered computing—Visualization—Visualization techniques—Treemaps; Human-centered computing—Visualization—Visualization design and evaluation methods

1 Introduction

Augmented, virtual, and mixed reality (AR/VR/MR), collectively referred to as extended reality (XR), enables highly immersive visual experiences for its end users. Achieving high-quality visuals at high resolutions, frame rates, and view counts poses a challenge for system designers because these highly compute-intensive workloads must often run on heavily resource-constrained devices. The challenge gets more interesting as there is a variety of subsystems in an XR system fighting for the same resources.

Many XR systems utilize foveated rendering to improve their rendering efficiency. This method works by adapting the quality of a rendered image over a user's visual field of view, where only a small region around the center of their gaze maintains the highest quality. Since humans have reduced visual acuity in peripheral vision, such adaptation has little impact on perceived quality. Thus, it can theoretically provide significant improvements in rendering efficiency. [8, 12, 19]

There are mainly two flavors of foveated rendering, Fixed Foveated Rendering (FFR) and Tracked Foveated Rendering (TFR).

FFR does not utilize eye tracking; instead, it relies on the assumption that users generally focus on the center of the screen. Thus, it renders the center of the screen at a high quality while reducing the resolution at the corners and edges of the screen. FFR is supported by Meta's Oculus Quest 2 and other newer headsets by HTC and Sony [11,23,32]. However, there are two drawbacks to this approach. First, the user may not always look at the center of the screen. In this case, the reduced visual fidelity on the edges may result in noticeably lower quality, especially if there are salient visual elements like text. Second, the size of the high-fidelity central foveal region generally must be large to compensate for this assumption.

TFR aims to solve this problem by tracking the user's gaze and shifting the high-fidelity foveal region accordingly. By tracking the user's gaze, TFR can use a smaller high- fidelity region while still maintaining quality, even if the user is looking at corners or edges of the display. Headsets like HTC Vive and Sony's PSVR claim to have achieved significant benefits in GPU frame times through TFR. In order to be visually effective, TFR requires a fast and accurate gaze-tracker. Recent advances in gaze-tracking [4, 40] demonstrate that the highest quality gaze-trackers utilize deep neural networks running at a high frame rate. Running a deep neural network for every frame can be expensive for an XR device, so for TFR to be more efficient than FFR, the savings must outweigh the added computational cost of gaze-tracking.

Most existing headsets that support TFR are tethered and run on a high-end GPU [11, 23, 32]. For low-power GPUs on an untethered VR headset, running a gaze-tracker can be very expensive [23]. Hence, it is important to evaluate whether TFR is feasible on a low-power XR system. To answer this question, we perform a study of non-foveated rendering, FFR, and TFR using parameters like performance (frame time and end-to-end latency), power/energy efficiency, and image quality. Based on this study, we co-design the gaze-tracker and foveated renderer by applying several foveated renderer-aware optimizations in the gaze-tracker to reduce its cost. Specifically, our contributions are:

1. A detailed study of Non-foveated (Regular) Rendering, Fixed Foveated Rendering (FFR), and Tracked Foveated Rendering (TFR) based on end-to-end performance, power, and quality. We analyze the circumstances where foveated rendering is beneficial.

2. A set of experiments demonstrating that the latency and power costs added by a gaze-tracker are non-trivial. TFR can often be more expensive than FFR and must be carefully designed.

3. An end-to-end system co-design using approximation techniques for gaze-tracking and foveated rendering, which reduces the overall cost of TFR below FFR while maintaining the Quality of Experience (QoE). Here we deal with the challenging task of balancing multiple parameters like end-to-end latency, frame rate, energy per frame, and QoE.

Overall, this paper provides a cost-benefit analysis of different types of foveated rendering, and co-designs a gaze-tracker and foveated renderer for future VR headsets. It provides a path to make TFR beneficial for visual quality, latency, and power consumption.

2 Related Works and Background

2.1 Foveated Rendering

Foveated rendering [8, 12, 19] is a technique that renders a high-quality image in a region of interest while reducing quality in other areas. This region is typically based on the user's gaze because humans are less sensitive to lower quality in their peripheral vision. Guenter et al. [8] and Vaidyanathan et al. [35] proposed techniques to extract performance benefits using foveated rendering. Later, Patney et al. [29] suggested post-processing after rendering to reduce temporal and spatial artifacts caused by foveation in peripheral vision. Albert et al. [2] studied how foveated rendering is sensitive to latency. They described the maximum latency budget in an end-to-end foveated rendering system, beyond which the user will begin noticing the quality loss caused by foveation. Another study was carried out by Hsiao et al. [5] in the context of gaze-contingent video streaming and foveated compression techniques. The work by Aksit et. al. [1] deals with designing and rapidly manufacturing both unfoveated and foveated AR displays, whereas the work by Kim et. al. [14] discusses the design of a foveated display tightly coupled with a simple rendering algorithm using on-axis gaze-tracking. Both these works deal with foveated displays and corresponding rendering algorithms. Mohanto et. al. [26] provide a detailed survey on foveated rendering techniques with different evaluation techniques focused on quality. They discuss the merits and demerits of multiple techniques but do not perform any measurements.

To the best of our knowledge, no previous work has presented a detailed end-to-end analysis of how beneficial foveated rendering can be in terms of performance, power, and quality. We compare non-foveated (Regular) rendering with FFR and TFR, and do a cost-benefit analysis between the three techniques by taking an end-to-end system into account.

We use NVIDIA's Variable Rate Shading (VRS) SDK demo as our rendering platform because it is a freely available foveated rendering platform. It also enables us to choose different levels of Coarse Pixel Shading (CPS) for foveation. While foveating, it divides the scene into three concentric ovals. The innermost region is rendered with the highest shading rate, the middle region is rendered with 2×2 coarse pixel shading ($4\times$ less work), and the outer region is rendered with 4×4 coarse pixel shading ($16\times$ less work). It allows us to change the shading rates and radii of these regions, so the degree of foveation can be customized for specific scenes and performance requirements.

2.2 Gaze-tracking

In an XR system, two cameras are typically used to capture the eye images of the user. These eye images are fed to a neural network, which outputs a gaze vector. We use a segmentation network to segment the eye images into pupil, iris, and sclera. We then use the segmentation result to calculate the center of the pupil. Finally, the user's gaze is extrapolated from the pupil center and is represented as the gaze vector [20]. In the case of TFR, this gaze estimation is then passed to the rendering component. We use RITNet [4] for the segmentation stage. It primarily consists of CNNs with more than 40 layers of convolution. For the extrapolation stage, we use DeepVOG [38] to compute the final gaze vector. There have been several optimizations suggested for gaze-tracking, such as event-driven segmentation by Feng et al. [7] and EyeCoD [39] by You et al. The event-driven segmentation work focuses on reducing the gaze-tracker computation by predicting a region of interest in the eye image before providing it to the gaze estimator. EyeCoD focuses on near-sensor computing and designs custom hardware for the gaze-tracker. Both these works optimize the gaze-tracker in isolation and their main focus is gaze error (degree error). The focus of our work is to compare different configurations of foveated rendering with non-foveated rendering and co-design the gaze-tracker with the renderer. We show that we can employ aggressive optimizations without hardware changes if we account for foveated rendering parameters when co-designing the gaze-tracker.

