Poster: EyeSpy: Inferring Eye Gaze via Side-Channel Attacks Against Foveated Rendering

Paul Maynard, Harris Amjad, Camila Molinares, Bo Ji, and Brendan David-John
Department of Computer Science
Virginia Tech
Blacksburg, VA, USA
{*pmaynard, harris98, camilavm, boji, bmdj*}*@vt.edu*

***Abstract*—Eye tracking supports valuable virtual reality (VR) capabilities, including gaze-based interaction, biometric authentication, and dynamic foveated rendering (DFR). DFR preserves high perceived visual quality by rendering the foveal region near the user's gaze at high resolution while rendering the periphery at lower resolution. While gaze data enables many useful capabilities, it is inherently privacy-sensitive. Existing and proposed protections primarily regulate direct access to gaze data, but these protections are insufficient on DFR-enabled systems because DFR continues to use gaze internally to allocate rendering load. Because rendering content in the foveal region incurs greater GPU cost than rendering the same content in the periphery, an app can induce measurable fluctuations in GPU workload by moving content through the field of view and observing app-visible GPU performance metrics. We present *EyeSpy*, a scan-based side-channel attack that sweeps transparent high-cost objects (HCOs) across the field of view and correlates the resulting changes in GPU performance metrics with HCO positions to infer gaze without using eye-tracking APIs. Across Meta Quest Pro, Varjo XR-4, and desktop settings, EyeSpy achieves mean gaze prediction errors ranging from 1.1 degrees to 4.4 degrees, showing that the attack generalizes across hardware platforms, rendering engines, and foveated rendering pipelines. Our results show that restricting direct gaze access alone is insufficient: if DFR is being used, apps can still infer privacy-relevant gaze through GPU performance metrics.**

1. Introduction

Many modern VR headsets now include eye tracking to enable features such as gaze-based interaction, intent modeling, and dynamic foveated rendering (DFR). DFR is especially useful as it reduces GPU load by rendering only the region near the user's gaze at higher fidelity and periphery at lower fidelity, without perceptible quality loss [1].

While eye tracking provides valuable capabilities for VR, it can also leak sensitive user information if not properly protected. Prior work shows that eye movements can reveal attention and cognitive state, support identity inference, and expose broader privacy-relevant characteristics and risks [2]. Existing defenses protect the gaze stream exposed to apps through permission controls, gatekeeper-style APIs [3], and differential privacy [4]. However, these defenses treat gaze as an explicit data channel requested directly by the app.

We show that even when apps are denied access to gaze data, gaze can still leak indirectly on DFR-enabled systems. DFR maintains a high-detail foveal region that follows the user's gaze, and rendering content in that region incurs greater GPU cost than rendering the same content in the periphery. Consequently, when an app moves scene content through the field of view, app-accessible GPU performance metrics vary with that content's overlap with the foveal region. EyeSpy exploits this variation to infer where the user is looking without directly accessing gaze data.

2. EyeSpy Overview

EyeSpy exploits DFR-induced workload variation to infer gaze without accessing eye-tracking APIs. The attack inserts transparent high-cost objects (HCOs) into an attacker-controlled scene and sweeps them across the user's field of view one axis at a time. One HCO scans horizontally while the other remains stationary; the roles then alternate for vertical scanning. Since rendering an object in the foveal region incurs greater GPU cost than rendering the same object in the periphery, HCO–fovea overlap produces measurable variation in app-accessible GPU performance metrics such as frame rate or frame time (see Fig. 1). By logging an affected GPU performance metric together with the HCO position over time, EyeSpy correlates metric variation with known HCO positions to infer time-stamped gaze coordinates.

