A3FR: Agile 3D Gaussian Splatting with Incremental Gaze Tracked Foveated Rendering in Virtual Reality

Shuo Xin*
Tandon School of Engineering
New York University
New York, USA

Haiyu Wang
Tandon School of Engineering
New York University
New York, USA

Sai Qian Zhang
Tandon School of Engineering
New York University
New York, USA

Abstract

Virtual reality (VR) significantly transforms immersive digital interfaces, greatly enhancing education, professional practices, and entertainment by increasing user engagement and opening up new possibilities in various industries. Among its numerous applications, image rendering is crucial. Nevertheless, rendering methodologies like 3D Gaussian Splatting impose high computational demands, driven predominantly by user expectations for superior visual quality. This results in notable processing delays for real-time image rendering, which greatly affects the user experience. Additionally, VR devices such as head-mounted displays (HMDs) are intricately linked to human visual behavior, leveraging knowledge from perception and cognition to improve user experience. These insights have spurred the development of foveated rendering, a technique that dynamically adjusts rendering resolution based on the user's gaze direction. The resultant solution, known as gaze-tracked foveated rendering, significantly reduces the computational burden of the rendering process.

Although gaze-tracked foveated rendering can reduce rendering costs, the computational overhead of the gaze tracking process itself can sometimes outweigh the rendering savings, leading to increased processing latency. To address this issue, we propose an efficient rendering framework called *A3FR*, designed to minimize the latency of gaze-tracked foveated rendering via the parallelization of gaze tracking and foveated rendering processes. For the rendering algorithm, we utilize 3D Gaussian Splatting, a state-of-the-art neural rendering technique. Evaluation results demonstrate that A3FR can reduce end-to-end rendering latency by up to 2× while maintaining visual quality.

CCS Concepts

• **Computing methodologies** → **Rendering**; **Tracking**; • **Human-centered computing** → **Virtual reality**.

Keywords

AR/VR, 3D Gaussian Splatting, Gaze tracking, Foveated rendering

*Also with Stanford University, work done during internship at New York University.

ICS '25, June 8–11, 2025, Salt Lake City, UT, USA

ACM ISBN 979-8-4007-1537-2/2025/06
https://doi.org/10.1145/3721145.3735112

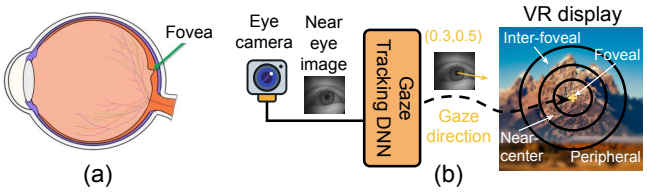

Figure 1: The framework of gaze-tracked foveated rendering.

1 Introduction

Virtual Reality (VR) technologies fundamentally modify interaction modalities with digital systems, integrating physical and virtual domains through immersive frameworks. Across fields like entertainment, gaming, education [3, 64, 67], healthcare [13, 21, 47], and others [14, 27], their significance remains substantial. VR facilitates user engagement with sophisticated simulations and scenarios via comprehensive immersive contexts. VR is reshaping not only content consumption but also how we learn, work, and communicate, fostering innovation and expanding possibilities for the future of human-computer interactions.

Image rendering critically determines AR/VR system efficacy and user satisfaction. Within VR, rendering precision substantially affects perceptual authenticity and immersion quality for users. It is vital to achieve high-resolution and low-latency rendering to ensure a seamless and interactive user experience, especially during movements of the head and body. One such rendering algorithm is 3D Gaussian Splatting (3DGS) [28] , a rasterization technique for real-time radiance field rendering. It enables the real-time rendering of photorealistic scenes learned from small image samples and achieves state-of-the-art performance in visual quality compared to other methodologies.

Although the computational power of VR devices has advanced rapidly over the past decade, most rendering algorithms implemented on VR devices, including 3DGS, are still very time-consuming: a major barrier to achieving the desired low-latency, high-resolution output. These complex algorithms, essential for creating detailed and immersive environments, require significant computational resources and processing time, which can disrupt the user experience by causing delays in visual feedback in response to user actions.