2.3 Quality Metric:

Optimizing a foveated renderer requires a model of human foveal-peripheral vision to help guide design choices without conducting expensive user studies. Traditional quality metrics like SSIM [36] and HDR-VDP-2 [21] perform well for static images. However, for foveated rendering, we emphasize an area of interest and can tolerate a graceful degradation of quality in the periphery. Both SSIM and HDR-VDP fail to capture this degradation of quality in the periphery. Other metrics like FWQI [37], FA-SSIM [31], and FSNR [16] are focused on foveated images. Swafford et. al. [34] compare these techniques and discuss the shortcomings of each. FSNR is a cumulative error metric without any perceptual information. Both FWQI and FA-SSIM are meant for static images and ignore any temporal information. They also don't consider the display parameters, which are now important due to the variety of displays; especially Near Eye Displays. The metric suggested by [34] is an extension of HDR-VDP2 and suffers from similar drawbacks of focusing on static images and ignoring display parameters.

We utilize **FovVideoVDP** [22] for our experiments, which is a recent and state-of-the-art quality metric for foveated image viewing. The metric considers the degradation of human visual perception across the field of view and helps identify spatial and temporal artifacts. FovVideoVDP takes a test and reference video input to compute a perceptual Just-Objectional-Difference (JOD) rating as a measure of the test image's quality compared to the reference. The JOD rating is computed on a scale of 10, where a higher JOD number corresponds to better quality. Note that a JOD of 9 indicates that 75% of human observers can perceive the difference between the test and reference images.

In Section 5.1, we present some of our conclusions from using this metric, as well as observations that indicate the need for further research in the area of peripheral image quality metrics.

We use another **intermediate quality metric** called degree tracking error to quantify the gaze-tracking error. Degree tracking error only targets the gaze-tracker and not the end-to-end system, so we limit its use to reducing the search space while optimizing the gaze-tracker.

2.4 Latency Requirements of a Gaze-tracked Foveated Rendering System

TFR requires gaze-tracking and is thus sensitive to latency. Prior work has shown that VR can tolerate up to 50 ms of total eye motion-to-photon latency [2]. This latency includes the time from eye movement to the time when the frame with the latest gaze position is displayed. Out of this 50 ms budget, **the gaze-tracking latency needs to be less than 15 ms** [33]. Lower gaze-tracking latencies yield higher system efficiencies and better user experience [10].

The following steps need to be performed within the gaze-tracker's 15 ms budget: sampling the eye movement, processing the camera image, transferring the image to the host system, running segmentation on the image (typically on a GPU), estimating the gaze using the inference result (typically on a CPU), filtering the gaze to reduce jitter, and sending the result to the foveated renderer. The slowest parts of this pipeline are typically the inference and gaze estimation steps. While inference runs fast on desktop GPUs, it is slow on mobile GPUs and can become the bottleneck. Furthermore, the remaining steps also consume a significant portion of the budget, making it difficult or impossible to stay within 15 ms on mobile platforms. In the next section (2.5), we discuss how this bottleneck can be alleviated without requiring hardware changes.

2.5 Structured Pruning of Neural Networks

To create variants with different accuracy and performance tradeoffs, we apply different levels of weight pruning to the RITNet CNN used in the gaze tracker. Weight pruning is a popular neural network compression technique that removes weights with a lower contribution toward the end-to-end classification/detection result. Weight pruning also requires model retraining to recover the accuracy lost due to pruning. Pruning is usually performed iteratively - each iteration removes a certain percentage (say X%) of the weight parameters, after which the model is retrained for a few epochs. Here "X" is a tunable parameter: the higher the X, the higher the approximation. Each pruning iteration progressively removes a higher fraction of weights, which improves performance but also degrades accuracy (effectively reducing classification/detection potential). The pruning variant that delivers the best accuracy-performance tradeoff is application/neural-network specific since different neural networks behave differently to pruning.

Weight pruning can be broadly categorized as: 1. *unstructured pruning* [9, 15, 41], which removes individual weights that are deemed less important to the overall computational accuracy (e.g, low-magnitude values), or 2. *structured pruning* [3,17,18,27], which removes groups of contiguous weights, e.g., entire filters and channels from convolution layer weights. Unstructured pruning provides a much higher reduction in model sizes compared to structured pruning (e.g., 13× in [9] vs. 4.5× in [18]). However, it also results in unpredictable sparsity that is much less efficient for highly parallel architectures, such as GPUs and CPU vector units (Cortex-A72 vector units in Pi4 [6]), which are not well-suited for irregular computational patterns.

Our implementation is based on the iterative structured pruning algorithm proposed by Renda et al. [30] and the L-norm based filter pruning approach proposed by Li et al. [17]. In each pruning iteration, we remove 20% of the filters with the lowest L1-norm sum of all its weights. This heuristic works on the assumption that low-magnitude weights contribute less to end-to-end classification results.

3 Co-designing Gaze-tracker and Foveated Rendering

In Section 5.1, we show that TFR in isolation provides speedups in rendering time compared to FFR, but consumes more energy when considered with the gaze-tracker. This makes TFR infeasible in an energy-constrained environment such as a head-mounted display. In our work, we propose a co-design of the gaze-tracker with the renderer to bring down the total energy cost of TFR compared to FFR.

We propose approximating the gaze-tracker to improve its performance and energy cost. Our goal is to do so without compromising end-to-end quality. Approximating the gaze-tracker in isolation causes intermediate errors. We mitigate these errors in the co-designed foveated renderer by manipulating the radii of foveated regions. As a result, the end-to-end user experience is not affected. This kind of optimization is not possible when looking at the sub-systems in isolation. In our work, accounting for the entire system enables us to use an aggressively optimized approximation in one sub-system while compensating for errors using a cheaper technique in the next sub-system.

3.1 Optimizing Gaze-tracker

Traditionally, it has been argued that the gaze-tracker needs to be highly accurate to be used in TFR. We demonstrate that a properly co-designed foveated renderer is resilient to errors in the gaze-tracker. We optimize the gaze-tracker using two techniques: i) structured pruning (Section 2.5) and ii) image resizing.

We first evaluate multiple configurations with pruning levels varying between $\sim$38% to 99.5% of weights pruned. We show the performance, energy, and quality tradeoffs of these optimizations in Section 5.

Next, we approximate the gaze-tracker by using smaller input images. We resize the input image by applying different levels of downsampling, ranging from 40% to 80% on both the X and Y axes. Both these techniques– pruning and resizing – introduce errors in the gaze-tracker. We measure this error with the intermediate quality metric of degree tracking error. Based on the degree tracking error, we select the top-performing models with errors that can be mitigated by the foveated renderer. We explain the mitigation strategy in the next section (3.2).

3.2 End-to-end Quality Improvement

We render frames using gazes from both the baseline network as well as the optimized networks. The inputs to these networks are eye video sequences from the OpenEDS 2020 dataset [28]. Based on the JOD values of the rendered frames, we show that in most cases, this optimization has little to no effect on the end-to-end quality. However, some cases require slight mitigation, which can be done by marginally increasing the radius of the middle region. The middle region is rendered with 2×2 coarse pixel shading. Increasing the size of this region significantly improves the quality by reducing visual artifacts in the periphery. Otherwise, these artifacts would be visible in the outer region (4×4 coarse pixel shading). More importantly, increasing the radius of the middle region has a minimal performance hit. We can thus tolerate a less accurate gaze-tracker while maintaining similar end-to-end image quality, without incurring a significant performance penalty in the rendering sub-system.