A key challenge is designing HCOs that induce enough workload variation to support inference while remaining imperceptible to the user. If too lightweight, the variation is buried in background rendering noise; if too costly, visible artifacts, judder, or responsiveness degradation can expose the attack. EyeSpy addresses this tradeoff through a radar-inspired scan design that uses imperceptible but high-impact HCOs, sequential horizontal and vertical scans with controlled scan times, and lightweight signal processing to convert noisy GPU performance traces into gaze estimates. The attack does not require privileged system access or specialized measurement tools. Instead, it relies only on

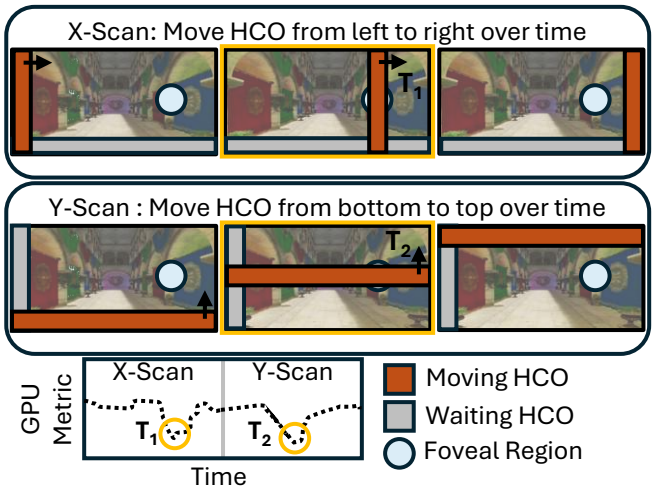


Figure 1. Attack process: One HCO scans while the other waits. Dips appear at T_1 and T_2 when the scanning HCO overlaps with the fovea during the X- and Y-scans respectively. The HCOs are invisible but colored here for illustration.

availability of app-accessible GPU performance metrics and DFR support on the target platform.

3. Evaluation

We evaluate EyeSpy across representative DFR environments spanning both VR and desktop settings. In VR, we implement the attack on a stand-alone Meta Quest Pro using Unity and on a tethered Varjo XR-4 using Unreal. In a desktop setting, we instantiate the same scan-based pipeline in Godot with simulated DFR driven by DGaze and ET-DK2 gaze traces [5], [6]. Across these environments, EyeSpy achieves mean gaze prediction errors ranging from approximately 1.1° to 4.4°, depending on axis and platform. Table 1 shows the errors for both headsets (running at approx 70 FPS), and Table 2 shows errors on the desktop implementation by FPS. Notably, the attack performed similarly across headsets using different hardware, DFR implementation, and game engines; the desktop implementation of the attack shows error decreases as frame rate increases, reaching approximately 1.19° on X and 0.91° on Y at 200 FPS. These results show that the attack generalizes across hardware platforms, rendering engines, and foveated rendering pipelines.

TABLE 1. MEAN AND SD OF ABSOLUTE X AND Y GAZE ERROR FOR MQP AND VARJO XR-4 (LEAVE-ONE-PARTICIPANT-OUT EVALUATION)

Headset	X error (°)	Y error (°)
MQP	4.36 (± 5.89)	2.88 (± 3.60)
Varjo XR-4	3.98 (± 4.81)	3.24 (± 4.18)

4. Defense Mechanisms

We propose a defense that detects GPU side-channel leakage. For detection, we evaluate two lightweight models:

TABLE 2. MEAN AND SD OF ABSOLUTE X AND Y GAZE ERROR IN THE DESKTOP EVALUATION AT 120, 160, AND 200 FPS.

FPS	X Error (°)	Y Error (°)
120	3.50 (±6.51)	1.59 (±2.26)
160	1.65 (±2.90)	1.12 (±1.75)
200	1.19 (±1.66)	0.91 (±1.47)

a supervised logistic regression model and an unsupervised K-means clustering model. The logistic regression model uses simple statistical features and achieves an F1 score of 0.93, while K-means detects anomalous patterns without labels and achieves up to 0.99. Once detected, potential mitigation strategies may include enabling VSync to limit frame-rate leakage, or artificially modulating GPU performance to obscure the attack's signature and prevent gaze inference.

5. Conclusion

EyeSpy is a novel side-channel attack that infers gaze by exploiting DFR-induced workload variation and using app-accessible GPU performance metrics as a side channel. Our results show that the attack is feasible across multiple platforms, rendering pipelines, and game engines. More broadly, they show that restricting direct gaze access alone is insufficient when apps can induce and observe gaze-dependent rendering-cost variation through GPU performance metrics.

Acknowledgments

Authors acknowledge funding from the National Science Foundation (CNS-2350116) and the Commonwealth Cyber Initiative.