Within VR, the human visual system primarily enables user engagement with virtual environments. Across the visual field, human acuity varies, peaking at the fovea in the retinal center for maximal resolution, as shown in Figure 1 (a). Foveated rendering exploits this acuity gradient, prioritizing computational resources for the central region while reducing allocation to peripheral areas. This technique, termed *Foveated Rendering*, greatly boosts VR system

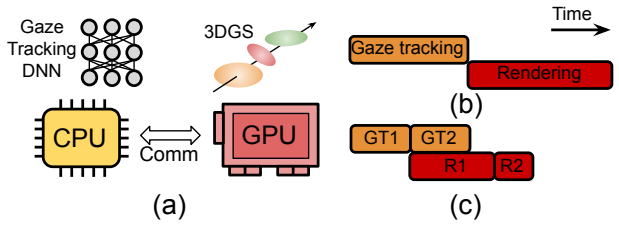

Figure 2: (a) A3FR framework. (b) and (c) depict the workflow of the conventional and A3FR TFR execution, respectively. GT1 and GT2 denote the gaze tracking process, with GT1 representing the completion of generating initial results and GT2 showing the final gaze tracking outcomes. R1 and R2 indicate the incremental rendering of 3DGS.

performance by decreasing the rendering load while maintaining perceived visual quality, establishing it as a vital innovation in VR technology.

To implement foveated rendering, it is essential to track the gaze direction of the users in real time, which then triggers the start of foveated rendering. Gaze-Tracked Foveated Rendering (TFR) enhances VR rendering efficiency through gaze-tracking systems, typically employing deep neural networks (DNNs) for implementation. By precisely determining the user's point of focus in real-time, TFR system can precisely catch the location of the foveal region which is rendered with the highest resolution (Figure 1 (b)), followed by the *near-center region*, *inter-foveal region* and *peripheral region*, which will be rendered from high to low resolution. However, while foveated rendering reduces computational costs, the gaze tracking process adds extra latency that can sometimes exceed the time saved in image rendering.

To reduce the overall processing latency of TFR, this paper introduces *A3FR*, an efficient execution framework for TFR with 3DGS that utilizes parallel processing by executing gaze tracking on the CPU and foveated rendering on the GPU of AR/VR devices, as illustrated in Figure 2 (a). A3FR enables interleaved execution of gaze tracking and rendering tasks, significantly reducing the end-to-end execution time per frame.

To achieve the interleaved operation between gaze tracking and foveated rendering, we have developed a multi-resolution DNN training framework that simultaneously trains the gaze-tracking DNN across different configurations. During the operation, the gaze tracking DNN, implemented on the CPU within the HMD, rapidly produces initial prediction results. These results initially guide the preliminary rendering process of 3DGS on the GPU in the VR system. As the predictions from the gaze tracking DNN become increasingly accurate, the rendering process is incrementally adjusted (Figure 2 (c)), significantly reducing the total execution latency of the conventional TFR, whose processing flow in Figure 2 (b).

To further optimize the TFR cost of 3DGS, we propose *Adaptive Mesh Refinement* (AMR), a technique that reduces computational and memory requirements compared to a uniform resolution while preserving the detailed representation of complex phenomena in advanced simulation and rendering algorithms. Overall, our contribution can be summarized as follows:

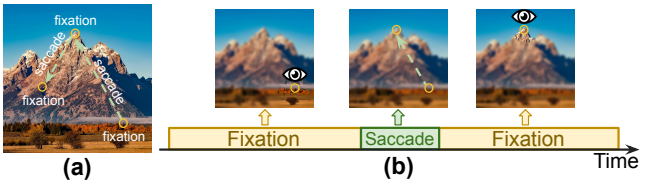

Figure 3: (a) Trace of gaze location. (b) Human eye motion across time.

- We propose a collaborative execution framework that performs foveated rendering and gaze tracking in parallel, reducing the overall processing latency of AR/VR systems. The framework also adapts to different computational resources by balancing the workload between the CPU and GPU.
- We propose an incremental rendering strategy that refines the output over multiple rounds of 3DGS rendering and adaptively adjusts the local resolution based on the visual complexity of the Gaussians.
- To enable incremental gaze prediction, we introduce an efficient gaze-tracking neural network called *A3FR-ViT*, which supports early gaze direction prediction and facilitates parallel processing for 3DGS rendering.
- The evaluation results show that A3FR can reduce the end-to-end rendering latency by over 2× without impacting the user experience, as shown by a comprehensive user study.