4 EXPERIMENTAL METHODOLOGY

We first describe the metrics, input scenes, and experimental platforms that we use to compare FFR and TFR. We then discuss the methodology for finding the optimal FFR and TFR configurations and comparing them in terms of performance and quality. Lastly, we discuss the cost of the gaze-tracker needed to support TFR.

4.1 Metrics

The metrics that we select for comparison are *the frame time (1/FPS), end-to-end latency, energy-efficiency, and JOD*. Frame time should match the 120 FPS frame rate requirement of VR headsets, which corresponds to a frame time of roughly 8 ms. As described in Section 2.4, TFR's end-to-end latency must be less than 50 ms. We use FovVideoVDP to quantify the foveated render quality in terms of JOD. We define an image or video sample to be of acceptable quality if it has a JOD ≥ 9.

We also use an additional metric, degree tracking error, to tune the approximation levels of the gaze-tracker.

4.2 Input Scenes

We select 4 scenes covering a mix of indoor and outdoor surroundings: Bistro (Outdoor), Sponza (Indoor), Classroom (Indoor), and San-Miguel (Indoor and Outdoor). The scenes also represent a range of complexity; e.g., Sponza has $<$300,000 triangles while Bistro has $>$2 million triangles. Ideally, the scenes should also vary in their shader complexity and number of draw calls. We emulate the shader complexity by using the Repeat Pixel Shading (RPS) feature in VRS. RPS increases the number of lighting function calls, thus increasing the shader's workload. Even though it is unreasonable to run complex scenes and games on a tetherless headset with existing hardware and software, our work is a push toward enabling higher-complexity experiences on a mobile headset. While the evaluated scenes are diverse, we expect that our results could be refined by evaluating more scenes, including those that are more diverse in content and lighting conditions. We leave this to future work.

4.3 Experimentation Platform

Renderer: For rendering, we use NVIDIA's VRS (Variable Rate Shading). The foveated rendering parameters that we vary are the radius of the inner region and the radius of the middle region (see Section 2.1). We also vary MSAA to further tune the image quality. Lowering these three parameters results in more aggressive foveation and higher performance. However, it also results in a degradation of quality.

Gaze-tracker: We need to run the gaze-tracker in isolation to measure its power consumption. Unfortunately, no commercial headset provides this flexibility, as most only provide gaze data without any technical details. Hence, it is difficult for us to model the cost of a gaze-tracker.

We decided to use RITNet [4] for tracking the eye and DeepVOG's extrapolation methods to calculate the gaze vectors. RITNet is an eye segmentation network that won the OpenEDS 2019 challenge for eye segmentation conducted by Meta Reality Labs [25]. Setting up our own gaze-tracker gives us full control over both power measurement and tracking parameters, allowing us to co-design it with the foveated renderer.

For the end-to-end TFR quality measurements, we need realistic gaze distributions, so we use the OpenEDS 2020 dataset [28]. It consists of user sequences of real eye images captured at 100 Hz. We randomly pick 10 user sequences from the dataset.

Hardware Setup: In order to conduct performance, power, and quality experiments with foveated rendering, we need a hardware setup that supports gaze-tracking, an accessible software stack, and a renderer like NVIDIA VRS, which currently only runs on a desktop. Such control of the hardware and the runtime system is impractical in commercial tetherless headsets, so we choose a discrete NVIDIA GPU to emulate both desktop and mobile GPUs.

For desktop GPU rendering, we select a high-frequency mode of a desktop GPU-—RTX 3090. We set the computation frequency to 1590 MHz and the memory to 9501 MHz. To emulate the high shader complexity found in modern desktop VR games, we use an RPS value of 2000.

To emulate mobile rendering, we consider a mobile GPU that can support modern games (like Half-life: Alyx) at a frame rate of more than 120 FPS. Since such hardware doesn't presently exist in the market, it is difficult to get an appropriate estimate of performance and power consumption. Furthermore, NVIDIA's VRS renderer provides ease of analysis but is only supported on limited hardware platforms. Therefore, to get reasonable estimates, we decided to use a low-power mode on the same GPU (RTX 3090) by lowering the compute and memory frequencies to 450 MHz and 5001 MHz. The **idle power** consumption in this mode is 80 W. This is significantly more than modern mobile GPUs, as some of NVIDIA's Jetson modules can only consume a maximum power of 60 W. Even mobile hardware that supports VRS, such as the Quadro RTX 3000, has a maximum power rating of 80 W. Thus, we note that low-power mode on a desktop GPU may not provide the most accurate power estimate, but it is a reasonable first-order estimate in the absence of actual hardware. To emulate mobile headset games in VRS, we select an RPS value of 500. We select this number by referring to Oculus Quest's developer guide [24]. We use the same number of draw calls used in their performance experiments since the Quest is a representative mobile device.

4.4 Finding the Optimal Configurations for FFR and TFR

To find the optimal foveation configuration for FFR and TFR, we first take a uniform sample of gazes on the screen. For example, for a headset with a total resolution of 2880×1600, the per-eye resolution is 1440×1600. We fix the resolution to this value to match most modern headsets. The resolution in next-generation headsets is only expected to increase, in which case we expect higher benefits from foveation. We divide the screen into an 8×8 grid and take a gaze sample from each tile. We then use this gaze distribution as input to the foveated renderer while trying different foveation configurations.

We compare multiple foveation configurations using the gaze distribution described above. We generate these configurations by varying three parameters - MSAA levels, inner radius, and middle radius. VRS represents the radius of foveation levels as a percentage of the screen's smaller dimension. For example, for a single eye resolution of 1440×1600, an inner radius of 0.1 would result in a circular region with a 0.1×1440 = 144-pixel diameter.

We test three MSAA values - $1\times$ (No MSAA), $2\times$, and $4\times$. For each MSAA value, we vary the inner radius from 0.1 to 0.5 with a step size of 0.1. Similarly, we vary the middle radius from 0.2 to 0.9. In total, we generate 36 foveation configurations for comparison.

Table 1: We use 3 hardware configurations - one for Jetson and two for RTX 3090 (in Desktop mode and Low-power mode)

Hardware setup		Configuration	
Jetson Xavier AGX		CPU	2.2 GHz (single core)
		GPU compute	675 MHz
		GPU memory	1 GHz
RTX 3090	Desktop	CPU	4.8 GHz
		GPU compute	1.5 GHz
		GPU memory	9.5 GHz
	Low-power	CPU	2.2 GHz
		GPU compute	450 MHz
		GPU memory	5 GHz

For each configuration, we render frames in VRS using a centered gaze for FFR and the gaze distribution for TFR. We then measure the quality of each configuration. We compare the FFR frames against the non-foveated frames while notifying the quality metric of the real gaze location. We repeat the same experiment for TFR, where the renderer shifts the region of high fidelity according to the gaze.

We then select the cheapest configurations that produce acceptable quality, which is defined to be a JOD ≥ 9 in Section 2.3.

4.5 Comparison between FFR and TFR

Once we decide the optimal configurations for FFR and TFR, we compare their frame time (1/FPS) and energy per frame. When comparing the energy, we first examine rendering energy in isolation, and then we also add the cost of the gaze-tracker to TFR. We show the results from these experiments in Section 5.1.

4.6 Adding the Cost of the Gaze-tracker

We run the baseline implementation of the gaze-tracker on a mobile GPU (Jetson Xavier AGX) and RTX 3090 in both low-power and high-power modes to measure the cost of the gaze-tracker. We use tensor cores in all three cases (Jetson, RTX 3090 with desktop and low-power mode) by using NVIDIA's TensorRT platform.