References

[1] 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 Transactions on Graphics (TOG)*, vol. 35, no. 6, pp. 1–12, 2016.

[2] J. L. Kröger, O. H.-M. Lutz, and F. Müller, "What does your gaze reveal about you? on the privacy implications of eye tracking," in *IFIP International Summer School on Privacy and Identity Management*. Springer, 2020, pp. 226–241.

[3] B. David-John, D. Hosfelt, K. Butler, and E. Jain, "A privacy-preserving approach to streaming eye-tracking data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 5, pp. 2555–2565, 2021.

[4] A. Liu, L. Xia, A. Duchowski, R. Bailey, K. Holmqvist, and E. Jain, "Differential privacy for eye-tracking data," in *Proceedings of the 11th ACM Symposium on Eye Tracking Research & Applications*, 2019, pp. 1–10.

[5] Z. Hu, S. Li, C. Zhang, K. Yi, G. Wang, and D. Manocha, "Dgaze: Cnn-based gaze prediction in dynamic scenes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 5, pp. 1902–1911, 2020.

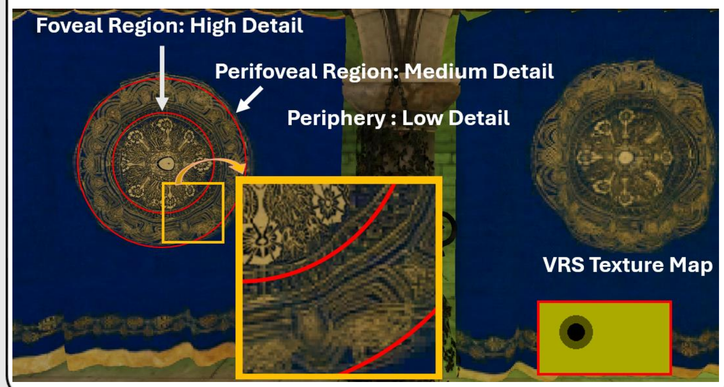
[6] B. David-John, R. Eiris, M. Gheisari, O. L. Meur, D. Hosfelt, K. Butler, and E. Jain, "Et-dk2 dataset," Mar. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.4642612

EyeSpy: Inferring Eye Gaze via Side-Channel Attacks Against Foveated Rendering

Paul Maynard, Harris Amjad, Camila Molinares, Bo Ji, and Brendan David-John
{pmaynard, harris98, camilavm, boji, bmdj} @vt.edu
Commonwealth Cyber Initiative

1. What is Dynamic Foveated Rendering (DFR)?

- High perceived resolution at a fraction of the cost.
- Gaze determines GPU resource allocation.



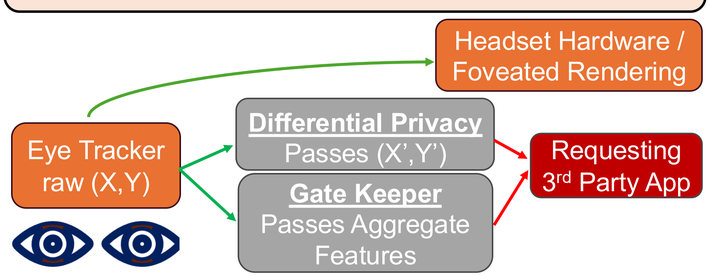
2. Why Gaze Privacy Matters, and why current protections are insufficient?

Gaze data can reveal:

