© 2025 IEEE. This is the author's version of the article that has been published in the proceedings of IEEE Visualization conference. The final version of this record is available at: xx.xxxx/TVCG.201x.xxxxxxx/

FovealNet: Advancing AI-Driven Gaze Tracking Solutions for Optimized Foveated Rendering System Performance in Virtual Reality

Wenxuan Liu, Monde Duinkharjav, Qi Sun, Sai Qian Zhang*

Tandon School of Engineering, New York University

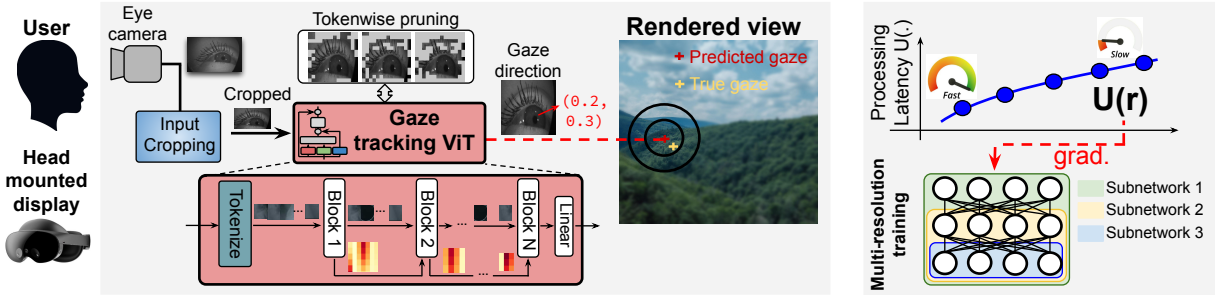

Figure 1: FovealNet for efficient gaze-tracked foveated rendering system operation.

Abstract

Leveraging real-time eye-tracking, foveated rendering optimizes hardware efficiency and enhances visual quality virtual reality (VR). This approach leverages eye-tracking techniques to determine where the user is looking, allowing the system to render high-resolution graphics only in the foveal region—the small area of the retina where visual acuity is highest, while the peripheral view is rendered at lower resolution. However, modern deep learning-based gaze-tracking solutions often exhibit a long-tail distribution of tracking errors, which can degrade user experience and reduce the benefits of foveated rendering by causing misalignment and decreased visual quality.

This paper introduces *FovealNet*, an advanced AI-driven gaze tracking framework designed to optimize system performance by strategically enhancing gaze tracking accuracy. To further reduce the implementation cost of the gaze tracking algorithm, FovealNet employs an event-based cropping method that eliminates over 64.8% of irrelevant pixels from the input image. Additionally, it incorporates a simple yet effective token-pruning strategy that dynamically removes tokens on the fly without compromising tracking accuracy. Finally, to support different runtime rendering configurations, we propose a system performance-aware multi-resolution training strategy, allowing the gaze tracking DNN to adapt and optimize overall system performance more effectively. Evaluation results demonstrate that FovealNet achieves at least 1.42× speed up compared to previous methods and 13% increase in perceptual quality for foveated output.

Index Terms: Foveated rendering, Gaze tracking, AR/VR.

1 Introduction

Human visual acuity varies across the visual field. The fovea, the central region of the retina, is responsible for our sharpest vision. This region, although small, is densely packed with photoreceptor cells, allowing us to perceive fine details and vibrant colors within our direct line of sight. As we move away from the fovea, our visual acuity decreases rapidly, meaning that the peripheral vision is less sensitive to fine details. Foveated rendering leverages this phenomenon by allocating more computational resources to the fovea while reducing detail in the periphery. This technique significantly enhances system performance by lowering the rendering workload without compromising the perceived visual quality, making it a critical innovation for applications such as virtual reality (VR) [1, 2, 3], video encoding [4, 5, 6] and gaming [7, 8, 9]. By aligning rendering fidelity with human gaze patterns, foveated rendering optimizes both visual experience and computational efficiency.

Therefore, VR systems commonly require gaze-tracking for foveated rendering, which is usually fulfilled with a deep neural networks (DNNs). By precisely determining the user's point of focus in real-time, the gaze-tracked foveated rendering (TFR) system can precisely catch the location of the foveal region which is rendered with the highest resolution, followed by the *inter-foveal region* and *peripheral region*, which then will be rendered from fine to coarse level, as shown in the left part of Fig. 1.

Accurate gaze tracking is fundamental to the successful implementation of TFR. Without reliable gaze tracking, the system cannot accurately adapt to the user's visual focus, leading to potential misalignments between rendered detail and real gaze position, which can result in noticeable artifacts and diminished user experience. Therefore, integrating robust gaze tracking mechanisms is imperative for optimizing performance and ensuring seamless, high-fidelity visuals in TFR. Although several prior studies have proposed AI-based gaze tracking solutions that perform well on test datasets [10, 11, 12, 13, 14], our experiments in this paper show that these solutions can considerably reduce the efficiency of TFR. This is because, despite having low average gaze tracking errors, the errors often follow a long-tail distribution, resulting in substantial inaccuracies in detecting the user's gaze location in various scenarios. These errors can further cause the rendered foveal region to be misaligned with the user's actual gaze, leading to decreased visual quality and undermining the intended performance gains of foveated rendering, ultimately diminishing the user experience.

To tackle this challenge, we introduce a novel training approach that integrates TFR system performance directly into the gaze tracking DNN's training process, thereby optimizing the overall performance. Specifically, we focus on minimizing system latency in this work, as latency is a key factor in VR environments. Moreover, our approach can be extended to optimize various system performance

*Corresponding author

© 2025 IEEE. This is the author's version of the article that has been published in the proceedings of IEEE Visualization conference. The final version of this record is available at: xx.xxxx/TVCG.201x.xxxxxxx/

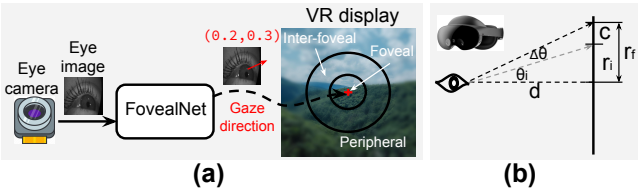

Figure 2: (a) TFR system configuration. (b) Foveated rendering in VR device.

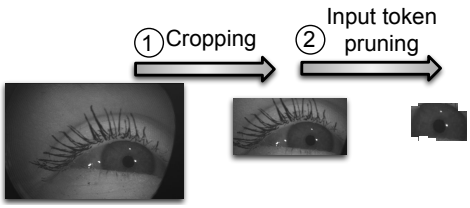

Figure 3: Given an input eye image, FovealNet first crops the image to remove background patches (step 1). The remaining patches undergo fine-grained token-wise pruning to eliminate unimportant tokens (step 2), and the remaining tokens are processed by the ViT.

metrics in different TFR scenarios (e.g., power consumption).

Furthermore, previous studies have highlighted the significance of implementation overhead for gaze-tracking DNNs [11, 2], as this additional cost can often outweigh the performance benefits gained from TFR. To reduce the computational complexity of gaze tracking DNN, we develop a simple approach that focuses on efficiently capturing the eye region centered around the pupil, minimizing computation for irrelevant peripheral pixels. This event-driven design also enables efficient reuse of buffered gaze-tracking results during the execution. In addition, we introduce fine-grained pruning mechanisms targeting input tokens within the gaze-tracking model, reducing unnecessary computation in non-informative areas such as the eyelashes.

Finally, the hardware processing latency for image rendering and tracking often demonstrates dynamic behavior, influenced by user modifications to system settings and resource allocation with other applications. This variability necessitates a dynamic configuration of the gaze tracking DNN to ensure optimal system performance. In address this, we introduce a multi-resolution DNN training framework that trains the gaze-tracking DNN across various configurations simultaneously, as shown in the right part of Fig. 1. During operation, the most suitable DNN configuration is chosen based on the current system conditions, facilitating optimal dynamic performance for TFR. Our contributions are summarized as follows:

- We propose *FovealNet*, an AI-driven gaze tracking solutions for optimized system performance of TFR. FovealNet employs a performance-aware training strategy by directly optimizing the TFR system latency during its training phase.
- To reduce the implementation cost of FovealNet, we introduce an efficient input cropping method that focuses on extracting the central eye region. Furthermore, we propose an adaptive input token pruning technique for the transformer-based gaze-tracking DNN, achieving superior computational efficiency while maintaining the tracking performance.
- To accommodate variations in TFR system configurations, we propose a multi-resolution DNN training framework that allows the resulting gaze-tracking DNN to dynamically adapt by selecting the optimal configuration based on current system conditions.

2 Background and Related Work

2.1 Gaze Tracking Algorithms

Generally, gaze tracking methods can be broadly classified into model-based and appearance-based approaches [15, 16]. Model-based techniques estimate gaze direction by utilizing a 3D eye model that mimics physiological structures [17, 18, 19, 20]. These approaches generally involve two stages: (1) employing an eye feature extraction neural network to produce salient eye features and fitting a geometric eye model, and (2) predicting the gaze direction based on the fitted eye model. Essentially, model-based methods transform the gaze tracking problem as an eye segmentation task. Most of model-based approaches utilize U-Net with convolutional operations for eye feature extraction, with most of the computations arising from this process [11, 10, 12, 21]. Although previous studies have shown high accuracy in the eye segmentation task [22], the geometric methods used to derive gaze direction from the fitted eye model can be prone to inaccuracies, inherently introducing an error of more than 2°, compared to the ground truth. This is primarily due to two reasons: (1) inaccuracies in fitting the eye center and radius during the eye model initialization phase, and (2) during the optimization phase, the geometric model imposes additional constraints on the potential shapes and positions of the pupils, further lead to mismatches between the output and the ground truth [17].