In a tetherless headset, the gaze-tracker would likely run on a mobile device. We emulate its latency by using Jetson Xavier AGX and compare it against the required latency budget of 15 ms (see Section 2.4). Using the same setup, we test the improvements of our approximated gaze-trackers and select the approximation levels that fit the latency budget on the Jetson. However, the VRS renderer is only supported on limited platforms and not on the Jetson. Since we run VRS on RTX 3090 in low-power and desktop mode, we also run the gaze-tracker with the same configurations. We then add the gaze-tracker energy cost to the energy cost of TFR and compare the end-to-end TFR to FFR. We discuss this further in Section 5.2.

5 Results

We first present the quality, performance, and energy analysis for FFR and TFR in Section 5.1. Since TFR requires a gaze-tracker, we determine an energy budget for the gaze-tracker based on the difference in energy cost between TFR and FFR (Section 5.2). For TFR to be beneficial (energy-efficient), the gaze-tracking solution's energy cost must fit within this budget. We show that the baseline gaze-tracker exceeds the budget in most cases. Hence, we focus on optimizing the gaze-tracker by using approximations. As shown in Section 5.3, our optimization techniques result in performance and energy improvement for the gaze-tracker in isolation. Approximation results in an intermediate quality loss. Based on this quality loss and performance benefits, we select the final approximation level (pruning level and input sizes) of the gaze-tracker for an end-to-end analysis. Lastly, we show the end-to-end quality of our approximated gaze-trackers in Section 5.4.

Table 2: Selected configurations of FFR and TFR (For MSAA 4×, there is no middle region, so there is no middle radius)

	Params	**Sponza**	**Bistro**	**San-Miguel**	**Classroom**
FFR	MSAA	4×	4×	1×	1×
	Inner Radius	0.1	0.1	0.1	0.1
	Middle Radius	-	-	0.4	0.2
FFR	MSAA	1×	1×	1×	1×
	Inner Radius	0.1	0.1	0.1	0.1
	Middle Radius	0.7	0.6	0.3	0.2

5.1 Performance and Energy Difference Between FFR and TFR

We select the optimal configuration for both TFR and FFR to compare their respective frame times and energy per frame based on the experiments described in Section 4.4.

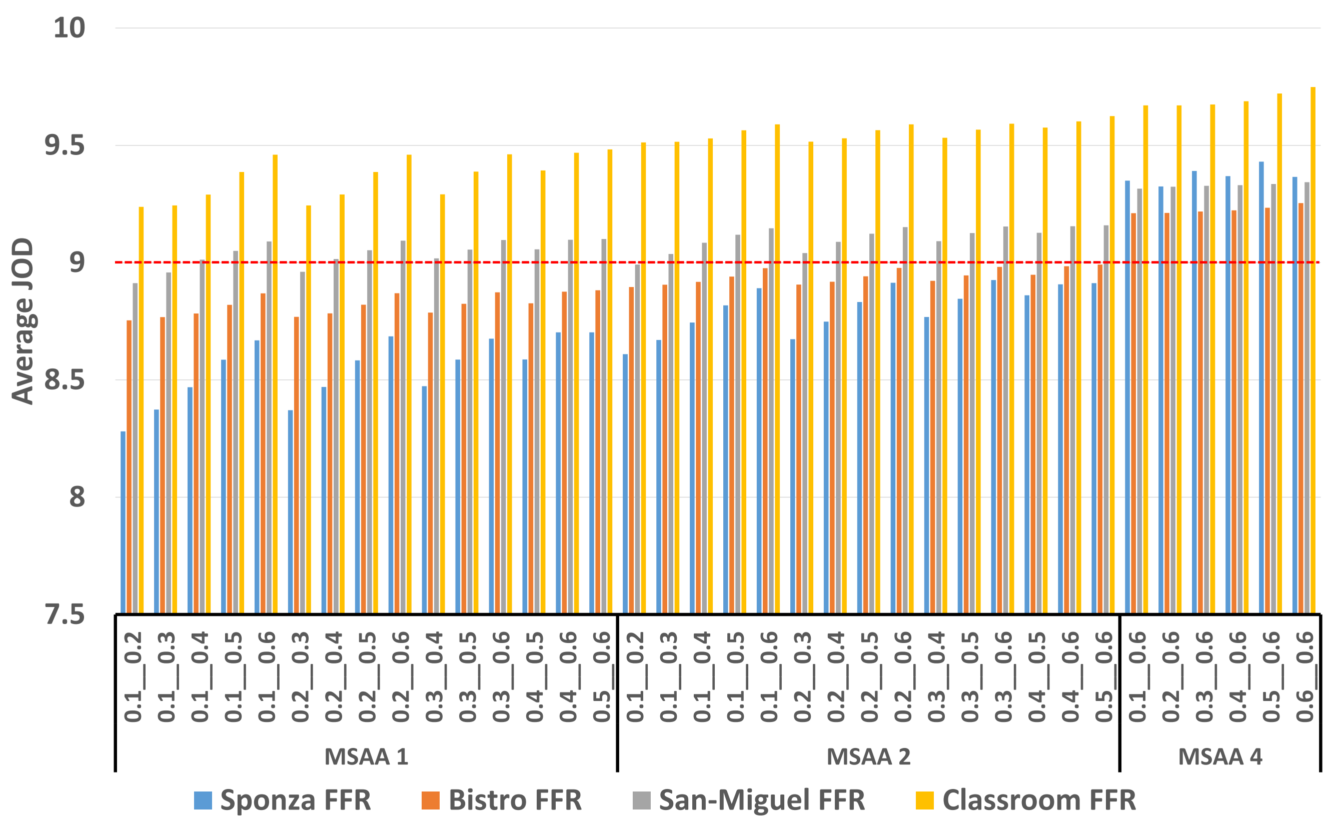
- **Selecting optimal foveation configurations** — We run the experiments for MSAA values of 1× (No MSAA), 2×, and 4×. The range of the inner fovea radius increments from 0.1 to 0.5, and the range of the middle fovea radius increments from 0.2 to 0.9.

  Fig 2 shows Average JOD vs different configurations, with FFR on the top and TFR on the bottom. Horizontal lines show the JOD value of 9, which is the acceptable frame quality. Based on this line, we observe that FFR requires a higher level of MSAA to achieve acceptable quality. For Bistro and Sponza, FFR needs an MSAA of 4× and a small inner radius. The exception is San-Miguel where an acceptable quality is achieved at an MSAA level of 1× (No MSAA) and a much smaller inner radius. For TFR to reach an acceptable frame quality level, an MSAA of 1× (No MSAA) is sufficient with a small inner radius. This is because the high-quality region always follows the gaze, so foveation artifacts only appear in the periphery.

  Using the results from Fig 2, we select configurations for FFR and TFR as depicted in Table 2. Based on these plots, we also notice that for the two scenes - Classroom and San-Miguel, the metric shows an acceptable JOD value (≥ 9) with highly aggressive foveation schemes for both FFR and TFR. Specifically, for FFR, the JOD becomes acceptable at an MSAA value of 1× and 0.1 inner radius. This is unusual as we expect FFR to demand conservative (less aggressive) foveation than TFR to achieve the same (acceptable) quality.