2 Background and Related Works

2.1 Oculomotor Behavior in Visual Perception

The human visual system operates through three primary motion modes, each serving distinct roles: *fixation*, which keeps the eye steady on a single point; *saccadic movements*, rapid shifts of gaze between targets; and *smooth pursuit*, which smoothly follows moving objects. Although smooth pursuit is less common, fixation and saccades dominate most visual activities, playing crucial roles in environmental scanning and detail focus, as depicted in Figure 3 (b). During fixation, the gaze centers on a single point with varying acuity across the visual field. The fovea, located centrally on the retina, provides the highest visual resolution due to its dense concentration of photoreceptor cells, enabling detailed and colorful perception within the direct line of sight.

Visual acuity diminishes rapidly outside the fovea, resulting in less sensitivity to fine details in peripheral vision. Humans typically perform one to three rapid saccadic eye movements per second [18, 33, 35], each lasting about 20–100 ms and reaching speeds over 200° per second [57]. These swift movements cause temporary visual blur, known as saccadic blur [11, 49]. Figure 3 (a) illustrates gaze behavior within a scene, showing fixation points connected by saccades before transitioning to another scene. This study focuses on optimizing rendering costs during fixation, which constitutes the majority of viewing time.

2.2 Foveated Rendering

Based on the user's visual behavior, rendering resources can be dynamically allocated. The TFR method employs DNN to process

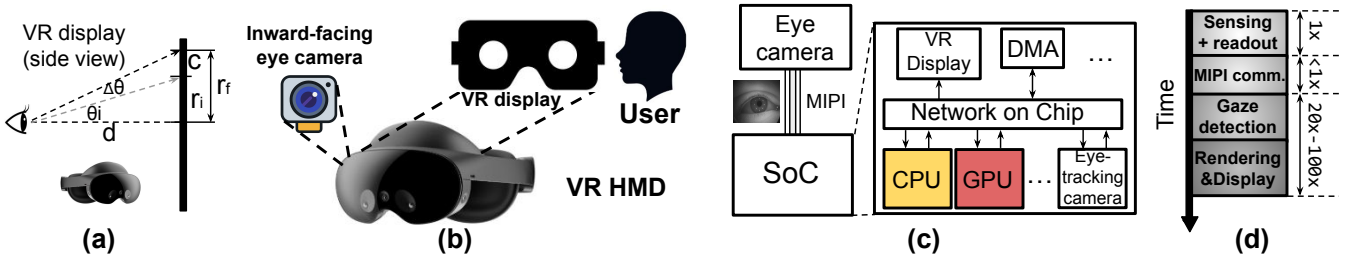

Figure 4: (a) Relationship between tracking error and foveal region size. (b) Meta Quest Pro Head-Mounted Display (HMD). (c) Hardware system layout of VR device. (d) Latency breakdown of TFR for a single frame.

images from the HMD's internal eye-facing cameras, precisely identifying where the user is focusing at any given moment. The system then creates a perceptually-optimized rendering hierarchy with the *foveal region* rendered at maximum resolution, surrounded by the *near-center region*, *inter-foveal region*, and *peripheral region* with progressively reduced detail [48, 55, 71], as illustrated in Figure 1 (b). Figure 4 (a) demonstrates how the high-resolution foveal regions are determined. The radius r_f of this zone is:

$$r_f = r_i + c = \rho d \cdot \tan(\theta_i + \Delta\theta) = \rho d \cdot \tan(\theta_f)^1, \quad (1)$$

where ρ represents the display's pixel density, d is the fixed viewing distance in HMD configurations, θ_i corresponds to the eccentricity angle of high-acuity vision, and $\Delta\theta$ accounts for gaze tracking measurement error. $r_i = \rho d \cdot \tan(\theta_i)$ defines the ideal high-resolution area, while $c = \rho d \cdot [\tan(\theta_i + \Delta\theta) - \tan(\theta_i)]$ provides a safety margin to maintain visual quality despite tracking uncertainties. Research by Lin et al. [40] established an optimal θ_i value of 18° for contemporary VR applications. In most HMD settings, the viewing distance d remains constant due to the fixed position of the display relative to the user's eyes.