In contrast, appearance-based gaze tracking methods directly use eye images as input and learn a mapping from these images to gaze direction [15, 23, 24]. Compared to model-based approaches, these methods typically require larger amounts of training data. The scale and complexity of the required training data have led to the development of a wide range of learning-based techniques, including linear regression [25], random forests [26], and k-nearest neighbors (KNN) [26, 27], and CNNs [28, 29]. Vision Transformers (ViTs) [30], renowned for their state-of-the-art performance in various vision tasks, have also been adopted for gaze tracking [31].

However, the significant computational demands of ViTs present a major challenge for real-time gaze tracking. To tackle this, we propose a system performance-driven, multi-resolution gaze tracking transformer network. To the best of our knowledge, this work represents the first attempt to train a gaze tracking DNN while considering the system performance of TFR.

2.2 Gaze-Tracked Foveated Rendering

The growing popularity of virtual reality has led to a significant demand for rendering high-resolution images on resource-limited hardware platforms, such as head-mounted displays (HMDs) [2, 32]. In these contexts, minimizing system latency is crucial to prevent visual distortions and visual artifacts. If the system fails to keep up, users may experience a disconnect between what they see and what they feel, leading to a quality degradation of user experience [33, 34]. Early HMDs employed full-resolution rendering, where each pixel within the user's field of view was rendered uniformly. While this method maintained visual quality, it resulted in unnecessary computational overhead and increased system latency.

TFR is a VR technique that optimizes rendering using gaze-tracking, typically powered by DNNs. It detects the user's gaze location in real-time, rendering the *foveal region* at the highest resolution (Fig. 2(a)), while surrounding areas, namely the *inter-foveal region* and *peripheral region*, are rendered from fine to coarse level without introducing visual artifacts [32, 34, 35]. As depicted in Fig. 2(b), in TFR, the radius r_f of the foveal region (in pixels) is set based on $r_f = r_i + c = \rho d \cdot \tan(\theta_i + \Delta\theta) = d \tan(\theta_f)$ [1], where ρ represents the display's pixel density, d is the distance between the user and the screen, θ_i is the eccentricity angle subtended by the fovea, and $\theta_f = \theta_i + \Delta\theta$ is the resultant eccentricity angle that

[1] This formula is derived by assuming the gaze is positioned at the center of the front view, representing the maximum radius of the rendering region.

© 2025 IEEE. This is the author's version of the article that has been published in the proceedings of IEEE Visualization conference. The final version of this record is available at: xx.xxxx/TVCG.201x.xxxxxxx/

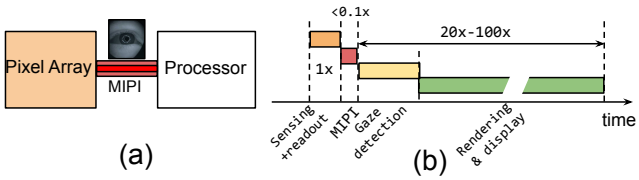

Figure 4: (a) TFR system configuration. (b) Latency breakdown (normalized) of TFR.

accounts for the gaze tracking error $\Delta\theta$. The radius of the foveal region without tracking error is $r_i = d\tan(\theta_i)$, an error constant $c = d\tan(\theta_f) - d\tan(\theta_i)$ is used to represent the changes on foveal region radius caused by $\Delta\theta$. The value of θ_i is typically adjusted differently [36, 37, 2] depending on the experimental results from user studies. For example, in [36], θ_i is set to 5.2° around the gaze location, while the inter-foveal region covers the area from 5.2° to 17°. In the context of HMD, the distance between the user and the screen d is remain fixed.

2.3 DNN Pruning

Pruning techniques are commonly used in DNNs to reduce memory and computational costs. Beyond weight pruning, research has focused on pruning intermediate tokens in ViT [38, 39]s. For instance, SPViT [40] removes redundant tokens with a token selector; S^2ViTE [41] uses sparse training to prune tokens and attention heads; and Evo-ViT [42] employs a slow-fast token evolution mechanism to retain essential information.

Although previous work has utilized cropping techniques to remove redundant surrounding pixels from eye images [11], the specific characteristics of the gaze-tracking task allow us to further eliminate redundant input tokens within near-eye images, such as those representing the eyelashes. Compared to tokens depicting the iris and pupil, these elements are relatively irrelevant to gaze tracking results [17]. This insight led us to implement fine-grained token-wise pruning on top of the cropped input of ViT and its intermediate activations (step 2 in Fig. 3). Our proposed token-wise pruning approach ranks input tokens based on their importance scores (attention scores) with respect to the final gaze prediction and eliminates unimportant tokens, leading to a significant reduction in computational costs with negligible accuracy impact.

3 Motivation

3.1 TFR System and Pipeline

A TFR system in modern VR devices (e.g., HMDs) usually comprises three main components (Fig. 4(a)): a near-eye camera, a host processor, and an interconnection link (MIPI [43]). A typical TFR pipeline is shown in Fig. 4(b). The process begins with capturing an eye image using a near-eye monochrome camera. This image is then preprocessed by the image signal processor (ISP) and readout before being transferred to the host processor via the MIPI connection. After the host processor receives the image, it is sent to the eye-tracking DNN, which estimates the gaze direction. This estimated gaze direction is then utilized to guide the foveated rendering process to produce the rendered VR scene.

Fig. 4 (b) also provides an approximate latency breakdown of the TFR process. Camera sensing and MIPI communication account for a small fraction of the total latency, approximately 1 ms [44, 45, 46, 47] and less than 1 ms [48, 49], respectively. In contrast, the gaze detection, along with the subsequent rendering and display process, usually consumes a much larger portion (20-100× longer) of the overall latency, based on the studies from [2, 50].

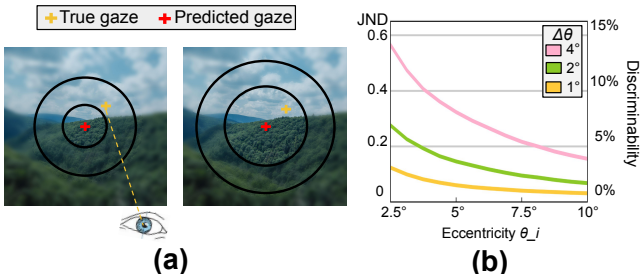

Figure 5: (a) Visual quality degradation due to tracking error, and then the foveal region is enlarged for better visual quality. (b) Observer's ability to discriminate foveated image with and without tracking error, measured by JND. x-axis indicates the eccentricity angle subtended by the fovea. Left y-axis is the JND score, right side of y-axis is the discriminability probabilities from ground truth.

3.2 Foveated Visual Quality Assessment

The primary goal of TFR is to reduce computational load while maintaining visual fidelity. To measure the impact on the visual fidelity, traditional visual similarity metrics typically model only image-space context, without considering the observer's visual field [51, 52]. An emerging body of literature introduces eccentricity angle relative to the fovea into metrics to assess the quality of TFR [53, 54, 55]. These computational metrics allow for the systematic prediction of foveated image quality, eliminating the need for user studies, which often suffer from limited sample sizes and subjective rating variances. Following other work in the VR community [56, 57, 58], we use the FovVideoVDP metric [54] to assess visual quality in terms of just-noticeable difference (JND) [59]. Expressing image similarity in terms of JND units offers the advantage of clearly quantifying the probability shift in a population's ability to distinguish between test and reference image pairs, as illustrated in Fig. 5(b). We observe that, for a fixed eccentricity angle θ_i, a larger tracking error $\Delta\theta$ results in a greater increase in discriminability from the ground truth (GT) and a higher JND, indicating poorer foveated image quality.

3.3 Latency Evaluation of Previous Approaches

While previous work on gaze tracking DNNs has made significant progress in reducing prediction errors, almost all of these approaches focus on minimizing the **average tracking error** across the training set, resulting in a low average tracking error during execution [11, 31, 28, 29, 18, 19].

To study the distribution of the tracking errors $\Delta\theta$ produced by the tracking algorithms, we train and test various gaze tracking DNNs proposed in [11, 31, 28, 29], on the OpenEDS 2020 dataset [60]. Fig. 6(a) shows the average tracking errors along with the 95th percentile of the tracking errors on the test dataset of OpenEDS 2020 dataset for five methods: DeepVOG [10], Seg [11], ResNet-based [29], IncResNet-based [28] and NVGaze [31]. Our observations are as follows: first, all methods, except NVGaze [31], achieve an small average value for $\Delta\theta$, ranging between 2° and 2.5°, with NVGaze showing a higher average tracking error of 9.2°. Notably, the tracking errors exhibit a long-tail distribution, characterized by a high 95th-percentile tracking error, suggesting that these gaze-tracking DNNs may incur large errors for certain eye images. As a result, the foveated image generated based on the predicted gaze may not align well with what the user is actually looking at, leading to significant degradation in visual quality for the user, as shown in left part of Fig. 5(a). As presented in Fig. 5(b), a larger error will further lead to a decrease on visual quality. Unfortunately, most previous work has focused solely on minimizing average gaze tracking performance, and none have optimized gaze

© 2025 IEEE. This is the author's version of the article that has been published in the proceedings of IEEE Visualization conference. The final version of this record is available at: xx.xxxx/TVCG.201x.xxxxxxx/

tracking DNNs by considering the impact of tracking error distribution in foveated rendering applications.

