What are the main hurdles to the adoption of foveated rendering in HMDs?

Joe Down

April 11, 2024

1 Background

Humans direct their gaze towards an object by rotating their eyes such that light reflected by the object viewed falls on the eye's fovea[11]. This is a small region at the centre of the retina with increased cone density[21]: the photoreceptor cells of the eye which send signals to brain when stimulated by light[3]. Additionally, the brain is structured to process this information with a space-variant resolution dedicating most resources to the fovea region[37]. These conditions mean that spatial acuity is strongest for objects viewed within this area[11].

In virtual reality systems, it has been found that delay in frame delivery results in motion sickness, discomfort[35], and decreased performance in certain motor tasks[22]. Since rendering requirements continue to increase as screen reolutions and scene complexities increase, it remains difficult for modern computing devices to render complex VR scenes with low visual latency[36]. Foveated rendering has frequently been proposed as a technique to reduce frame render times and help mitigate this problem. It aims to, in one of a few ways, reduce render quality for parts of a VR scene in the periphery and/or increase render quality within the foveal view[38], exploiting the eye's space-variant resolution. Mohanto et al. [16] specify a range of subcategories for render quality reductions, all falling within 4 main subcategories: adaptive resolution (i.e. render the periphery with lower resolution or some other form of reduced sampling), geometric simplification (i.e. render objects with a less detailed model in the periphery), shading simplification/chromatic degeneration (i.e. simplify lighting simulation for pixels in the periphery), and spatio-temporal deterioration (i.e. partially reusing parts of the frame in the periphery to avoid having to re-render the entire view at each refresh). A simplified breakdown of these categories is shown in fig. 1.

To understand existing limitations, a high-level structure of a foveated rendering pipeline needs to be defined. There are two main types of foveated rendering pipeline with slightly different structures: static, and dynamic[16].

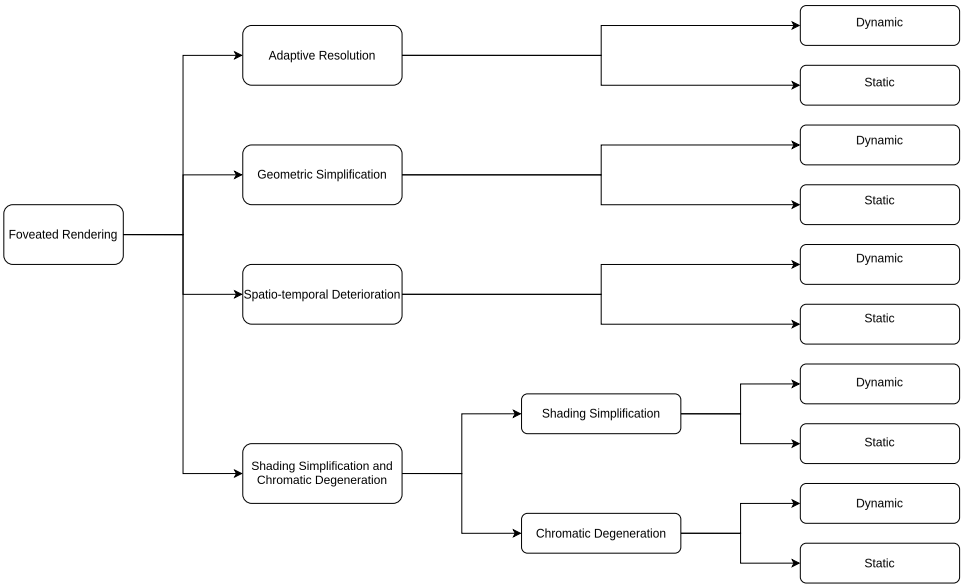

Figure 1: Mohanto et al.'s[16] foveated rendering subcategories.