According to equation 1. a significant gaze tracking errors would require larger high-resolution rendering zones, potentially undermining the computational benefits of foveated rendering. Processing additional high-resolution pixels substantially increases rendering demands and power consumption. Therefore, precise gaze tracking becomes essential for maximizing system efficiency. Prior studies by Lavalle et al. [37] and Kress et al. [34] demonstrate that maintaining a seamless visual experience requires total system latency between **20-50ms** [4], emphasizing the importance of optimizing both tracking accuracy and rendering performance.

2.3 Neural Rendering

3D reconstruction and novel view generation are essential tasks in computer vision and graphics. Recent years have seen the emergence of neural rendering techniques, which generate high-quality images by inference from a learned neural network. Among these, Neural Radiance Fields (NeRF) [51] has gained significant attention for its ability to generate photorealistic images from 3D scenes. However, NeRF is computationally expensive. Several variants [7–9, 70] have been proposed to reduce rendering latencies while preserving visual quality, but they still face challenges in achieving real-time performance.

Recently, an alternative approach, 3D Gaussian Splatting [28] (3DGS), has been shown to significantly reduces the computational cost of the rendering process. Gaussian Splatting represents scenes using a point cloud, with each point endowed with trainbale position, color, opacity, angular distribution, and radial Gaussian distribution for rasterization. An example is shown in Figure 5. The rendering begins with a model trained offline, representing the scene as a point cloud. Each point corresponds to an ellipsoid, shaped by the scales of 3D Gaussian distributions—referred to as a Gaussian point. These ellipsoids are equipped with trainable parameters that control their scales, positions, orientations, opacity, and color distribution, the latter of which is parameterized using Spherical Harmonics (SH). Once the trained points and ellipsoids are established, the online rendering process consists of three primary steps: *Projection*, *Sorting*, and *Rasterization*. Each step plays a critical role in transforming the 3D model data for final image production on screen.

During the projection stage (Figure 5 (a)), each ellipsoid is initially projected as an ellipse onto the image plane. The process involves determining which ellipses intersect with a given pixel tile (for instance, a 4×4 area as shown by the shaded region in the figure) to accurately contribute to the pixel colors within that tile. After that, for each tile, intersecting ellipses are sorted by their depth relative to the image plane (Figure 5 (b)). This sorting prioritizes ellipses closer to the camera screen, allowing them to have greater influence on the calculation of pixel colors.

Finally, the intersections of all ellipses within a tile and each pixel are calculated. The color of a pixel is then determined using the classical volume rendering method. This method sums the contributions of all intersecting ellipses, processing them from the nearest to the farthest. This technique ensures that the pixel color accurately reflects the visual depth of the scene, as described in Figure 5 (c). Note that, since the images are rendered at the granularity of tiles, it is possible to selectively process only a portion of pixels on each tile. This allows us to make a tradeoff between rendering resolution and GPU workload.

[1]This formula assumes the gaze is centered at the front view, representing the maximum radius of the rendering region.

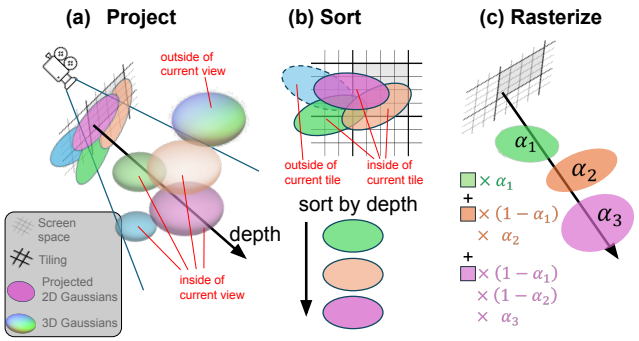

Figure 5: An example of 3DGS Process. (a) Given a camera pose, all 3G Gaussians (colored ellipsoids in figure) that are inside the current camera view are projected to screen space. (b) The projected 2D Gaussians on screen space (colored ellipses in figure) that are inside the current tile are sorted by depth, i.e. distance to camera screen. (c) The opacities α_i of the relevant Gaussians are computed. The Gaussians deposit their color one by one in the previously sorted order.

2.4 Eye-tracking

Eye-tracking methodologies can be classified into two principal categories: appearance-based and model-based techniques [23, 78].