To maintain visual quality without affecting user experience, the high-resolution foveal and inter-foveal regions must be enlarged to compensate for gaze tracking errors, as shown in right part of Fig. 5(a). However, this increases rendering latency, as rendering larger foveal regions at higher resolution raises computational costs.

To investigate how the size of the central foveal region influences the performance of the TFR system, we analyze the changes in rendering latency across different foveal region sizes. Latencies are measured on two types of devices: the Meta Quest Pro [61] and the Quadro RTX 3000 Mobile GPU [62], which has been utilized by the early work to simulate the behavior of the UMD [50]. For the Meta Quest Pro, we randomly select 50 frames from the Digital Combat Simulator World (DCS) VR [63] to measure their rendering latency. For mobile GPU, we employ NVIDIA's Variable Rate Shading (VRS) for rendering and measure the average rendering latency across four scenes, which include a mix of indoor and outdoor environments: Bistro (Outdoor), Sponza (Indoor), Classroom (Indoor), and San-Miguel (Indoor and Outdoor). For both devices, we apply three different rendering resolutions, 720×1280 (720P), 1080×1920 (1080P) and 1440×2560 (1440P).

Following the settings used in previous works [2, 37, 32], we initially set the eccentricity angle θ_f for the foveal region and inter-foveal regions to 5° and 20°, respectively. Subsequently, we gradually increase θ_f to 25° and the inter-foveal eccentricity to 40° to account for the gaze tracking errors generated by various gaze tracking DNNs. The resolution drop of the inter-foveal region and the peripheral regions are set to 4× and 16×, respectively. We record the average rendering latencies as the eccentricity angles change. The results are shown in Fig. 6(c). From the data, it is evident that rendering latency increases superlinearly as the eccentricity angle θ_f grows for both the Meta Quest Pro and the Mobile GPU. Notably, the rendering latency nearly doubles when the eccentricity angle reaches 20°, compared to when the eccentricity angle is 5°. Consequently, a large gaze tracking error $\Delta\theta$ will inevitably cause a larger central foveal region to ensure the visual quality, which in turn results in a significant increase in latency overhead.

To analyze the latency overhead caused by gaze tracking errors for each method, Fig. 6(b) shows the rendering latency at a resolution of 1080×1920. We set $\Delta\theta$ to represent either the average tracking error or the 95th percentile of the tracking errors for each method, using the test dataset from the OpenEDS 2020 dataset. Our observations reveal that when $\Delta\theta$ is set to the average gaze tracking error, most algorithms achieve very low rendering latency. However, when considering the 95th percentile of the tracking error, all algorithms experience an average of 2.2× increase in tracking error, with some algorithms (such as NVGaze [31], Seg [11], and ResNet-based [29]) even exceeding 20 ms, failing to achieve real-time rendering requirement of 60 FPS, according to [64].

According to Fig. 4, the per-frame latency T_{total} for TFR can be expressed as $T_{total} = T_{sensing} + T_{comm} + T_{tracking} + T_{fr}$, where $T_{sensing}, T_{comm}, T_{tracking}, T_{fr}$ represent the latency of camera sensing, MIPI communication, gaze tracking and foveated rendering, respectively. As described in the Sec. 3.1, $T_{sensing}$ and T_{comm} is small compared with $T_{tracking}$ and T_{fr}. We will describe the efficient gaze tracking DNN design to minimize $T_{tracking}$ in Sec. 4.1 and Sec. 4.2, and discuss a novel training framework of the gaze tracking DNN to minimize T_{fr} in Sec. 4.3.

3.4 Tracking Performance and Latency Tradeoffs in TFR

In practice, rendering resolution settings for in AR/VR devices can vary significantly during usage, depending on the context and the specific needs of the experience. For example, in highly detailed, visually rich environments like gaming or virtual simulations, users may prefer higher resolutions to capture intricate details and enhance immersion. However, in scenarios requiring rapid interaction or movement, such as fast-paced games or real-time collaboration, users might prioritize performance over maximum resolution to maintain smoothness and responsiveness. For popular HMD, such as Apple Vision Pro, supports a render resolution setting as high as 3860×3200 [65]. These different resolutions place varying workloads on the host processor, resulting in different rendering times, as presented in Fig. 6(c).

A previous study [50] indicates that the additional implementation overhead associated with executing gaze tracking DNN might surpass the savings achieved through efficient foveated rendering. Specifically, if we denote T_{full} as the execution latency for processing an image frame and rendering it at full resolution, then $T_{tracking}$ could exceed $T_{full} - T_{fr}$. Moreover, since users may dynamically adjust the rendering resolution, T_{fr} can also vary dynamically. Therefore, it is essential to appropriately balance the running costs and tracking accuracy of the gaze tracking DNN to minimize the overall latency. To ensure that the gaze tracking DNN remains adaptable and performs effectively, we have developed a multi-resolution DNN training approach (Sec. 4.4) that simultaneously optimizes numerous sub-networks across a variety of DNN architectures (Fig. 9 (a)). This training method produces a multi-resolution DNN that can produce sub-networks at various resolutions during runtime, enabling selecting the optimal gaze tracking DNN configuration based on the current TFR system conditions.

4 FovealNet Design

The overall structure of FovealNet is illustrated in Fig. 7. When an eye image is received, the eye image is cropped using the method outlined in Sec. 4.1. The cropped image is then passed to the gaze tracking ViT, which is explained in Sec. 4.2 and Sec. 4.3. Finally, the multi-resolution training approach is described in Sec. 4.4.

4.1 Event-based Cropping for Efficient TFR

Eye images captured by near-eye cameras often contain redundant pixels (e.g., background, facial muscles) that are irrelevant for gaze tracking prediction. These pixels can negatively impact the prediction results and increase the computational cost. To tackle this, we propose an event-based analytical approach that efficiently crops the informative regions of the eye.

4.1.1 Region Cropping Algorithm

Given that the pupil is the most relevant area for human gaze, we focus on cropping the informative region of the input frame around the pupil's position. To do this, we employ an efficient analytical approach to precisely detect the pupil location, enabling us to accurately crop a fixed-size region centered on the detected pupil.

We begin by applying a masking process to eliminate the background region around the image edges. Then, leveraging the prior knowledge that the pupil is typically darker than the surrounding sclera and iris [66], we perform inverse binarization to emphasize the darker regions of the image. Next, we apply morphological opening to reduce noise in the image. At this stage, only the pupil and other dark regions like eyelashes remain, as shown in the connected components (CC) maps in Fig. 8. Given the fact that the pixel density in the pupil region is significantly higher than in other areas. Thus, we can identify the pupil by searching for the largest connected component (LCC) in the image and using the center of this component to represent the pupil's center. Once the LCC is identified in the image, since most pupils are either circular or elliptical [66], we apply an operation, *is_pupil*, to determine the shape of the region, by calculating the roundness [67] of this region, if the value indicates an approximate circle or ellipse, we classify it as a pupil. Once the presence of the pupil is confirmed, a rectangle of predefined size (450×200) is fitted around the pupil center to crop the informative region. If the rectangle touch the boundary of

© 2025 IEEE. This is the author's version of the article that has been published in the proceedings of IEEE Visualization conference. The final version of this record is available at: xx.xxxx/TVCG.201x.xxxxxxx/

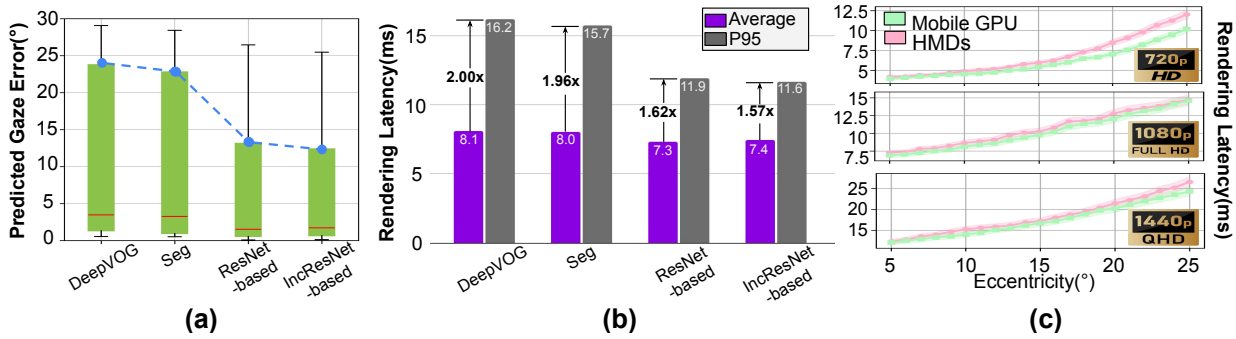

Figure 6: (a) Predicted gaze error distributions on the OpenEDS2020 dataset, showing mean, 5th, 95th percentiles, min, and max angular errors. NVgaze results were excluded due to high tracking error and inconsistent performance. (b) Rendering latency for existing methods in both average and max error scenarios with a resolution of 1080×1920. (c) Rendering latency increases with eccentricity on HMD and GPU at resolutions 720×1280, 1080×1920, and 1440×2560, with the shaded area showing the 5%-95% confidence interval.