  We verify this visually on both an HMD and a monitor. We found that for these two scenes (Classroom and San-Miguel), FovVideoVDP appeared to overestimate the quality scores compared to what we observed manually. We illustrate this in Fig. 5. For all three scenes, the left part is rendered with TFR with the gaze dot (red dot) and the gaze area highlighted with a rectangle. On the right, we show how this looks with FFR and TFR. The difference between FFR and TFR is clear as FFR appears blurred, and this experience is further pronounced in the HMD. However, the JOD values of FFR and TFR are both above 9 for two out of these three scenes (San-Miguel and Classroom), which suggests a mismatch between the visual observation and the existing state-of-the-art metric.

- **Comparing FFR and TFR** — Fig 3 shows the frame time speedup given by TFR compared to non-foveated rendering and FFR based on selected configurations for different scenes. As discussed in the previous point, the foveation configurations for TFR and FFR are different. Thus, TFR should ideally perform better than FFR. We show the comparison for two power modes - desktop mode (high frequency and high power) and



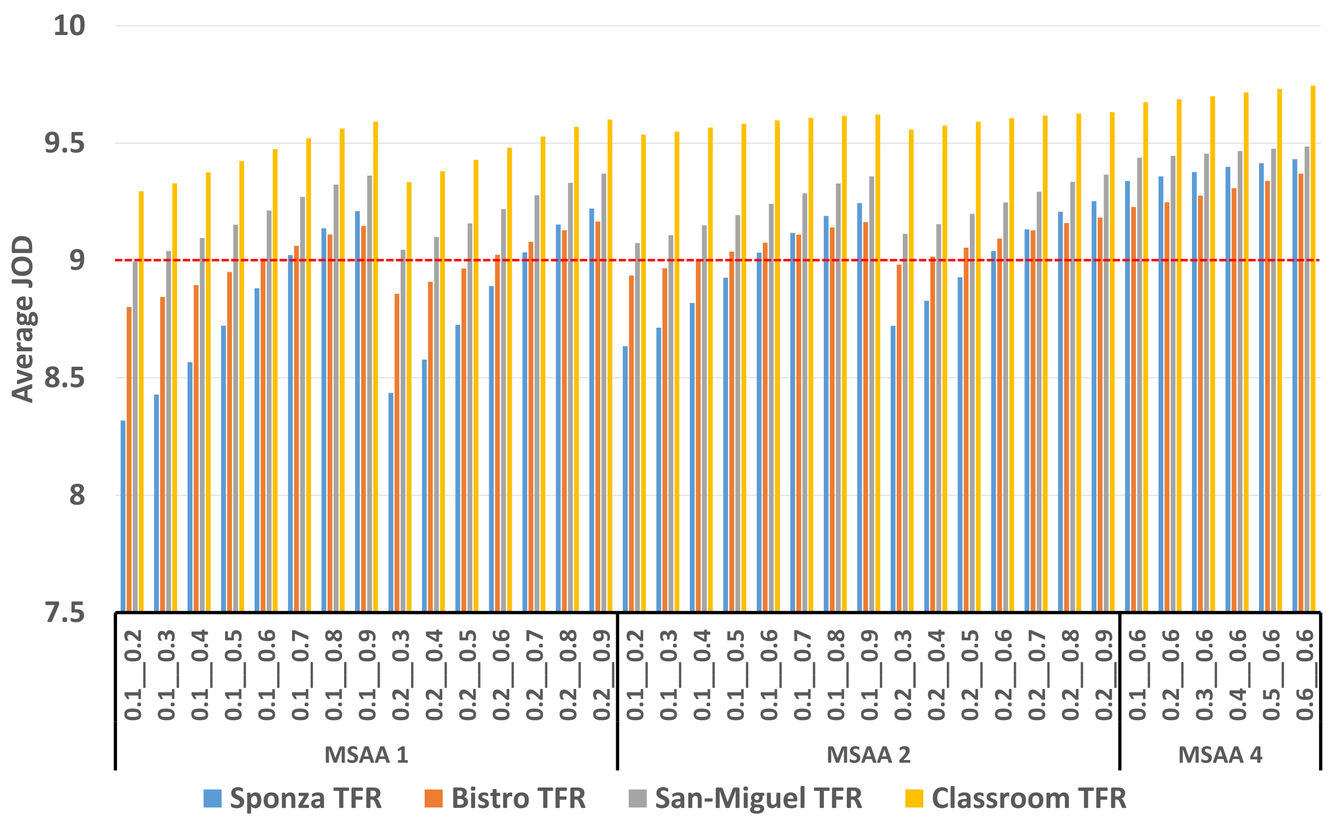


Figure 2: The top is FFR JOD values, while the bottom is TFR JOD values. The horizontal dotted line represents the acceptable frame quality. On the X axis, we have different MSAA levels and different fovea radii in the format - *inner-radius__middle-radius*.*inner-radius* represents the radius of the fovea rendered at the highest resolution, and the *middle-radius* represents the second foveation radius rendered at a lower resolution. For FFR, an acceptable frame quality is achieved towards the right at a higher MSAA value and fovea radius. For TFR, JOD reaches an acceptable level at a much lower MSAA level with a smaller fovea radius, resulting in lower power consumption and faster frame times.

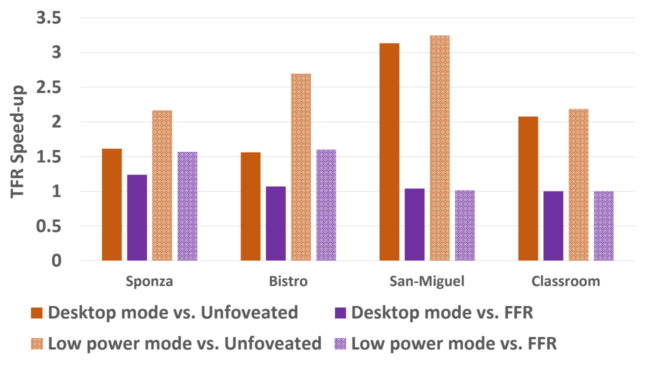


Figure 3: Speedup of TFR compared to non-foveated rendering and FFR for different power modes. There is a significant speedup compared to non-foveated rendering for all the scenes. However, compared to FFR, the speedup is limited and varies based on the scene. In the case of San-Miguel and Classroom, there is marginal or no speedup compared to FFR.

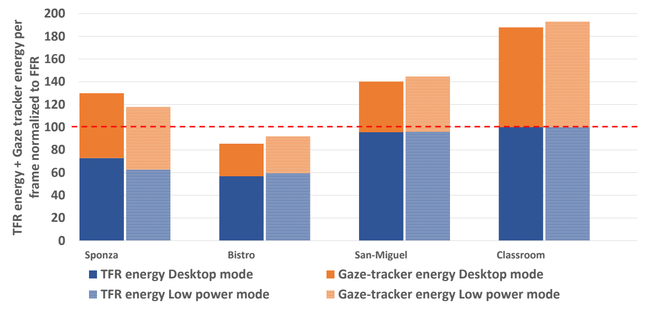


Figure 4: Total energy per frame (normalized to FFR) = energy per frame of TFR + energy per frame of the gaze-tracker normalized to the energy per frame of FFR

low-power mode (low frequency) as set in Table 1. We notice that all the scenes show speedup compared to non-foveated rendering, but only 2 out of 4 scenes show speedup compared to FFR. This is because we selected the FFR and TFR configurations based on the quality metrics, and for two scenes (San-Miguel and Classroom) the metric overestimates the quality of FFR as shown in Fig 5. In reality, TFR would still perform better than FFR for these particular scenes. This also motivates further improvement in quality metrics for foveated rendering. From the above experiments, we establish that the frame time speedup by TFR can vary based on the scene in question.