Appearance-based tracking techniques establish direct relationships between eye imagery and gaze directions [23, 42, 65, 79, 80]. These methods necessitate extensive training datasets and have catalyzed innovations across various machine learning frameworks, including linear regression models [46], random forest algorithms [61], k-nearest neighbor approaches [69], and convolutional neural networks [6, 50]. Vision Transformers [15], which have achieved remarkable results across numerous computer vision applications, have also been integrated into gaze tracking systems [30]. However, their substantial computational demands present considerable obstacles for deployment in real-time tracking environments.

In model-based approaches, researchers employ three-dimensional eye models that replicate the eye's biological structure to determine gaze vectors [45, 63, 66, 68]. The implementation typically involves a dual-stage process: initially extracting critical eye features through specialized neural networks and mapping these onto geometric models, followed by calculating the gaze direction from this representation. This methodology effectively transforms gaze tracking into a segmentation task, frequently utilizing convolutional U-Net architectures, which account for the majority of computational resources [19, 39, 73, 76]. While previous research has demonstrated excellent accuracy in eye segmentation [12], the subsequent gaze direction estimations can deviate by more than 2° from actual measurements. These discrepancies primarily stem from imprecise eye center and radius calculations during initialization, alongside limitations inherent to geometric models during optimization procedures, creating misalignment between estimated and actual gaze patterns [63].

	GPU (6 layers)	CPU (3 layers)	CPU (6 layers)
ViT latency	11.43 ms	16.44 ms	30.93 ms
	GPU(720P)	GPU(1080P)	GPU(1440P)
3DGS latency	27.00 ms	37.78 ms	53.14 ms

Table 1: Latency of ViT eye-tracking and 3DGS rendering

3 Methods

3.1 Overview

A typical VR device, such as a HMD, is shown in Figure 4 (b). The inward-facing eye camera of the VR device continuously captures images of the eye to facilitate the execution of TFR. The foveated rendered scene is then displayed on the screen for the user to see. Figure 4 (c) shows the high-level architecture of TFR system, which consists of three primary components: a near-eye camera (image sensor), a host processor, and a interconnection link (MIPI [36]). The standard TFR workflow is depicted in Figure 4 (d). Initially, an eye image is captured by a monochrome camera situated near the eye. This image undergoes preprocessing and readout by the image signal processor (ISP) before it is transmitted to the host processor via the MIPI link. Upon reception, the host processor forwards the image to an eye-tracking DNN, which determines the gaze direction. The gaze direction then informs the foveated rendering process, which adjusts the rendering of the VR scene accordingly.

Figure 4 (d) provides an approximate latency breakdown of the conventional TFR process. Camera sensing latency T_s and MIPI communication delay T_c approximately 1 ms [5, 41, 62, 74, 82] and less than 1 ms [2, 38], respectively, accounting for a small fraction of the total latency. In contrast, the gaze detection latency T_d, along with the subsequent rendering and display process T_r, usually consumes a much larger portion (20-100× longer) of the overall latency, based on the studies from [4, 59]. It is evident from Figure 4 (d) that the majority of the total processing time $T_{tot} = T_s + T_c + T_d + T_r$ is consumed by gaze detection T_d and rendering T_r. Most TFR frameworks in HMDs execute the gaze tracking and foveated rendering process sequentially on the GPU within the VR device [26, 54, 60], which results in the underutilization of other computational resources (e.g., CPU shown in Figure 4 (c)).

To illustrate this, Table 1 presents the latency of two gaze tracking DNNs, as introduced in [43], with three and six layers, respectively. These DNNs are tested on the CPU and GPU of the Nvidia Jetson Orin NX, a commonly used platform to simulate VR devices [22, 53, 75, 81]. Additionally, we simulate the 3DGS rendering process at three resolutions: 720P, 1080P, and 1440P on the GPU of the Jetson Orin NX. We note that running eye tracking concurrently with the 3DGS introduces an additional 12ms of TFR latency. In contrast, offloading the gaze tracking task to the CPU results in a latency similar to that of 3DGS rendering on the GPU.

Therefore, it would be advantageous to parallelize the gaze tracking process with the 3DGS rendering process, utilizing both the CPU and GPU to minimize the total TFR latency, denoted as T_{tot}. Thus, T_{tot} becomes $T_{tot} = T_s + T_c + \max(T_d, T_r)$.