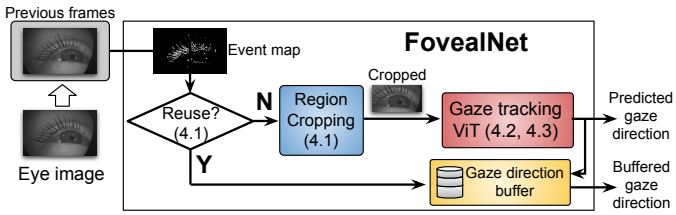

Figure 7: Overall architecture of FovealNet.

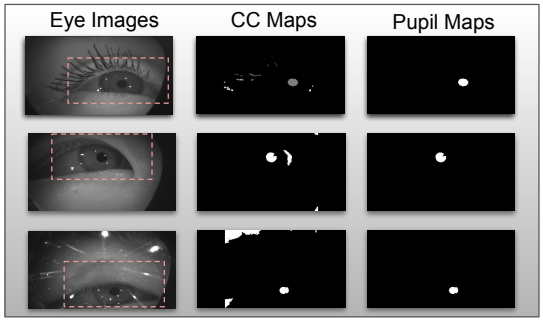

Figure 8: Examples of pupil-centered cropping with intermediate results: connected components (CC) maps and pupil maps. The pink dashed rectangles represent the cropped region.

the image, the rectangle will be translated accordingly (last row of Fig. 8) to ensure it properly fits within the region.

4.2 Efficient Gaze Tracking Network

The cropped eye images containing informative content are first resized to a smaller square (224×224) and then processed by the gaze tracking DNN to predict gaze direction. FovealNet employs a vision transformer (ViT) architecture [30] for this task, as it provides superior performance compared to convolutional neural network (CNN)-based architectures. The ViT operates by dividing the input image into patches, which are then tokenized and appended with positional information, the outputs are then passed through the transformer block. The ViT contains 8 transformer block, each block consists of 6 heads with an embedding dimension of 384. We also modify the original ViT by replacing the classifier MLP layers with a sequence of linear layers to output the 2D gaze direction.

A key advantage of ViT over CNN is its ability to fine-grain prune input tokens, enabling the removal of image tokens with unimportant content, as shown in Fig. 3. In the self-attention mechanism, tokens are linearly transformed into Query, Key, and Value matrices. The attention score is then computed by performing a dot product between the Query and Key matrices, followed by scaling and Softmax operation. The attention score reflects the importance of each token in relation to the gaze prediction result.

4.3 Performance-aware Training Strategy

As discussed in Section 3.3, most existing gaze-tracking DNNs focus on minimizing the average tracking error. However, this often leads to a higher 95th percentile error in gaze tracking $\Delta\theta$, which increases rendering times in the TFR system. We propose a training strategy to addresses this issue by minimizing the maximum tracking error during training, which can be formulated as:

$$\min \max_{d \in D_{train}} (||\theta_d - \theta_d^g||^2) \tag{1}$$

where θ_d and θ_d^g denote the predicted gaze direction and the ground-truth gaze direction (in radians) for the input sample d in the training dataset D_{train}, respectively. To enhance training stability, the DNN is trained using multiple batches of training samples, resulting in Eq. (2) being:

$$\min \sum_{b \in B} \max_{d \in D_{train}^b} (||\theta_d - \theta_d^g||^2) \tag{2}$$

where B denotes the set of training dataset batches, and D_{train}^b represents the set of training data in batch b. However, using this formula directly as the loss function can result in underutilization of the training dataset, as it tends to focus on only optimizing the sample with the highest tracking error. Empirically, we find it more effective to optimize an approximate version of Eq. (2) by replacing the max operation with an alternative approach, using the approximation $max(x_1, x_2) \approx \frac{1}{N} \ln(e^{Nx_1} + e^{Nx_2})$, namely:

$$\sum_{b \in B} \frac{1}{N} \ln \left(\sum_{d \in D_{train}^b} e^{N||\theta_d - \theta_d^g||^2} \right) \tag{3}$$

where N is the scaling factor that controls the temperature of the approximation. During the training process, the value of N is tuned carefully to adapt to the value distribution of the input training data to ensure the better convergence of the training process.

Finally, we can directly relate the gaze error from Eq. (3) to the TFR latency, enabling us to optimize T_{tf}. To achieve this, we profile rendering latencies across different VR devices and develop a piecewise linear function $U(.)$ that links the gaze tracking error to

© 2025 IEEE. This is the author's version of the article that has been published in the proceedings of IEEE Visualization conference. The final version of this record is available at: xx.xxxx/TVCG.201x.xxxxxxx/

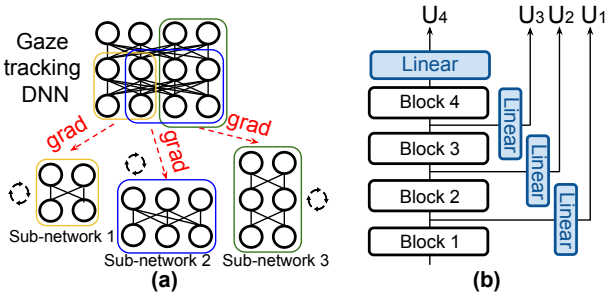

Figure 9: (a) Multi-resolution training framework. (b) An example of four layer ViT with early-exit mechanism, the output from each early-exit is sent to a linear layer to produce the gaze prediction.

Table 1: Accuracy performance and computational cost of different approaches by minimizing average tracking error. P0 and P95 represent 90% and 95% percentiles, respectively.

Network	Mean	P90	P95	Min	Max	FLOPS (billions)
NVGaze [31]	6.81	13.07	18.62	0.94	42.30	0.021
DeepVoG [10]	3.47	17.76	23.77	0.55	29.06	36.5
Seg [11]	3.25	18.29	22.80	0.52	28.42	2.6
ResNet-based [29]	1.52	5.96	13.15	0.07	26.46	3.6
IncResNet-based [28]	1.72	6.23	12.4	0.12	25.47	13.12
FovealNet (0.2)	1.27	4.92	**8.09**	**0**	24.92	2.08
FovealNet (0.1)	1.05	5.75	9.63	**0**	25.54	2.42
FovealNet (0.0)	**0.93**	**4.71**	8.21	**0**	**24.2**	2.80

the corresponding rendering latency, as shown in Fig. 4(a), facilitating the minimization of rendering overhead while preserving visual quality. Thus, the training objective becomes:

$$\sum_{b \in B} U \left(\frac{1}{N} \ln \left(\sum_{d \in D^b_{train}} e^{N||\theta_d - \theta^g_d||^2} \right) \right) \tag{4}$$

4.4 Multi-resolution Training Mechanism

In Sec. 4.3, we depict the design of the loss function (Eq. (4)) by directly optimizing the rendering latency. In practice, the processing latency $T_{tracking}$ of gaze tracking DNN will also contribute to the total latency T_{total}, and the hardware processing latencies for rendering and tracking can vary due to user settings and resource sharing with other applications.

To maintain adaptability and performance, we develop a multi-resolution DNN training approach that optimizes multiple subnetworks across various DNN architectures simultaneously (Fig. 9 (a)). This joint-optimization training framework will produce a multi-resolution model that can execute at varying depths, allowing for the selection of the optimal gaze tracking DNN based on the current system conditions.

To achieve this, we attach a linear layer at the end of each encoder block within the gaze tracking ViT, which will produce a prediction on gaze direction based on the intermediate results from the early-exit points, as shown in Fig. 9 (b). Specifically, let L denote the total number of layer blocks within the ViT, and U_l denote the loss generated from the output of layer $l \in L$. The loss function U_{multi} for the multiresolution training can be formalized as:

$$U_{multi} = \sum_{l} U_l \tag{5}$$

Using early-exit mechanisms, the resulting gaze tracking DNN can operate at different depths to balance gaze tracking latency and accuracy. During execution, the depth can be adjusted adaptively based on the TFR system's condition to optimize the overall per-frame latency T_{total}.

5 Tracking Performance Evaluation

5.1 Settings

We evaluate the tracking performance of FovealNet using the OpenEDS2020 dataset [60], which consists of 128,000 images from 32 participants in the training set and 70,400 images from 8 participants in the validation set. All participants wore a VR-HMD, and the images were captured with a near-eye camera operating at 100Hz and a resolution of 640×400 pixels. The dataset includes ground truth 3D gaze vectors, which we converted into 2D gaze vectors (horizontal and vertical components) [23] to enable a more effective evaluation.