Takeaway - Based on the optimal FFR and TFR configurations, we find that TFR can provide 1.5× to >3× speedup compared to non-foveated rendering. Benefits compared to FFR are limited to only two scenes and are $\leq$1.5×. We attribute this reason to the mismatch between the quality reported by the metric and visual observation. This motivates further research in quality metrics for foveated rendering.

5.2 Energy Efficiency of FFR vs TFR

Fig. 4 shows the percentage of energy savings by TFR (normalized to FFR) in desktop mode and low-power mode for different scenes. Based on this plot, we show that TFR (in isolation) provides reasonable energy savings compared to FFR in 2 out of 4 scenes with no savings in Classroom and negligible savings in San-Miguel. However, when we add the cost of the gaze-tracker, the energy cost of the total system exceeds the total cost of FFR. This shows that the baseline gaze-tracker cannot be directly used for TFR.

Takeaway - TFR consumes more energy than FFR when using the baseline gaze-tracker. We must optimize the gaze-tracker to make TFR feasible.

5.3 Optimizing the Gaze-tracker

As mentioned in Section 3.1, we optimize the gaze-tracker by using approximation techniques - structured pruning and resizing. We show the performance (latency) improvement on a Jetson Xavier AGX as the gaze-tracker would likely run on a mobile device (see Section 4.6).

- Fig. 6 shows how the gaze-tracker latency varies with different levels of pruning and resizing on the Jetson. Pruning levels are varied from 38% to 99.5%, while the input image has 3 sizes: 1. *default size* 640x400 pixels 2. *Resize 1* 256x160 pixels 3. *Resize 2* 128x80 pixels. As discussed in Section 2.4, the required latency for the gaze-tracker is <15 ms. This plot shows that the gaze-tracker is too slow (roughly 40 ms) when using the default input size, even at the highest pruning level.

- For Resize 1 (256x160) and Resize 2 (128x80), we compare the intermediate quality of different pruning levels in Fig. 7. We represent the intermediate quality using degree tracking error. We also show the latency of these approximation levels. Based on this plot, we notice that the latency requirement of 15 ms is met by the higher pruning levels of Resize 1 and by most pruning levels (including the baseline) of Resize 2. However, we notice that the intermediate quality (the degree tracking error) of Resize 2 is worse compared to Resize 1. For Resize 2, only 50% of the gazes from the unpruned version are below the degree error of 3. In comparison, even the most pruned version of Resize 1 has better intermediate quality compared to the unpruned version of Resize 2. To compensate for this error later in the pipeline, we must increase the fovea size (Section 3), i.e. reduce the level of foveation. Since the error of Resize 2 is high, we will need to increase the fovea size significantly. This would negate the benefits of TFR compared to FFR, so even though Resize 2 has better performance and energy efficiency compared to Resize 1 in isolation, we choose to use Resize 1 for the co-design.

Takeaway - Based on the above experiments, we realize that over-resizing results in a loss of essential features. We instead choose to apply higher pruning levels with limited resizing. Thus, we select Resize 1 (256x160) and a pruning level of 98.29, 99, and 99.5.

5.4 End-to-end Quality and Performance Tradeoff of the Approximated Gaze-tracker

Based on the TFR configurations that we finalized in Section 5.1 for all 4 scenes, we now show the end-to-end quality using our approximated gaze-trackers. We use eye video sequences from OpenEDS 2020 dataset as input to the gaze-trackers. Fig. 8 shows the average JOD value for all scenes with TFR using our approximated gaze-tracking models. Here, we show 3 of our highest pruned models that generate gazes for 10 different gaze sequences/paths, and we notice that the quality is above the acceptable level (JOD $\geq$ 9) for the most part. For one sequence in Bistro, the quality marginally degrades from 9 to 8.95. To recover this, we try the next configuration for Bistro - MSAA = 1x, inner radius = 0.1, middle radius = 0.9. This configuration makes sure that the JOD is always $\geq$ 9 while incurring minimum performance loss, i.e. 1.09x and 1.02x slower compared to the previous configuration in the low-power and desktop modes respectively. For desktop mode, the energy cost increases from 56%

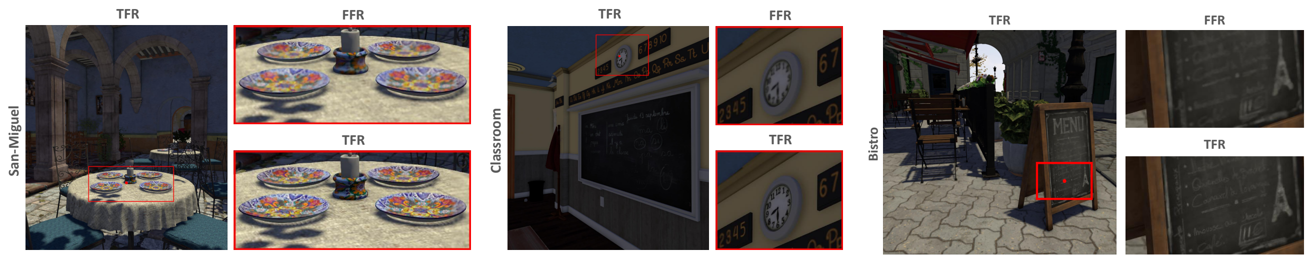


Figure 5: An illustration of the perceived visual difference between FFR and TFR for San-Miguel, Classroom, and Bistro scenes. To users observing these scenes in VR, the plates (San Miguel; left), clock face (Classroom; center), and the menu text (Bistro; right) look blurred when using FFR and degraded quality is easily noticeable. FovVideoVDP works well for Bistro as the JOD number for FFR is 8.5, which is considered unacceptable according to our metric. However, the JOD values reported by FovVideoVDP for both FFR and TFR are above 9 in the case of Classroom and San-Miguel.

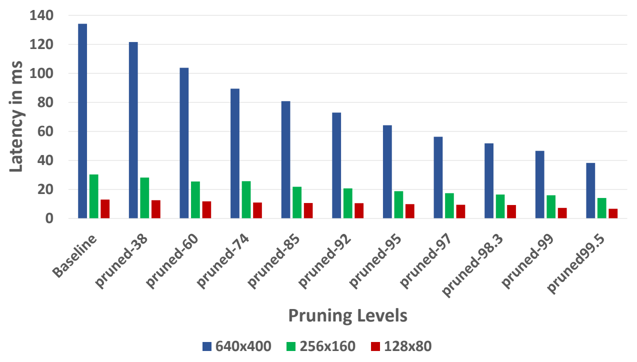


Figure 6: Latency of the gaze-tracker for different levels of optimization. The plot shows the latency improvement on a mobile device (Jetson Xavier AGX) by ten levels of pruning and two levels of resizing

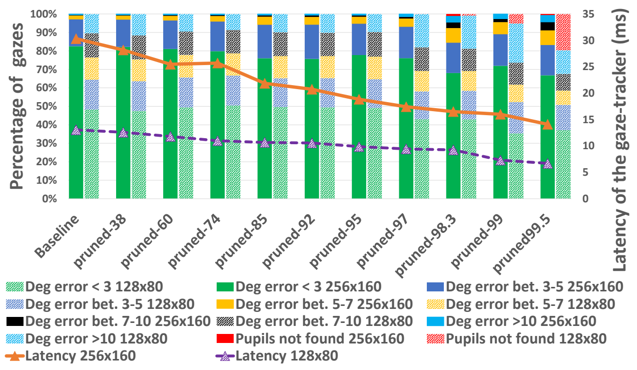


Figure 7: Stacked bars - the intermediate quality of the gaze-tracker (%age of gazes and their degree tracking error) for different approximated versions of the gaze-tracker. The secondary axis represents the latency in ms. The acceptable latency for gaze-tracking is 15 ms.

to 58% of the FFR energy/frame, and for the low-power mode, it increases from 59% to 64% of the FFR energy/frame.