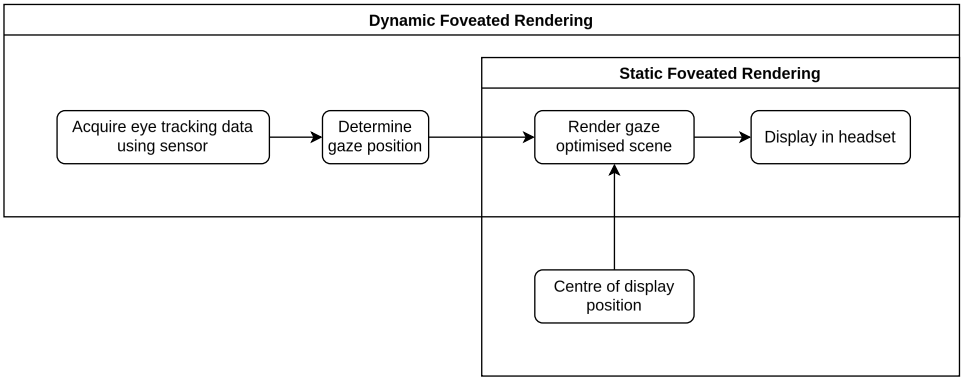

Figure 2: High level foveated rendering pipeline.

Dynamic foveated rendering requires some form of eye tracking capability within the HMD to determine which part of the stereoscopic displays the corresponding eye is directed towards. In software, frames must be received from this eye tracking device, and processed using some algorithm to determine where the eye is directed within a scene and/or on the physical display. The program being used must then be able to take this information and use it to in some way change how the scene is rendered. The rendered image must then be transmitted back to the headset and displayed, all within a short enough timeframe to avoid noticeable visual latency. Static foveated rendering is mostly the same regarding the software implementation, however rather than requiring eye tracking hardware and processing, it simply assumes the fovea region is in the centre of each display and performs the rendering optimisations accordingly. This is not a realistic model, but will lead to performance gains at the cost of likely noticeable quality loss when the user averts their gaze from the centre. A summary of this pipeline can be viewed in fig. 2.

2 Proposal

Static foveated rendering has been used by certain applications with non specialised hardware for years now. Dynamic foveated rendering has only started to become more common within the past few years, as increasingly mainstream HMDs such as PSVR2[19] and Apple Vision Pro[8] have been released with eye tracking functionality. The following sections aim to address why dynamic foveated rendering adoption is not yet ubiquitous across all HMDs, and identify shortcomings with existing implementations of both types of foveated rendering. First, I will address the hardware and software latency requirements, followed by other software requirements, hardware requirements, and usability requirements. I believe that the main obstacles relate to the visual latency resulting from acquiring eye tracking data from a sensor and performing gaze processing, the current lack of clear implementation standards for gaze tracking, the lack of reference dynamic foveated rendering implementations in common engines, the high costs for HMDs with eye tracking hardware, and the lack of mainstream adoption of HMDs with appropriate hardware to justify a userbase.

3 Time Limitations

The most significant limitations relate to the time taken to perform all of the foveated rendering process and optimisations. If this cannot be done in a shorter amount of time than just rendering an unchanged frame, then the entire process is redundant as visual latency has been worsened and could result in discomfort for the user. Additionally, with dynamic foveated

rendering, if the foveated region cannot be updated with low enough latency, the issue shown in fig. 3 will occur where the user noticeably looks into the lower quality region of the scene after large, rapid eye movements (saccades). Albert et al. [1] suggest gaze tracking latency for foveation of any level is unnoticeable with 20-40ms latency, becomes difficult to notice with less than 50-70ms of latency, while less aggressive foveation could be tolerated with latencies up to 150ms. Li et al. [12] similarly suggest an upper bound of 50ms latecy, while no paper on the topic seems to aim for a latency below 14ms[10]. In this section, issues will be broken down in the order of the dynamic foveated rendering pipeline as defined in fig. 2. Difficulties faced will be similar for static foveated rendering, ignoring parts relating to eye tracking.

3.1 Eye Tracking Sensor and Gaze Processing

The first time-based consideration is the time taken to acquire sensor data for eye tracking and then process this data to determine gaze position. Stein et al. [25] performed experiments to determine eye tracking delays for three headsets with eye tracking capabilities: FOVE-0, Varjo VR-1, and the HTC Vive Pro Eye. The delay is the time between an eye movement and the gaze data becoming available for usage. They found that the delay before the eye movement data became available ranged from between 15-52ms. Given the previously described latency requirements for unnoticeable gaze tracking, this leaves little time to process, render, and display within the 20-40ms available[1]. It does leave some room however for less aggressive foveation possible with 50-70ms latency[1]. This suggests that perhaps foveated rendering has not yet become ubiquitous due to current hardware latency difficulties in existing, expensive, headsets.