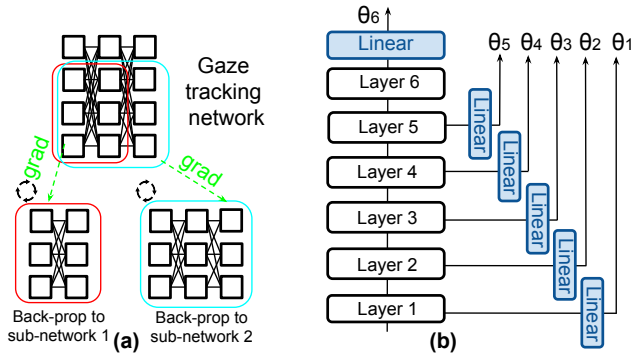

Figure 6: (a) Multi-resolution training framework. (b) An example of N=6 layer ViT with early-exit mechanism, the output from each layer is connected to a linear layer to produce the gaze prediction.

The results in Table 1 indicate that it would be advantageous to parallelize the gaze tracking process with the 3DGS rendering process, utilizing both the CPU and GPU to minimize the total TFR latency, denoted as T_{tot}. Thus, T_{tot} becomes $T_{tot} = T_s + T_c + \max(T_d, T_r)$. To accomplish this, A3FR offloads the less demanding gaze tracking DNN to the CPU, which allows the GPU to dedicate its resources to the more demanding rendering tasks. Furthermore, we introduce A3FR-ViT (Section 3.2) to facilitate an early exit capability for the gaze tracking DNN. This feature produces preliminary gaze tracking results that are immediately sent to the GPU to start the foveated rendering process. As the gaze tracking continues, increasingly accurate results are generated, allowing for ongoing enhancements in the rendering quality. This approach significantly reduces overall latency, as depicted in Figure 2 (c). Next, we describe the design of A3FR-ViT in Section 3.2 and A3FR intremental rendering scheme in Section 3.3.

3.2 Design of Gaze Tracking Neural Network

Vision transformer (ViT) [15] has been shown to perform well on various tasks in computer vision. In this work, we use multilayer ViT [15] for eye-tracking, where the input is the image of the user's eye captured by VR camera and the output is the user's gaze direction. The resultant ViT, termed *A3FR-ViT*, processes the input image by segmenting it into patches, tokenizing the patches, and adding positional data before passing them through the transformer block. The architecture includes 6 transformer blocks, each featuring 6 heads and an embedding dimension of 384. Modifications to the original ViT design replace the classifier MLP layers with a series of linear layers, which output the 2D gaze direction.

To achieve parallel operation between gaze tracking and foveated rendering, we train the ViT so that the parameters in intermediate layers are used to give early predictions, which are used for foveated rendering, before all layers finish. Specifically, we design a multi-resolution DNN training strategy that simultaneously optimizes several sub-networks across distinct DNN architectures (Figure 6 (a)). This joint optimization framework yields a multi-resolution model that operates at varying depths. Initially, preliminary gaze tracking results are generated, which are then utilized by the foveated rendering process. Subsequently, the two processes run in parallel, enhancing the efficiency and responsiveness of the system.

To implement this, we add a linear layer to the end of the selected encoder blocks within the A3FR-ViT, allowing it to generate a gaze direction prediction based on the intermediate outputs, as depicted in Figure 6 (b). Here, let N represent the total number of selected layer blocks in the A3FR-ViT that are appended with local exit. This configuration yields a series of N predictions on the gaze location, u_n, derived from the intermediate results, where $1 \leq n \leq N$. $u_n = \{u_{n,x}, u_{n,y}\}$ further consists of x and y coordinate of the VR display. Consequently, the loss function L_f for the multi-resolution training is defined as the sum of the training losses from each of these early-exit points.

$$L_f = \sum_{n=1}^{N} \lambda_n L_g(u_n, u_{gt}), \tag{2}$$

where u_{gt} denotes the ground truth gaze location in the dataset, and λ_n is a hyperparameter that specifies the importance of gaze predictions at each early-exit point. L_g represents the gaze prediction loss function, with L_2 loss being utilized in this study. The parameter λ_n reflects the relative weight assigned to each loss function. u_N denotes the final gaze prediction.

To further minimize the computational demands of A3FR-ViT, we implement tokenwise pruning [32] on the input tokens by evaluating their significance (attention scores) in relation to the final gaze prediction and discarding the less important tokens. In the self-attention mechanism of the model, tokens undergo a linear transformation into Query, Key, and Value matrices. The attention score is calculated through a dot product between the Query and Key matrices, which is then scaled and processed through a Softmax operation. This score reflects each token's relevance to the gaze prediction outcome. Using these scores, tokens with an attention score below a specified threshold, σ, are filtered out. This pruning effectively reduces the computational load on subsequent ViT blocks by lowering the number of input tokens and, consequently, shrinking the size of the intermediate activations.