Finally, Fig. 9 demonstrates that TFR with the added cost of the gaze-tracker still saves significant energy for Bistro and Sponza. This is possible because of the optimizations that we carried out in the gaze-tracker. For San-Miguel and Classroom, we don't see any benefit, as TFR in isolation is almost as expensive as FFR. As explained before, this is because the quality metric is not able to differentiate between FFR and TFR even though there is a noticeable difference in visual quality.

Takeaway - We use three approximated gaze-trackers selected from Section 5.3 and measure the end-to-end quality (JOD) of all the scenes. We note that for the most part, the quality is acceptable, except in one scene for certain gaze sequences. We mitigate this by choosing the next higher-quality foveation configuration. This achieves minimal performance loss for TFR while maintaining acceptable quality. Overall, TFR added with our gaze-tracker remains cheaper and faster than FFR in most cases.

6 Discussion and Conclusion

Our study has shown that foveated rendering continues to be an effective optimization for XR systems, even those constrained by aggressive computational and power budgets. When rendering scenes with high complexity and heavy shading workload, the benefits of foveated rendering are much more pronounced. However, we demonstrate that the benefits of TFR vs FFR are not always clear. Even though TFR renders at a better quality than FFR and is more energy efficient in most cases, the additional energy cost of gaze-tracking can sometimes outweigh TFR's advantages. To our knowledge, this paper provides a first detailed study of power, performance, and quality comparison between TFR and FFR, and shows the need to improve the gaze-tracker not just in accuracy, but also in energy and latency. We show that using our approximation techniques, we can reduce the gaze tracking energy by 20× with a speedup of 9×. Further, by co-designing the gaze-tracker with the foveated renderer, we can lower the total energy cost of TFR by up to 40% compared to FFR while maintaining an acceptable end-to-end image quality.

Our study also highlights the shortcomings in existing quality metrics for peripheral vision. While designing our foveated rendering techniques, we found that JOD values predicted by our state-of-the-art metric didn't always match our observations when viewed with a VR headset. In order to enable rapid and effective evaluation of design choices, the community would benefit from quality metrics with closer correlations to human perception.

7 Limitations and Future Work

While this work opens up a discussion about the benefits of different foveated rendering modes, there are some limitations and new directions that we would like to explore in our future work.

We have considered 4 scenes in this work for their diversity in indoor and outdoor surroundings, as well as their number of polygons. For our future work, we would like to further diversify our scenes, especially based on scene content and lighting conditions.

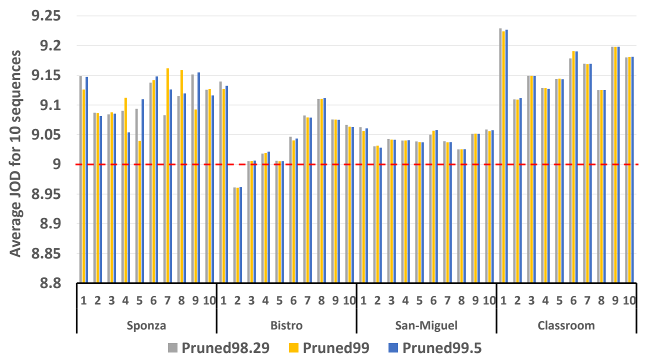


Figure 8: Average JOD for 10 user sequences for different scenes and multiple pruning levels

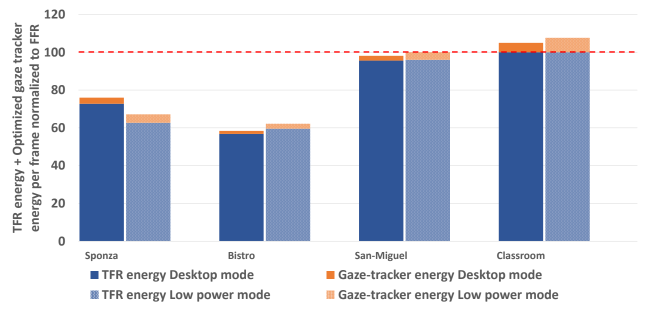


Figure 9: Energy per frame with optimized eye tracker, normalized to the energy/frame of FFR

We don't consider any hardware optimizations when optimizing our gaze-tracker, but we would like to highlight the need to do so—with existing hardware technologies, it is infeasible to have a complete and comfortable tetherless experience in a VR headset.

Another direction that we wish to pursue is to move the compute closer to the sensor. This can be helpful in the case of the gaze-tracker, as we can save the energy and latency of transferring the eye images from the camera to the compute unit. We also plan to explore offloading rendering to the cloud while utilizing foveation to reduce the bandwidth and latency requirements of the cloud rendering pipeline.

Finally, our study was limited by (un)availability of hardware with measurable and flexible support for foveated rendering and gaze-tracking, especially in the range of low-power systems. In the future, we plan to extend emerging end-to-end open-source XR research testbeds such as ILLIXR [13] to enable even more realistic and comprehensive evaluations.

Acknowledgments

This work was supported in part by the Applications Driving Architectures (ADA) Center, a JUMP Center co-sponsored by SRC and DARPA, the DARPA DSSOC program, the National Science Foundation under grants 1956374, 2120464, and 2217144, and by a gift from Cisco Systems Inc.

References

[1] K. Akşit, P. Chakravarthula, K. Rathinavel, Y. Jeong, R. Albert, H. Fuchs, and D. Luebke. Manufacturing application-driven foveated near-eye displays. IEEE Transactions on Visualization and Computer Graphics, 25(5):1928–1939, 2019. doi: 10.1109/TVCG.2019.2898781

[2] R. Albert, A. Patney, D. Luebke, and J. Kim. Latency requirements for foveated rendering in virtual reality. ACM Trans. Appl. Percept., 14(4), sep 2017. doi: 10.1145/3127589

[3] S. Anwar, K. Hwang, and W. Sung. Structured pruning of deep convolutional neural networks. ACM Journal on Emerging Technologies in Computing Systems (JETC), 13(3), 2017.

[4] A. K. Chaudhary, R. S. Kothari, M. Acharya, S. Dangi, N. Nair, R. Bailey, C. Kanan, G. J. Diaz, and J. B. Pelz. Ritnet: Real-time semantic segmentation of the eye for gaze tracking. CoRR, abs/1910.00694, 2019.

[5] S. Chen, B. Duinkharjav, X. Sun, L. Wei, S. Petrangeli, J. Echevarria, C. T. Silva, and Q. Sun. Instant reality: Gaze-contingent perceptual optimization for 3d virtual reality streaming. CoRR, abs/2201.03484, 2022.

[6] A. Developer. Cortex-a72, 2021.

[7] Y. Feng, N. Goulding-Hotta, A. Khan, H. Reyserhove, and Y. Zhu. Real-time gaze tracking with event-driven eye segmentation. CoRR, abs/2201.07367, 2022.