Stein et al. [25] also test the EyeLink 1000 eye tracker, an external webcam-like device capable of sampling at 1000HZ. From the same experimental setup this had a measurable delay of 0ms, suggesting significant room for improvement if this technology can be miniaturised. I was unable to find gaze delay data for newer headsets so better results may already be present, but these headsets remain expensive.

3.2 Scene Rendering Optimisation

The next time-based consideration is regarding the gaze optimised rendering of the scene. Various experiments indicate successful results in accelerated rendering. Meng et al. [14] demonstrate at least 2x framerates when using their kernel foveated rendering pipeline for rendering with decreased resolution in the peripheral (adaptive resolution in fig. 1). Guenter et al. [7] demonstrate 5-6x performance increases by collecting fewer samples for antialiasing in the peripheral view (shading simplification in fig. 1). Weier

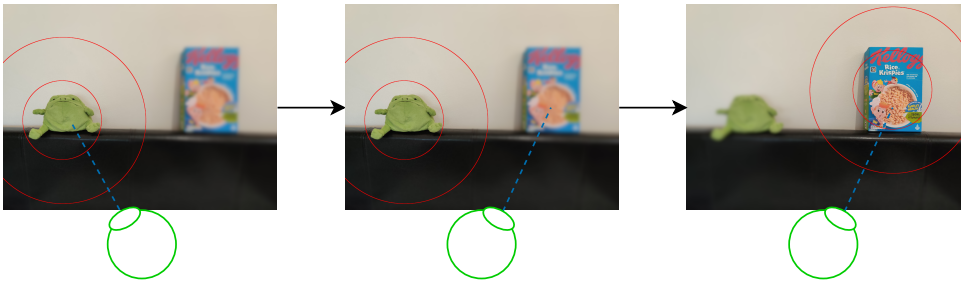

Figure 3: Large eye movements (saccades) result in the eye looking into the lower quality region before the frame updates. If eye tracking or image display is of high latency, this will be noticeable. Based on a similar diagram by Albert et al. [1].

et al. [38] demonstrate at least 1.46x performance increases using a reprojection technique which recycles pixels in the peripheral (spatio-temporal deterioration in fig. 1). I identified a few papers presenting geometric simplification techniques which claim improved performance, though without clearly presented figures, such as Tiwary, Ramanathan, and Kosinka [29] and Zheng et al. [39]

Overall it seems that many scene optimisation implementations have already been proposed and tested, proving significant performance gains can already be achieved with foveated rendering. Finding compelling scene rendering optimisations does not currently seem to be an obstacle to foveated rendering adoption when considering the overall pipeline.

3.3 Displaying in Headset

The final time-based consideration is regarding displaying the image in the headset. Since this must be performed even when not performing foveated rendering it presumably is not a source of problems. I was unable to find any literature referencing foveated rendering specific difficulties with this step.

3.4 Summary

Generally it seems that the biggest obstacle for time optimisations is regarding the latency of acquiring and processing eye tracking data, which is currently difficult to achieve within comfort requirements. Scene rendering optimisations already provide compelling results which do not appear to impede progress since render times can already be greatly reduced.

4 Other Software Limitations

Besides render time restrictions, there are some other software limitations to consider, largely regarding implementation difficulty.

4.1 Implementation Standards

One issue is with conflicting implementation standards for gaze tracking. OpenXR provides extensions for gaze interaction extensions [27], which is implemented by the Unity engine in the XR Interaction Toolkit [4] and Unreal engine through the OpenXREyeTracker plugin[18]. This would seem to imply some sort of standardisation, however I was unable to find clear documentation for how these plugins can be used for foveated rendering. Unity provides the FoveatedRenderingMode render flag[6], however this is not well documented and I couldn't determine whether this is dynamic or static foveated rendering. Vendor specific dynamic foveated rendering Unity implementations seem to exist for Meta Quest Pro[5] and Vive Pro Eye[33], but these will work only for those specific headsets and appear to implement only resolution based optimisations (see fig. 1 for a breakdown of other techniques). Additionally, not all headsets support the OpenXR standard at all such as Apple Vision Pro and PSVR[17]. All of these obstacles would seem to make implementation of foveated rendering for a cross-platform application a significant undertaking.