To evaluate FovealNet, we select two model-based methods and three appearance-based methods. The model-based approaches include Seg [11], an efficiency-focused segmentation network, and DeepVoG [10], a popular encoder-decoder network for gaze tracking. The appearance-based approaches are NVGaze [31], a one-shot CNN-based network, and ResNet-based [29] and Inception-ResNet-based models [28]. For the model-based DNNs, we download the pre-trained models from the code repository [68] and run the code using the settings reported in their respective papers. However, since no pre-trained weights are available for the appearance-based methods, we train these methods using the same procedure reported in their work, and report the corresponding results.

In the data pre-processing stage, we first horizontally flip the right eye images to align them with the left eye images, a common practice that allows the application of a single mapping for both eyes [23]. We apply $\beta_1 = 0.2$ and $\beta_2 = 500$ for cropping the input image in **??**. To train the gaze tracking ViT, we use a series of data augmentations techniques to enhance the training convergence speed. These techniques include random cropping, where the image is cropped to a random size between 80% to 100% of its original size, followed by a random shift in position by up to 10% of the image's width and height. Finally, we normalize the images to standardize the input data for the model.

To train the FovealNet, we utilize the Adam optimizer with a learning rate of 5e-4 and a momentum of 0.9, a step learning scheduler that reduces the learning rate by a factor of 0.2 every 10 epochs and a batch size of 512 for 100 epochs. We also implement an early stopping mechanism that halts training if no improvement is observed on the validation set after 10 consecutive epochs. Specifically, we utilize the Adam optimizer with a learning rate of 5e-5 and a momentum of 0.9 and a batch size of 512 for 50 epochs to fine-tune the token pruned FovealNet. All experiments are conducted on a single RTX 4090D GPU.

We evaluate the performance of FovealNet by comparing it with other baseline approaches under two training objectives. First, we train FovealNet using an objective function that aims to minimize the average gaze tracking error, which is the same training objective used by all other baselines. We show that even with this baseline objective, FovealNet still outperforms all other approaches (Sec. 5.2). Next, we train FovealNet using the objective function described in Eq. (4) and compare its performance against other methods in Sec. 5.3. Finally, we present the performance of FovealNet at different resolutions in Sec. 5.5.

5.2 Accuracy Evaluation by Minimizing Average Gaze Error

We train all the models using an objective function designed to minimize the average gaze tracking error, defined as: $L_{mse} = \sum_{b \in B} \sum_{d \in D^b_{train}} (y_d - \hat{y}_d)^2$. B denotes the number of batches within the training data, y_d and $\hat{y}_d$ represent the predicted and ground truth gaze direction for training data d. This is the same training objective used by all other previous works on gaze tracking. For DeepVoG [10] and Seg [11], we use their pretrained model weights and deploy them directly for evaluation. For FovealNet, we evaluate its performance under three settings with tokenwise pruning ratios of

© 2025 IEEE. This is the author's version of the article that has been published in the proceedings of IEEE Visualization conference. The final version of this record is available at: xx.xxxx/TVCG.201x.xxxxxxx/

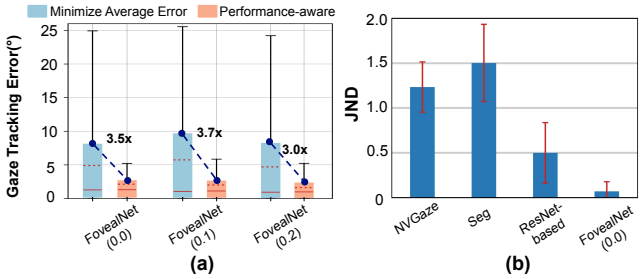

Figure 10: (a) Distribution of gaze tracking error of FovealNet trained with performance-aware loss. The box plot covers 5th to 95th percentile of gaze tracking error, red line denotes average error, red dashed line denotes 90th percentile error. (b) Perceptual Quality Measurement in JND.

0.2, 0.1, and 0.0 (no pruning), labeled as FovealNet (0.2), FovealNet (0.1), and FovealNet (0.0), respectively.

Tab. 1 compares the predicted gaze error of our algorithms against various baselines. To represent the predicted gaze error, we use the eccentricity from true gaze direction, a method that widely utilized in [29, 31]. FovealNet (0.0), without the token pruning mechanism, maintains a minimal mean gaze error of 0.93° across all evaluated models. When different token pruning ratios are applied, the models exhibit gaze errors of 1.05° and 1.27°, respectively, both of which are lower than those of most baseline methods. Additionally, our models show the lowest values for both the 95th-percentile and maximum errors. Specifically, the FovealNet (0.2) achieves the smallest 95th-percentile error of 8.09°.

We also compare the computational complexity in terms of Floating Point Operations (FLOPs), as shown in Tab. 1. FovealNet (0.2) and FovealNet (0.1) achieve lower computational costs compared to most methods, except for NVGaze, which has significantly worse tracking performance. Specifically, the computational cost of FovealNet (0.2) is about 15% of that of the IncResNet-based model [28], 70% lower than the ResNet-based model [29], and 27% lower comparable to Seg [11].

5.3 Accuracy Evaluation by Performance-aware Loss

In this section, we train FovealNet using the performance-aware loss function specified in Eq. (4) with a scaling factor N of 100. Specifically, we use the rendering latency measurements of the Meta Quest Pro on input images with a size of 1080×1920, as shown in Fig. 6(c), to generate the piecewise linear function $U(.)$ and evaluate its impact on the gaze tracking error for FovealNet.

Fig. 10 shows the effectiveness of the performance-aware training strategy from Sec. 4.3. The blue bars indicate the error distribution using the objective function that minimizes average gaze error, as in Tab. 1, while the orange bars represent the gaze error distribution using the performance-aware loss function.

For FovealNet (0.0), it exhibits a notable reduction in the 95th-percentile tracking error, decreasing from 8.21° to 2.31°, with the 90th-percentile error reducing from 4.71° to 1.62° and maximum error diminishing significantly from 24.2° to 5.22°. For FovealNet (0.1) and FovealNet (0.2), the pruned models show only a minor accuracy degradation, with a 90th-percentile error of 2.02° and 2.13° and 95th-percentile error of 2.6° and 2.72°. Overall, by using the performance-aware training strategy, our model achieves an average reduction of over 65% in 95th-percentile and over 70% in maximum errors compared to minimizing average error strategy. Finally, compared to other baselines that aim to minimize the average gaze tracking error shown in Tab. 1, the performance-aware training strategies significantly improve the worst-case gaze error distribution, leading to notable system performance enhancements, as detailed in Sec. 6.

Table 2: Evaluation of Multi-resolution FovealNet.

Model Depth	Gaze Error (°)				FLOPs (Billion)
	Mean	P90	P95	Max	
3	2.93	5.23	7.35	15.70	1.06
4	1.90	3.87	5.23	10.36	1.41
5	1.68	3.43	3.98	8.28	1.76
6	1.30	2.78	3.38	7.78	2.10
7	1.15	2.61	3.05	6.98	2.45
8	1.08	1.95	2.54	5.94	2.80

5.4 Perceptual Quality Measurement

To evaluate the impact of gaze-tracking errors on the perceptual quality of the foveated output, we use the FovVideoVDP metric in Sec. 3.2. Specifically, we sampled 400 random images from the MS COCO test dataset [69], and applied the foveation algorithm [70, 71], configured with a $\theta_i = 5°$ of the eccentricity angle subtended by fovea, when displayed on an HTC Vive Pro HMD (i.e., 13.2° pixels per degree [54]).

For each image, 1080 gaze snapshots were used to simulate foveated rendering corresponding to each snapshot. We measure the similarity between images generated with the predicted gaze direction and those generated with the ground truth gaze direction for each gaze snapshot. We adopt the FovealNet(0.0) discussed in Sec. 5.3 for evaluation. Fig. 10(b) shows the experiment results. Our FovealNet(0.0) achieves a minimal JND of 0.07, meaning users were unable to perceive any noticeable difference from the ground truth. In comparison, the ResNet-based model produces a JND of 0.5, corresponding to a 13% increase in discriminability, while the Seg model results in a JND of 1.5, indicating a 34% likelihood that the rendered image is significantly distinguishable from the ground truth. Statistical analysis reveals a highly significant difference in visual quality between our method and the others, with a p-value of less than 10^{-6}, suggesting the observed differences are extremely unlikely to be due to chance.

5.5 Multi-Resolution Accuracy

While Sec. 5.2 and Sec. 5.3 focus on single-resolution training with the performance-aware training strategy, this section evaluates the performance of FovealNet using the multi-resolution training method from Sec. 4.4. Specifically, six early exits branches are introduced at the end of each transformer block of gaze tracking ViT from block 3 to 8, with a small linear layer for gaze prediction at each exit, resulting in six subnetworks of varying depths. During training, we compute the training losses from the six loss functions and sum them to generate the final multi-resolution loss, as shown in Eq. (5). The rest of the training settings follows Sec. 5.3.

Tab. 2 shows the gaze tracking error and computational cost for each sub-network. FovealNet with a depth of 3 reduces computational complexity by 62.1% compared to the 8-layer subnetwork, while achieving a 95th-percentile error of 7.35° and a maximum error of 15.70°. Notably, the performance 8-layer subnetwork shows a 0.1° increase in mean error, a 0.33° increase in 90th-percentile error and a 0.23° increase in 95th-percentile error compared to the 8-layer FovealNet trained solely with the loss at layer 8. This increase is because of the changes in the training loss function from a single loss (Eq. (4)) to multi-resolution loss (Eq. (5)).