[8] B. Guenter, M. Finch, S. Drucker, D. Tan, and J. Snyder. Foveated 3d graphics. ACM Transactions on Graphics (TOG), 31, 11 2012. doi: 10.1145/2366145.2366183

[9] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In Y. Bengio and Y. LeCun, eds., 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings, 2016.

[10] L. Hsiao, B. Krajancich, P. Levis, G. Wetzstein, and K. Winstein. Towards retina-quality vr video streaming: 15ms could save you 80% of your bandwidth. SIGCOMM Comput. Commun. Rev., 52(1):10–19, mar 2022. doi: 10.1145/3523230.3523233

[11] HTC. HTC Vive Pro. Available at `https://www.vive.com/us/product/vive-pro-eye/overview/`.

[12] W. Hunt, M. Mara, and A. Nankervis. Hierarchical visibility for virtual reality. Proceedings of the ACM on Computer Graphics and Interactive Techniques, 1:1–18, 07 2018. doi: 10.1145/3203191

[13] M. Huzaifa, R. Desai, S. Grayson, X. Jiang, Y. Jing, J. Lee, F. Lu, Y. Pang, J. Ravichandran, F. Sinclair, B. Tian, H. Yuan, J. Zhang, and S. V. Adve. Illixr: An open testbed to enable extended reality systems research. IEEE Micro, 42(4):97–106, 2022. doi: 10.1109/MM.2022.3161018

[14] J. Kim, Y. Jeong, M. Stengel, K. Akşit, R. Albert, B. Boudaoud, T. Greer, J. Kim, W. Lopes, Z. Majercik, P. Shirley, J. Spjut, M. McGuire, and D. Luebke. Foveated ar: Dynamically-foveated augmented reality display. ACM Trans. Graph., 38(4), jul 2019. doi: 10.1145/3306346.3322987

[15] Y. LeCun, J. S. Denker, and S. A. Solla. Optimal brain damage. In Advances in neural information processing systems, 1990.

[16] S. Lee, M. Pattichis, and A. Bovik. Foveated video quality assessment. IEEE Transactions on Multimedia, 4(1):129–132, 2002. doi: 10.1109/6046.985561

[17] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning filters for efficient convnets. International Conference on Learning Representations (ICLR), 2017.

[18] S. Lin, R. Ji, C. Yan, B. Zhang, L. Cao, Q. Ye, F. Huang, and D. S. Doermann. Towards optimal structured CNN pruning via generative adversarial learning. CoRR, abs/1903.09291, 2019.

[19] E. Malkin, A. Deza, and T. Poggio. Cuda-optimized real-time rendering of a foveated visual system, 2020.

[20] M. Mansouryar, J. Steil, Y. Sugano, and A. Bulling. 3d gaze estimation from 2d pupil positions on monocular head-mounted eye trackers. Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications, Mar 2016. doi: 10.1145/2857491.2857530

[21] R. Mantiuk, K. J. Kim, A. G. Rempel, and W. Heidrich. Hdr-vdp-2: A calibrated visual metric for visibility and quality predictions in all luminance conditions. ACM Trans. Graph., 30(4), jul 2011. doi: 10.1145/2010324.1964935

[22] R. K. Mantiuk, G. Denes, A. Chapiro, A. Kaplanyan, G. Rufo, R. Bachy, T. Lian, and A. Patney. FovVideoVDP: A visible difference predictor for wide field-of-view video. ACM Transactions on Graphics, 40(4), jul 2021.

[23] Meta. Meta Quest Pro. Available at https://www.oculus.com/blog/meta-quest-pro-price-release-date/.

[24] Meta. Oculus performance and optimization guide.

[25] Meta. Openeds challenge 2019.

[26] B. Mohanto, A. T. Islam, E. Gobbetti, and O. Staadt. An integrative view of foveated rendering. Comput. Graph., 102(C):474–501, feb 2022. doi: 10.1016/j.cag.2021.10.010

[27] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz. Pruning convolutional neural networks for resource efficient inference. In 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April, 2017, Conference Track Proceedings, 2017.

[28] C. Palmero, A. Sharma, K. Behrendt, K. Krishnakumar, O. V. Komogortsev, and S. S. Talathi. Openeds2020: Open eyes dataset. CoRR, abs/2005.03876, 2020.

[29] 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), nov 2016. doi: 10.1145/2980179.2980246

[30] A. Renda, J. Frankle, and M. Carbin. Comparing rewinding and fine-tuning in neural network pruning. In International Conference on Learning Representations, 2020.

[31] S. Rimac-Drlje, G. Martinović, and B. Zovko-Cihlar. Foveation-based content adaptive structural similarity index. In 2011 18th International Conference on Systems, Signals and Image Processing, pp. 1–4, 2011.

[32] Sony. PSVR 2. Available at https://www.playstation.com/en-us/ps-vr2/.

[33] N. Stein, D. C. Niehorster, T. Watson, F. Steinicke, K. Rifai, S. Wahl, and M. Lappe. A comparison of eye tracking latencies among several commercial head-mounted displays. i-Perception, 12(1):2041669520983338, 2021. PMID: 33628410. doi: 10.1177/2041669520983338

[34] N. T. Swafford, J. A. Iglesias-Guitian, C. Koniaris, B. Moon, D. Cosker, and K. Mitchell. User, metric, and computational evaluation of foveated rendering methods. SAP '16, p. 7–14. Association for Computing Machinery, New York, NY, USA, 2016. doi: 10.1145/2931002.2931011

[35] K. Vaidyanathan, M. Salvi, R. Toth, T. Foley, T. Akenine-Möller, J. Nilsson, J. Munkberg, J. Hasselgren, M. Sugihara, P. Clarberg, T. Janczak, and A. Lefohn. Coarse Pixel Shading. In I. Wald and J. Ragan-Kelley, eds., Eurographics/ ACM SIGGRAPH Symposium on High Performance Graphics. The Eurographics Association, 2014. doi: 10.2312/hpg.20141089

[36] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli. Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing, 13(4):600–612, 2004. doi: 10.1109/TIP.2003.819861

[37] Z. Wang, A. C. Bovik, L. Lu, and J. L. Kouloheris. Foveated wavelet image quality index. In Applications of digital image processing XXIV, vol. 4472, pp. 42–52. SPIE, 2001.

[38] Y.-H. Yiu, M. Aboulatta, T. Raiser, L. Ophey, V. L. Flanagin, P. Zu Eulenburg, and S.-A. Ahmadi. Deepvog: Open-source pupil segmentation and gaze estimation in neuroscience using deep learning. Journal of neuroscience methods, 324:108307, August 2019. doi: 10.1016/j.jneumeth.2019.05.016

[39] H. You, C. Wan, Y. Zhao, Z. Yu, Y. Fu, J. Yuan, S. Wu, S. Zhang, Y. Zhang, C. Li, V. Boominathan, A. Veeraraghavan, Z. Li, and Y. Lin. EyeCoD. In Proceedings of the 49th Annual International Symposium on Computer Architecture. ACM, jun 2022. doi: 10.1145/3470496.3527443

[40] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling. Mpiigaze: Real-world dataset and deep appearance-based gaze estimation. CoRR, abs/1711.09017, 2017.

[41] M. Zhu and S. Gupta. To prune, or not to prune: Exploring the efficacy of pruning for model compression. In 6th International Conference on Learning Representations, ICLR 2018, Workshop Track, 2018.