4.2 Gaze Tracking Implementation

Gaze tracking has been successfully implemented in a VR headset by, among others, Li, Liu, and Zhou [13] and Thies et al. [28], proving implementation is possible. As discussed in section 4.1, eye tracking headsets with OpenXR support provide an interface to gaze data (set up by the HMD manufacturer), which is implemented in both Unity and Unreal engines, meaning gaze data should be relatively easily usable for developers.

4.3 Scene Rendering Optimisation Implementation

As discussed in section 3.2, a range of scene rendering optimisations have been successfully implemented and used in a research setting. Following on from section 4.1, a lack of standardised, cross platform implementations in the common engines is perhaps an obstacle to many developers which cannot justify implementing these techniques from scratch.

5 Other Hardware Limitations

Besides eye tracking hardware latency, there are some other hardware limitations to consider regarding cost and accessibility.

Headset	Cost (USD)	Cost Source
Vive Focus 3 Add-On	$250 (+$1300 for headset)	[32, 31]
PSVR2	$550	[20]
Meta Quest Pro	$1500	[15]
Vive Pro Eye	$1600	[34]
Apple Vision Pro	$3500	[2]
Varjo XR-4	$3990	[30]

Table 1: Costs for current eye tracking enabled VR hardware.

5.1 Adoption

A significant hardware limitation is simply adoption. Since only very few headsets currently implement eye tracking of any sort, there is little motivation for developers to implement foveated rendering support if there is not a significant enough user base. For headsets using SteamVR, the March 2024 Steam hardware survey[24] indicates the most common headset with eye tracking capabilities is the Meta Quest Pro, used by only 0.44% of VR users. The list of more common headsets accounts for 97.37% of users. This implies that no more than 2.64% of users have headsets with eye tracking; certainly less since many headsets ranked lower on the list lack the capability.

5.2 Cost

Another large hardware limitation is the cost of eye tracking hardware. I struggled to find any specific cost breakdowns for the required hardware, however some indication can perhaps be given by current product prices. An eye tracker add-on for the Vive Focus 3 is sold for $250[32], while costs for a selection of headsets with eye tracking (and further "premium" features, meaning the price is not solely indicative of the cost of eye tracking) are broken down in table 1. These factors would seem to imply costs remain quite high for this feature, likely out of reach for the $300[9] market of a product such as the Meta Quest 2 (used by 37.84% of SteamVR users[24], the largest share for any headset).

5.3 Eye Tracking Accuracy and Precision

If eye tracking precision and accuracy is not sufficient, a user may notice incorrect quality decreases due to incorrectly placed quality reductions from foveated rendering. Schuetz and Fiehler [23] performed an investigation into the eye tracking spatial accuracy and precision of the Vive Pro Eye headset. They found through their procedure that it exhibited a mean accuracy of 1.08°, outside the 0.5° they claim is typically required to accurately identify a user's fixation location. However on a per-user basis, some individuals

(usually those without glasses) saw individual mean gaze errors as low as 0.58°; close to the required threshold. These findings indicate that modern, high end, consumer VR eye tracking is nearing the required accuracy levels for most eye tracking, however is not yet ready for very small and rapid movements such as micro-saccades. Schuetz and Fiehler [23] therefore suggest that scenes in foveated applications should be designed with gaze targets large enough that these very small eye movements are avoided.

6 User Perception

When implementing foveated rendering, the developer needs to consider how much quality degradation can be allowed in the periphery before it can be perceived by the user and compromise their immersion.

6.1 Noticeable Foveation

Swafford et al. [26] performed a trial to determine conditions under which foveated rendering could be reliably distinguished from a fully rendered frame. They investigated for three foveated rendering techniques: variable resolution (adaptive resolution in fig. 1), ambient occlusion (shading simpltification in fig. 1), and tessellation(geometric simplification in fig. 1). They found that for all techniques, when peripheral quality decreases were minor, it was very difficult for a user to identify. More users were able to correctly detect the lower quality image when values were more extreme. Even with strong foveation, it was found that diminished peripheral resolution was difficult to detect. Users found distinguishing ambient occlusion and tessellation significantly easier as peripheral quality was diminished. These findings imply that resolution can be greatly diminished for larger performance uplifts, while shading and model quality must be more cautiously tuned to avoid noticeable quality loss.