5.6 Ablation Study

In this section, we first examine the effect of the input cropping method, introduced in Sec. 4.1, on the gaze tracking performance and computational cost of FovealNet. Tab. 3 presents the tracking errors and computational cost of FovealNet when using cropped versus uncropped input. For the gaze tracking DNN to process the

© 2025 IEEE. This is the author's version of the article that has been published in the proceedings of IEEE Visualization conference. The final version of this record is available at: xx.xxxx/TVCG.201x.xxxxxxx/

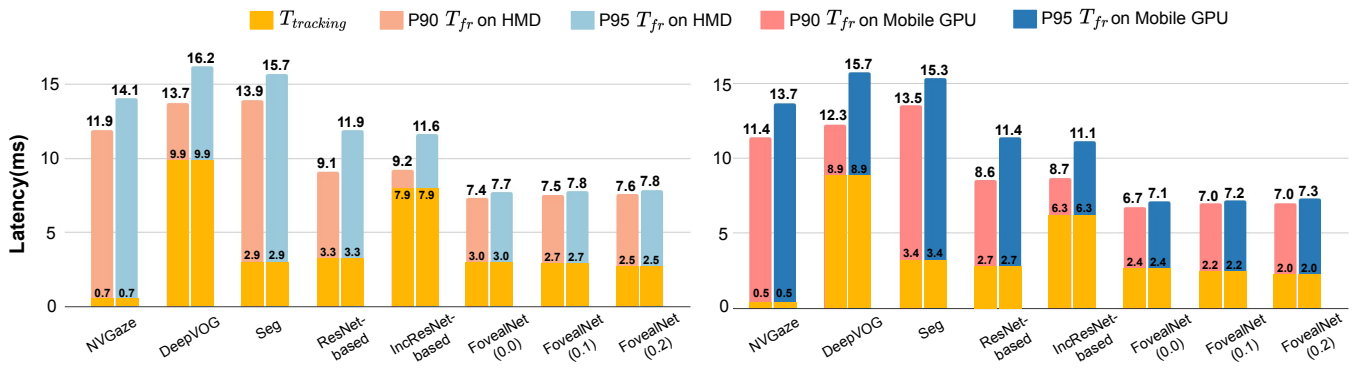

Figure 11: Evaluation on overall processing latency. (a) Measurement on Quest Pro at a resolution of 1080P (1080×1920), including 90th-percentile rendering latency, 95th-percentile rendering latency and tracking latency. (b) Measurement on mobile GPU at a resolution of 1080P (1080×1920), including 90th-percentile rendering latency, 95th-percentile rendering latency and tracking latency.

Table 3: Performance with and without cropping method (in degrees).

Method	Mean error	P90 error	P95 error	Max error	FLOPs (Billions)
with cropping	0.98	1.62	2.31	5.23	2.810
with/o cropping	1.10	1.82	2.65	5.98	2.799

Table 4: Performance with different scaling factor N (in degrees).

N	Mean	P90 error	P95 error	Max error
10	0.97	4.57	6.92	18.98
50	1.02	1.68	2.42	5.37
100	0.98	1.62	2.31	5.23
150	Inf	Inf	Inf	Inf
200	Inf	Inf	Inf	Inf

uncropped input, it is also resized to a square shape of 224×224 pixels. We observe that the involvement of the cropping method results in a mean error decrease from 1.1° to 0.98°, and 90th-percentile and 95th-percentile error drops 0.2° and 0.34° respectively. Meanwhile, the introduction of cropping only increase 11M FLOPs, results in a $< 0.5\%$ FLOPs increase.

In Sec. 5.3, when training with the performance-aware loss, we use a scaling factor of $N = 100$. Here, we explore the impact of different choices of the scaling factor N. As shown in Tab. 4, for a smaller value of $N = 10$, Eq. (3) is unable to effectively minimize the maximum error, leading to a large 95th-percentile error of 6.92°. On the other hand, excessively large values of N, such as 150 or 200, may cause overflow during training, resulting in training failure. Thus, selecting an appropriate N is crucial for ensuring the effectiveness of Eq. (4).

6 TFR System Performance Evaluation

In this section, we evaluate the system performance by measuring processing latency for various methods. First, in Sec. 6.1, we evaluate the system performance of the FovealNet trained with the performance-aware loss defined in Eq. (4). We then demonstrate the system performance of FovealNet under different rendering system conditions by switching the rendering resolution in Sec. 6.2.

6.1 Evaluation with Performance-aware Training Loss

In this section, we compare the system performance across different approaches with the performance-aware training of FovealNet, considering both $T_{tracking}$ and various T_{fr} values. For T_{fr}, we evaluate the foveated rendering configurations that account for the 90th-percentile and 95th-percentile gaze tracking errors to ensure greater versatility. The T_{fr} values are derived by determining latency based on the latency analysis outlined in Sec. 3.3 based on gaze tracking error. We compare the latency performance of various approaches on both HMD (Meta Quest Pro) and a mobile GPU (Quadro RTX 3000). For FovealNet, we train it using the performance-aware loss function described in Eq. (4), where $U(.)$ represents the processing latency under different eccentricity angle θ_f on either the HMD or mobile GPU. To compute the θ_f, we set the eccentricity angle θ_i subtended by the fovea to 5°, and $\Delta\theta$ is set to P95 or P90 of the gaze error distribution on OpenEDS 2020 for different approaches. We adopt the single resolution FovealNet that contains 8 ViT blocks under different pruning ratios.

We profile the processing latency $T_{tracking}$ of FovealNet on both the Meta Quest Pro and a Quadro RTX 3000 Mobile GPU, as discussed in Sec. 3. Since we do not have access to run ViT on the Meta Quest Pro directly, we use GPGPU-sim [72] to simulate the performance of the Adreno 650, which is integrated into the Qualcomm Snapdragon XR2+ and deployed in the Meta Quest Pro. The GPGPU-sim is configured according to the specifications of the Adreno 650 [73].

As shown in Fig. 11, for HMD, FovealNet (0.0) achieves the lowest T_{fr} values of 7.4ms and 7.7ms when setting $\Delta\theta$ to P95 or P90 of the gaze error distribution, representing a reduction of at least 1.7ms and 3.9ms compared to previous methods. Since $T_{sensing}$ and T_{comm} are relatively small and consistent across different methods, as shown in Fig. 2, the per frame latency T_{total} will be mainly determined by $T_{tracking} + T_{fr}$. Our FovealNet (0.0) achieves a $T_{tracking} + T_{fr}$ values of 10.4ms and 10.7ms for P90 and P95, respectively. For the FovealNet with tokenwise pruning, the slight growth in gaze prediction error results in only a minimal increase in T_{fr}, staying below 0.3ms when compared to the FovealNet (0.0). Notably, FovealNet (0.2) achieves the lowest $T_{tracking} + T_{fr}$, with 10.2ms in P90 scenario and 10.4ms in P95 scenario. Similarly in the evaluation results on mobile GPU, FovealNet (0.0) achieves the lowest T_{fr} across different scenarios. This demonstrates the effectiveness of our performance-aware training strategy.

6.2 Latency Evaluation under Varying Conditions

In this section, we evaluate the performance of the multi-resolution FovealNet by using the same model described in Sec. 5.5 and simulating the corresponding execution latencies on the Meta Quest Pro, following the methods outlined in Sec. 6.1. We simulate variations in system configuration by adjusting the rendering resolution to 720P (720 × 1080), 1080P (1080 × 1920), and 1440P (1440 × 2560). For each rendering resolution, we assess the $T_{tracking}$ and T_{fr} of the FovealNet at different depths, as produced by the multi-resolution training mechanism detailed in Sec. 4.4, and search for the optimal FovealNet configurations to minimize the per-frame latency T_{total}. We assume that the TFR system is capable of adapting its optimal rendering settings by adjusting the eccentricity angles θ_f.

© 2025 IEEE. This is the author's version of the article that has been published in the proceedings of IEEE Visualization conference. The final version of this record is available at: xx.xxxx/TVCG.201x.xxxxxxx/

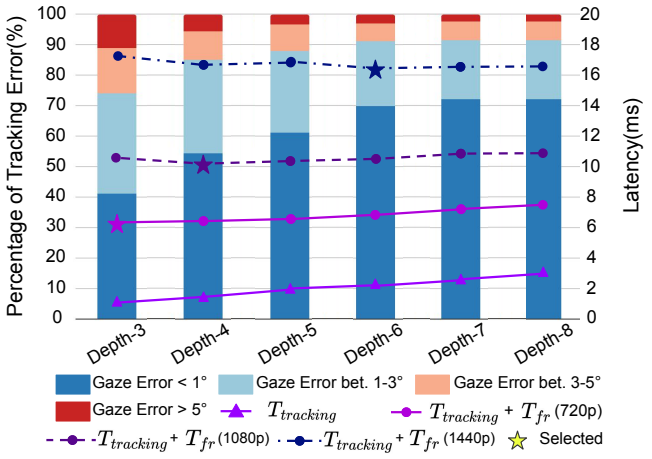

Figure 12: The stacked bars show the distribution of the gaze errors for each subnetwork of FovealNet. The lines show the $T_{tracking}$ under different depth and $T_{tracking} + T_{fr}$ under different rendering resolution. The star marks the optimal selection of the subnetwork that achieves the lowest $T_{tracking} + T_{fr}$ at each rendering resolution.