- Demographics
- Intent
- Mental State
- Biometric identifiers
- Attention

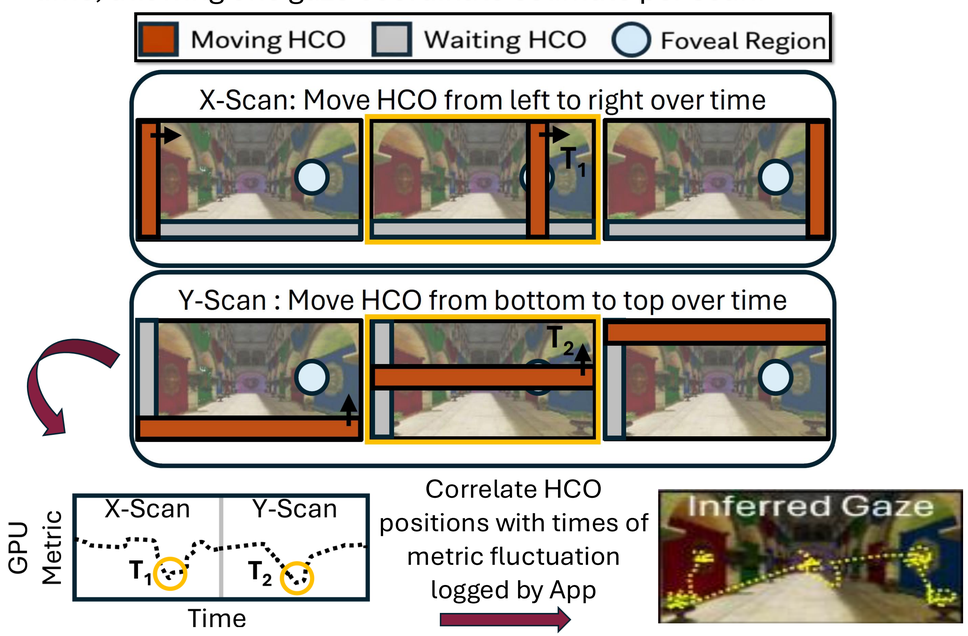
Possible Uses:

- Profiling user for targeted ads, phishing, other malicious attacks
- Re-identification across platforms
- Selling data to 3rd party



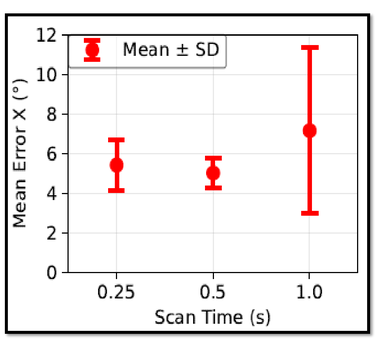
3. EyeSpy Attack

- Invisible high-render-cost objects scan FOV in X and Y directions.
- HCOs inevitably pass through the foveal region, inducing additional GPU workload.
- Additional GPU workload is detectable via app-visible performance metrics (e.g., frame time or framerate).
- Time of additional workload is correlated with HCO position at that time, allowing one gaze coordinate estimate per scan.



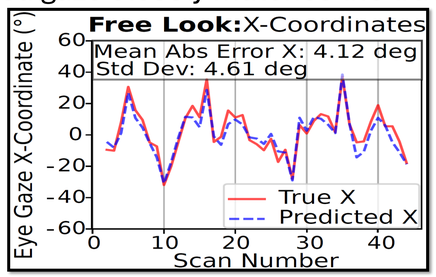
4. Limitations

- Enables downstream user inferences, not directly though.
- Scan speed inversely correlated with accuracy causing a tradeoff.
- Temporal recovery of gaze allows for some fixation but not saccadic activity.



5. Results

- Inferred gaze closely matches the true.



- Inferred gaze signals help identify some fixations, revealing regions of interest.

3 Platforms with different DFR implementations and game engines:

- Meta Quest Pro (using Unity)
- Varjo XR-4 (using Unreal)
- Desktop (using Godot at 3 FPS settings)

Headset	X error (°)	Y error (°)
MQP	4.36 (± 5.89)	2.88 (± 3.60)
Varjo XR-4	3.98 (± 4.81)	3.24 (± 4.18)

FPS	X Error (°)	Y Error (°)
120	3.50 (±6.51)	1.59 (±2.26)
160	1.65 (±2.90)	1.12 (±1.75)
200	1.19 (±1.66)	0.91 (±1.47)

6. Defense

- Detection: Logistic Regression (0.93) + K-means (0.99) for anomaly detection.
- Mitigation: Potential strategies include VSync or controlled GPU noise to obscure leakage.

7. Takeaways

- DFR uses gaze internally to allocate rendering resources.
- EyeSpy exploits DFR by inducing GPU metric fluctuations via HCO-fovea interaction.
- Metric fluctuations are correlated with logged HCO position to infer gaze.