6.2 Accessibility

In a previously mentioned experiment by Schuetz and Fiehler [23] on eye tracking accuracy with the Vive Pro Eye, the researchers found that lower accuracy and precision was yielded for users with glasses (though not with contact lenses). While not identifying a precise reason, the researchers suggested optical factors from the glasses' lenses and sensor obstructions by the glasses' frames as possibilities. While this increased error was not enough to indicate a significant usability impedance, this potential accessibility concern could be relevant in future system designs. Since many people wear glasses, issues regarding this could possibly block increased adoption.

7 Conclusion

In summary, the most significant limitation to adoption of foveated rendering in HMDs appears to be due to eye tracking hardware limitations. The most significant of these difficulties is attaining eye tracking data with comfortable visual latency (section 3.1), followed by some precision and accuracy limitation when determining gaze position (section 5.3). There are also some accessibility concerns regarding wearing glasses in existing eye tracking HMDs (section 6.2).

Additionally, high hardware costs likely drive away significant consumer adoption (section 5.2), while low user adoption rates (section 5.1) and a lack of reference implementations or standards (section 4.1) may drive away developer adoption.

On a more positive note, the actual rendering optimisations for foveated rendering have been well researched and implemented, with strong performance uplifts (sections 3.2 and 4.3), even with large peripheral quality degradation (section 6.1). This motivates continued research in the field under the hope that hardware limitations can be improved.

References

[1] Rachel Albert et al. "Latency requirements for foveated rendering in virtual reality". In: *ACM Transactions on Applied Perception (TAP)* 14.4 (2017), pp. 1–13.

[2] *Apple Vision Pro Purchase*. URL: https://www.apple.com/shop/buy-vision/apple-vision-pro.

[3] Detlev Arendt. "Evolution of eyes and photoreceptor cell types." In: *International Journal of Developmental Biology* 47.7-8 (2003), pp. 563–571.

[4] *Class XRGazeInteractor*. URL: https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@2.4/api/UnityEngine.XR.Interaction.Toolkit.XRGazeInteractor.html.

[5] *Eye Tracked Foveated Rendering*. URL: https://developer.oculus.com/documentation/unity/unity-eye-tracked-foveated-rendering/.

[6] *FoveatedRenderingMode*. URL: https://docs.unity3d.com/ScriptReference/Rendering.FoveatedRenderingMode.html.

[7] Brian Guenter et al. "Foveated 3D graphics". In: *ACM transactions on Graphics (tOG)* 31.6 (2012), pp. 1–10.

[8] *Introducing Apple Vision Pro: Apple's first spatial computer*. URL: https://www.apple.com/newsroom/2023/06/introducing-apple-vision-pro/.

[9] *Introducing Oculus Quest 2, the Next Generation of All-in-One VR*. URL: https://about.fb.com/news/2020/09/introducing-oculus-quest-2-the-next-generation-of-all-in-one-vr/.

[10] Matias K Koskela et al. "Instantaneous foveated preview for progressive Monte Carlo rendering". In: *Computational Visual Media* 4 (2018), pp. 267–276.

[11] Marc Levoy and Ross Whitaker. "Gaze-directed volume rendering". In: *Proceedings of the 1990 symposium on interactive 3d graphics*. 1990, pp. 217–223.

[12] Richard Li et al. "Optical gaze tracking with spatially-sparse single-pixel detectors". In: *2020 IEEE international symposium on mixed and augmented reality (ISMAR)*. IEEE. 2020, pp. 117–126.

[13] Tianxing Li, Qiang Liu, and Xia Zhou. "Ultra-low power gaze tracking for virtual reality". In: *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*. 2017, pp. 1–14.

[14] Xiaoxu Meng et al. "Kernel foveated rendering". In: *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1.1 (2018), pp. 1–20.

[15] *Meta Quest Pro Is Now Available*. URL: https://about.fb.com/news/2022/10/meta-quest-pro-is-now-available/.