The results are shown in Fig. 12. The distribution of gaze error and $T_{tracking}$ across the sub-networks remains consistent, with the lowest $T_{tracking}$ at 1.14 ms and the highest at 3 ms. However, in low-resolution 720P scenarios, increasing the depth of the sub-networks offers minimal reduction in T_{fr} while introducing additional $T_{tracking}$. With a sub-network depth of 3, we achieve the optimal configuration, resulting in a $T_{tracking} + T_{fr}$ of 6.19 ms. In contrast, for high-resolution 1440P settings, deepening the sub-networks significantly reduces T_{fr}. The optimal configuration is found at a depth of 6, yielding the lowest latency of 16.4 ms.

7 Conclusion

In this work, we introduce FovealNet, an AI-based gaze tracking solution designed to enhance the performance of TFR systems. FovealNet can be directly optimized using a loss function that incorporates system performance metrics, resulting in superior outcomes compared to baseline algorithms. To further reduce the implementation cost of the gaze tracking algorithm, FovealNet utilizes an event-based cropping technique that discards irrelevant pixels from the input image. Moreover, it features an efficient token-pruning strategy that dynamically eliminates tokens during processing without sacrificing tracking accuracy.

References

[1] Anjul Patney, Joohwan Kim, Marco Salvi, Anton Kaplanyan, Chris Wyman, Nir Benty, Aaron Lefohn, and David Luebke. Perceptually-based foveated virtual reality. In *ACM SIGGRAPH 2016 Emerging Technologies*, SIGGRAPH '16, New York, NY, USA, 2016. Association for Computing Machinery. 1

[2] Rachel Albert, Anjul Patney, David Luebke, and Joohwan Kim. Latency requirements for foveated rendering in virtual reality. *ACM Transactions on Applied Perception (TAP)*, 14(4):1–13, 2017. 1, 2, 3, 4

[3] Linus Franke, Laura Fink, Jana Martschinke, Kai Selgrad, and Marc Stamminger. Time-warped foveated rendering for virtual reality headsets. In *Computer Graphics Forum*, volume 40, pages 110–123. Wiley Online Library, 2021. 1

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

[5] Gazi Karam Illahi, Matti Siekkinen, Teemu Kämäräinen, and Antti Ylä-Jääski. On the interplay of foveated rendering and video encoding. In *Proceedings of the 26th ACM Symposium on Virtual Reality Software and Technology*, pages 1–3, 2020. 1

[6] Gazi Karam Illahi, Matti Siekkinen, Teemu Kämäräinen, and Antti Ylä-Jääski. Foveated streaming of real-time graphics. In *Proceedings of the 12th ACM Multimedia Systems Conference*, pages 214–226, 2021. 1

[7] Mohamed Hegazy, Khaled Diab, Mehdi Saeedi, Boris Ivanovic, Ihab Amer, Yang Liu, Gabor Sines, and Mohamed Hefeeda. Content-aware video encoding for cloud gaming. In *Proceedings of the 10th ACM multimedia systems conference*, pages 60–73, 2019. 1

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

[9] Wenjie Zou, Shixuan Feng, Xionghui Mao, Fuzheng Yang, and Zhibin Ma. Enhancing quality of experience for cloud virtual reality gaming: An object-aware video encoding. In *2021 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, pages 1–6. IEEE, 2021. 1

[10] Yuk-Hoi Yiu, Moustafa Aboulatta, Theresa Raiser, Leoni Ophey, Virginia L. Flanagin, Peter zu Eulenburg, and Seyed-Ahmad Ahmadi. Deepvog: Open-source pupil segmentation and gaze estimation in neuroscience using deep learning. *Journal of Neuroscience Methods*, 324:108307, 2019. 1, 2, 3, 6

[11] Yu Feng, Nathan Goulding-Hotta, Asif Khan, Hans Reyserhove, and Yuhao Zhu. Real-time gaze tracking with event-driven eye segmentation. In *2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 399–408, 2022. 1, 2, 3, 4, 6, 7

[12] Bin Li, Hong Fu, Desheng Wen, and WaiLun LO. Etracker: A mobile gaze-tracking system with near-eye display based on a combined gaze-tracking algorithm. *Sensors*, 18(5), 2018. 1, 2

[13] Anastasios N. Angelopoulos, Julien N.P. Martel, Amit P. Kohli, Jörg Conradt, and Gordon Wetzstein. Event-based near-eye gaze tracking beyond 10,000 hz. *IEEE Transactions on Visualization and Computer Graphics*, 27(5):2577–2586, 2021. 1

[14] Rakshit Kothari, Aayush Kumar Chaudhary, Reynold J. Bailey, Jeff B. Pelz, and Gabriel J. Diaz. Ellseg: An ellipse segmentation framework for robust gaze tracking. *IEEE Transactions on Visualization and Computer Graphics*, 27:2757–2767, 2020. 1

[15] Dan Witzner Hansen and Qiang Ji. In the eye of the beholder: A survey of models for eyes and gaze. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(3):478–500, 2010. 2

[16] Xucong Zhang, Yusuke Sugano, and Andreas Bulling. Evaluation of appearance-based methods and implications for gaze-based applications. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, May 2019. 2

[17] Lech Świrski and Neil A. Dodgson. A fully-automatic, temporal approach to single camera, glint-free 3d eye model fitting [abstract]. In

© 2025 IEEE. This is the author's version of the article that has been published in the proceedings of IEEE Visualization conference. The final version of this record is available at: xx.xxxx/TVCG.201x.xxxxxxx/

Proceedings of ECEM 2013, August 2013. 2, 3

[18] Kang Wang and Qiang Ji. Real time eye gaze tracking with 3d deformable eye-face model. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1003–1011, 2017. 2, 3

[19] Erroll Wood, Tadas Baltrušaitis, Louis-Philippe Morency, Peter Robinson, and Andreas Bulling. A 3d morphable eye region model for gaze estimation. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 297–313, Cham, 2016. Springer International Publishing. 2, 3

[20] Conny Lu, Praneeth Chakravarthula, Kaihao Liu, Xixiang Liu, Siyuan Li, and Henry Fuchs. Neural 3d gaze: 3d pupil localization and gaze tracking based on anatomical eye model and neural refraction correction. In *2022 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 375–383, 2022. 2

[21] Tongyu Zhang, Yiran Shen, Guangrong Zhao, Lin Wang, Xiaoming Chen, Lu Bai, and Yuanfeng Zhou. Swift-eye: Towards anti-blink pupil tracking for precise and robust high-frequency near-eye movement analysis with event cameras. *IEEE Transactions on Visualization and Computer Graphics*, 30(5):2077–2086, 2024. 2

[22] Aayush K. Chaudhary, Rakshit Kothari, Manoj Acharya, Shusil Dangi, Nitinraj Nair, Reynold Bailey, Christopher Kanan, Gabriel Diaz, and Jeff B. Pelz. Ritnet: Real-time semantic segmentation of the eye for gaze tracking. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. IEEE, 2019. 2

[23] Xucong Zhang, Yusuke Sugano, Mario Fritz, and Andreas Bulling. Appearance-based gaze estimation in the wild. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4511–4520, 2015. 2, 6

[24] Xucong Zhang, Yusuke Sugano, Mario Fritz, and Andreas Bulling. Mpiigaze: Real-world dataset and deep appearance-based gaze estimation, 2017. 2

[25] Feng Lu, Takahiro Okabe, Yusuke Sugano, and Yoichi Sato. A head pose-free approach for appearance-based gaze estimation. In *British Machine Vision Conference*, 2011. 2

[26] Yusuke Sugano, Yasuyuki Matsushita, and Yoichi Sato. Learning-by-synthesis for appearance-based 3d gaze estimation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1821–1828, 2014. 2

[27] Erroll Wood, Tadas Baltrušaitis, Louis-Philippe Morency, Peter Robinson, and Andreas Bulling. Learning an appearance-based gaze estimator from one million synthesised images. In *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications*, ETRA '16, page 131–138, 2016. 2

[28] Rishi Athavale, Lakshmi Sritan Motati, and Rohan Kalahasty. One eye is all you need: Lightweight ensembles for gaze estimation with single encoders, 2022. 2, 3, 6, 7

[29] Pier Luigi Mazzeo, Dilan D'Amico, Paolo Spagnolo, and Cosimo Distante. Deep learning based eye gaze estimation and prediction. In *2021 6th International Conference on Smart and Sustainable Technologies (SpliTech)*, pages 1–6, 2021. 2, 3, 4, 6, 7

[30] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 2, 5

[31] Joohwan Kim, Michael Stengel, Alexander Majercik, Shalini De Mello, David Dunn, Samuli Laine, Morgan McGuire, and David Luebke. Nvgaze: An anatomically-informed dataset for low-latency, near-eye gaze estimation. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, page 1–12, New York, NY, USA, 2019. Association for Computing Machinery. 2, 3, 4, 6, 7

[32] Anjul Patney, Marco Salvi, Joohwan Kim, Anton Kaplanyan, Chris Wyman, Nir Benty, David Luebke, and Aaron Lefohn. Towards foveated rendering for gaze-tracked virtual reality. *ACM Trans. Graph.*, 35(6), 2016. 2, 4