[16] Bipul Mohanto et al. "An integrative view of foveated rendering". In: *Computers & Graphics* 102 (2022), pp. 474–501.

[17] *OpenXR Conformant Products*. URL: https://www.khronos.org/conformance/adopters/conformant-products/openxr.

[18] *OpenXREyeTracker*. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/API/Plugins/OpenXREyeTracker?application_version=5.0.

[19] *PlayStation VR2: The ultimate FAQ*. URL: https://blog.playstation.com/2023/02/06/playstation-vr2-the-ultimate-faq/.

[20] *PSVR2 Purchase*. URL: https://direct.playstation.com/en-us/buy-consoles/playstationvr2.

[21] Richard J Pumphrey. "The theory of the fovea". In: *Journal of Experimental Biology* 25.3 (1948), pp. 299–312.

[22] Kjetil Raaen and Ivar Kjellmo. "Measuring latency in virtual reality systems". In: *Entertainment Computing-ICEC 2015: 14th International Conference, ICEC 2015, Trondheim, Norway, September 29-Ocotober 2, 2015, Proceedings 14*. Springer. 2015, pp. 457–462.

[23] Immo Schuetz and Katja Fiehler. "Eye tracking in virtual reality: Vive pro eye spatial accuracy, precision, and calibration reliability". In: *Journal of Eye Movement Research* 15.3 (2022).

[24] *Steam Hardware Survey March 2024*. URL: https://web.archive.org/web/20240406095349/http://store.steampowered.com/hwsurvey/.

[25] Niklas Stein et al. "A comparison of eye tracking latencies among several commercial head-mounted displays". In: *i-Perception* 12.1 (2021), p. 2041669520983338.

[26] Nicholas T Swafford et al. "User, metric, and computational evaluation of foveated rendering methods". In: *Proceedings of the ACM Symposium on Applied Perception*. 2016, pp. 7–14.

[27] *The OpenXR Specification*. URL: https://registry.khronos.org/OpenXR/specs/1.0/html/xrspec.html#XR_EXT_eye_gaze_interaction.

[28] Justus Thies et al. "Facevr: Real-time facial reenactment and eye gaze control in virtual reality". In: *arXiv preprint arXiv:1610.03151* (2016).

[29] Ankur Tiwary, Muthuganapathy Ramanathan, and Jiri Kosinka. "Accelerated foveated rendering based on adaptive tessellation". In: *Eurographics 2020-Short Papers*. The Eurographics Association, 2020.

[30] *Varjo XR-4 Buy*. URL: https://b2b-store.varjo.com/.

[31] *Vive Focus 3 Buy*. URL: https://www.vive.com/us/product/vive-focus3/overview/.

[32] *Vive Focus Eye Tracker*. URL: https://business.vive.com/us/product/vive-focus-3-eye-tracker/.

[33] *Vive Foveated Rendering for Unity*. URL: https://github.com/ViveSoftware/ViveFoveatedRendering.

[34] *VIVE Pro Eye Launches Today In North America, Setting A New Standard For Enterprise Virtual Reality*. URL: https://www.vive.com/us/newsroom/2019-06-06/.

[35] Thomas Waltemate et al. "The impact of latency on perceptual judgments and motor performance in closed-loop interaction in virtual reality". In: *Proceedings of the 22nd ACM conference on virtual reality software and technology*. 2016, pp. 27–35.

[36] Lili Wang, Xuehuai Shi, and Yi Liu. "Foveated rendering: A state-of-the-art survey". In: *Computational Visual Media* 9.2 (2023), pp. 195–228.

[37] Cornelius Weber and Jochen Triesch. "Implementations and implications of foveated vision". In: *Recent Patents on Computer Science* 2.1 (2009), pp. 75–85.

[38] Martin Weier et al. "Foveated real-time ray tracing for head-mounted displays". In: *Computer Graphics Forum*. Vol. 35. 7. Wiley Online Library. 2016, pp. 289–298.

[39] Zipeng Zheng et al. "Perceptual model optimized efficient foveated rendering". In: *Proceedings of the 24th ACM Symposium on Virtual Reality Software and Technology*. 2018, pp. 1–2.