[33] Andrew T. Duchowski, Donald H. House, Jordan Gestring, Rui I. Wang, Krzysztof Krejtz, Izabela Krejtz, Radosław Mantiuk, and Bartosz Bazyluk. Reducing visual discomfort of 3d stereoscopic displays with gaze-contingent depth-of-field. In *Proceedings of the ACM Symposium on Applied Perception*, New York, NY, USA, 2014. Association for Computing Machinery. 2

[34] Rados Mantiuk, Bartosz Bazyluk, and Anna Tomaszewska. Gaze-dependent depth-of-field effect rendering in virtual environments. In *Proceedings of the Second International Conference on Serious Games Development and Applications*, Berlin, Heidelberg, 2011. Springer-Verlag. 2

[35] Jiannan Ye, Anqi Xie, Susmija Jabbireddy, Yunchuan Li, Xubo Yang, and Xiaoxu Meng. Rectangular mapping-based foveated rendering. In *2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 756–764, 2022. 2

[36] M. Weier, M. Stengel, T. Roth, P. Didyk, E. Eisemann, M. Eisemann, S. Grogorick, A. Hinkenjann, E. Kruijff, M. Magnor, K. Myszkowski, and P. Slusallek. Perception-driven accelerated rendering. *Computer Graphics Forum*, 36(2):611–643, 2017. 3

[37] NVIDIA Corporation. Turing variable rate shading in vrworks. https://developer.nvidia.com/blog/turing-variable-rate-shading-vrworks/, 2018. 3, 4

[38] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. 3

[39] Aleksei Karpov and Ilya Makarov. Exploring efficiency of vision transformers for self-supervised monocular depth estimation. In *2022 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 711–719, 2022. 3

[40] Zhenglun Kong, Peiyan Dong, Xiaolong Ma, Xin Meng, Mengshu Sun, Wei Niu, Xuan Shen, Geng Yuan, Bin Ren, Minghai Qin, Hao Tang, and Yanzhi Wang. Spvit: Enabling faster vision transformers via soft token pruning, 2022. 3

[41] Tianlong Chen, Yu Cheng, Zhe Gan, Lu Yuan, Lei Zhang, and Zhangyang Wang. Chasing sparsity in vision transformers: An end-to-end exploration, 2021. 3

[42] Yifan Xu, Zhijie Zhang, Mengdan Zhang, Kekai Sheng, Ke Li, Weiming Dong, Liqing Zhang, Changsheng Xu, and Xing Sun. Evo-vit: Slow-fast token evolution for dynamic vision transformer, 2021. 3

[43] Philippe Lancheres and Mohamed Hafed. The mipi c-phy standard: A generalized multiconductor signaling scheme. *IEEE Solid-State Circuits Magazine*, 11(2):69–77, 2019. 3

[44] Haoran You, Yang Zhao, Cheng Wan, Zhongzhi Yu, Yonggan Fu, Jiayi Yuan, Shang Wu, Shunyao Zhang, Yongan Zhang, Chaojian Li, et al. Eyecod: Eye tracking system acceleration via flatcam-based algorithm and hardware co-design. *IEEE Micro*, 43(4):88–97, 2023. 3

[45] Chiao Liu, Lyle Bainbridge, Andrew Berkovich, Song Chen, Wei Gao, Tsung-Hsun Tsai, Kazuya Mori, Rimon Ikeno, Masayuki Uno, Toshiyuki Isozaki, et al. A 4.6 μm, 512$\times$ 512, ultra-low power stacked digital pixel sensor with triple quantization and 127db dynamic range. In *2020 IEEE International Electron Devices Meeting (IEDM)*, pages 16–1. IEEE, 2020. 3

[46] Anastasios N Angelopoulos, Julien NP Martel, Amit PS Kohli, Jorg Conradt, and Gordon Wetzstein. Event based, near eye gaze tracking beyond 10,000 hz. *arXiv preprint arXiv:2004.03577*, 2020. 3

[47] Xiaoyu Sun, Xiaochen Peng, Sai Zhang, Jorge Gomez, Win-San Khwa, Syed Sarwar, Ziyun Li, Weidong Cao, Zhao Wang, Chiao Liu, et al. Estimating power, performance, and area for on-sensor deployment of ar/vr workloads using an analytical framework. *ACM Transactions on Design Automation of Electronic Systems*, 2024. 3

[48] Pil-Ho Lee and Young-Chan Jang. A 6.84 gbps/lane mipi c-phy transceiver bridge chip with level-dependent equalization. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 67(11):2672–2676, 2019. 3

[49] What is Mobile Industry Processor Interface (MIPI) Protocol? 3

[50] Rahul Singh, Muhammad Huzaifa, Jeffrey Liu, Anjul Patney, Hashim Sharif, Yifan Zhao, and Sarita Adve. Power, performance, and image quality tradeoffs in foveated rendering. In *2023 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*, pages 205–214, 2023. 3, 4

[51] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 3

[52] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and

© 2025 IEEE. This is the author's version of the article that has been published in the proceedings of IEEE Visualization conference. The final version of this record is available at: xx.xxxx/TVCG.201x.xxxxxxx/

Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 3

[53] Kenneth Chen, Thomas Wan, Nathan Matsuda, Ajit Ninan, Alexandre Chapiro, and Qi Sun. Pea-pods: Perceptual evaluation of algorithms for power optimization in xr displays. *ACM Transactions on Graphics (TOG)*, 43(4):1–17, 2024. 3

[54] Rafał K Mantiuk, Gyorgy Denes, Alexandre Chapiro, Anton Kaplanyan, Gizem Rufo, Romain Bachy, Trisha Lian, and Anjul Patney. Fovvideovdp: A visible difference predictor for wide field-of-view video. *ACM Transactions on Graphics (TOG)*, 40(4):1–19, 2021. 3, 7

[55] Budmonde Duinkharjav, Praneeth Chakravarthula, Rachel Brown, Anjul Patney, and Qi Sun. Image features influence reaction time: A learned probabilistic perceptual model for saccade latency. *ACM Transactions on Graphics (TOG)*, 41(4):1–15, 2022. 3

[56] Xincheng Huang, James Riddell, and Robert Xiao. Virtual reality telepresence: 360-degree video streaming with edge-compute assisted static foveated compression. *IEEE Transactions on Visualization and Computer Graphics*, 2023. 3

[57] Nianchen Deng, Zhenyi He, Jiannan Ye, Budmonde Duinkharjav, Praneeth Chakravarthula, Xubo Yang, and Qi Sun. Fov-nerf: Foveated neural radiance fields for virtual reality. *IEEE Transactions on Visualization and Computer Graphics*, 28(11):3854–3864, 2022. 3

[58] David Bauer, Qi Wu, and Kwan-Liu Ma. Fovolnet: Fast volume rendering using foveated deep neural networks. *IEEE transactions on visualization and computer graphics*, 29(1):515–525, 2022. 3

[59] Dixuan Cui and Christos Mousas. Estimating the just noticeable difference of tactile feedback in oculus quest 2 controllers. In *2022 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 1–7, 2022. 3

[60] Cristina Palmero, Abhishek Sharma, Karsten Behrendt, Kapil Krishnakumar, Oleg V. Komogortsev, and Sachin S. Talathi. Openeds2020: Open eyes dataset, 2020. 3, 6

[61] Meta Platform Inc. Meta quest pro. https://www.meta.com/quest/quest-pro/, 2022. 4

[62] Quadro rtx. https://www.nvidia.com/en-us/design-visualization/rtx/, 2022. 4

[63] Eagle Dynamics. Digital Combat Simulator. https://www.digitalcombatsimulator.com/en/, 2008. 4

[64] David J Zielinski, Hrishikesh M Rao, Mark A Sommer, and Regis Kopper. Exploring the effects of image persistence in low frame rate virtual environments. In *2015 IEEE Virtual Reality (VR)*, pages 19–26. IEEE, 2015. 4

[65] Apple Inc. Apple Vision Pro, 2024. 4

[66] W. Sprague, Zachary Helft, Jared Parnell, J. Schmoll, G. Love, and Martin Banks. Pupil shape is adaptive for many species. *Journal of Vision*, 13:607–607, 07 2013. 4

[67] Wikipedia contributors. Roundness. https://en.wikipedia.org/wiki/Roundness, 2024. 4

[68] Horizon Research and PyDSGZ. Edgaze: Efficient gaze tracking and deepvog: Deep learning for eye tracking in vr. https://github.com/horizon-research/edgaze and https://github.com/pydsgz/DeepVOG, 2022. Accessed: 2024-09-14. 6

[69] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014. 7

[70] Jeffrey S Perry and Wilson S Geisler. Gaze-contingent real-time simulation of arbitrary visual fields. In *Human vision and electronic imaging VII*, volume 4662, pages 57–69. SPIE, 2002. 7

[71] Ming Jiang, Shengsheng Huang, Juanyong Duan, and Qi Zhao. Salicon: Saliency in context. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1072–1080, 2015. 7

[72] Gpgpu-sim. https://github.com/gpgpu-sim/gpgpu-sim_distribution. 8

[73] Adreno gpu. https://www.notebookcheck.net/Qualcomm-Adreno-650-GPU-Benchmarks-and-Specs.448196.0.html. 8