3.3 A3FR Incremental Rendering Scheme

As discussed in Section 3.2, the ViT eye-tracking system provides a series of gaze predictions u_i at each exit point. These predictions become more accurate progressively. The A3FR system begins foveated rendering using these early predictions even before the final gaze location is confirmed, refining the rendering focus as more precise predictions become available. This strategy leverages early predictions effectively, reducing unnecessary processing and wait times. In terms of system operations, the A3FR-ViT and 3DGS rendering processes run simultaneously on the CPU and GPU within the HMD, enhancing hardware utilization and reducing total computational delay for TFR. Prior to execution, the CPU and GPU latencies on gaze tracking neural network execution and foveated rendering are first profiled offline (Figure 7 (a)), and a subset of M exit points ($M < N$) is selected from the N available local exit points

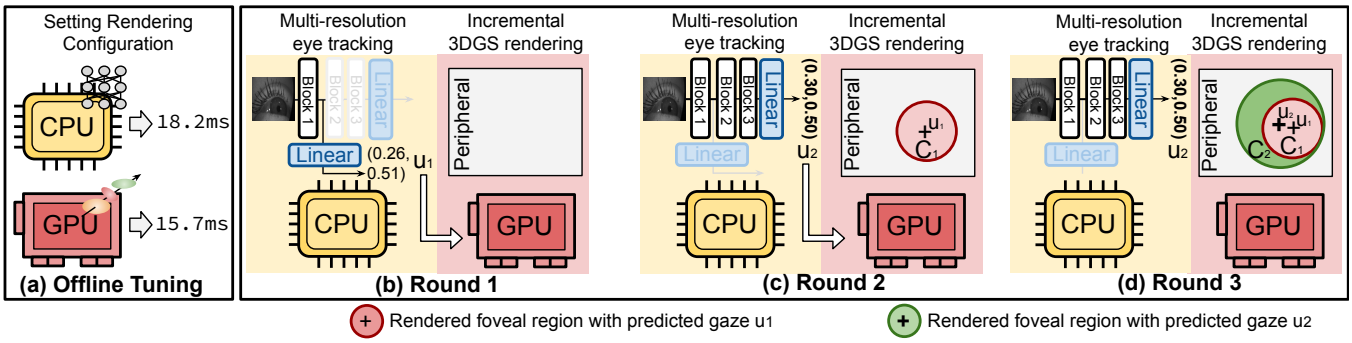

Figure 7: Illustration of the proposed parallel TFR procedure for a simple eye-tracking model only 1 early exit. In round 1, the 3DGS model starts precomputing and rendering at peripheral resolution. In round 2, 3DGS model receives an intermediate gaze prediction and refines pixels the foveal regions. In round 3, final gaze prediction is passed and 3DGS model makes corrections using the more accurate gaze.

to ensure optimal synchronization between the CPU and GPU. Figure 7 illustrates an example where $M = 2$. While A3FR-ViT works on its initial layers, 3DGS concurrently renders the peripheral areas which do not require precise gaze information as illustrated in Figure 7 (b). When the first gaze prediction u_1 is made, it is sent to the GPU to adjust the rendering to higher resolutions as needed, shown in Figure 7 (c). With each subsequent, more accurate prediction u_1, u_2, etc., 3DGS adjusts its rendering area to align with the new gaze data, as depicted in Figure 7 (d).

3.3.1 Incremental Rendering Strategy. Due to potential errors in initial gaze predictions, the areas rendered in high resolution based on early results (e.g., u_1) might not correspond to the actual regions requiring high-resolution detail. For simplicity, we consider the scenario where the scene is rendered at two levels of resolution—one for the foveal area and another for the peripheral area. However, the proposed incremental rendering strategy can be adapted to accommodate multiple resolution levels. An example is illustrated in Figure 8 (a): Suppose the preliminary gaze prediction u_1 is initially sent to the GPU, triggering the rendering process for the region C_1 with a radius of $r_{f,1}$ at the highest resolution. If u_1 is significantly far from the final predicted gaze location u_N, this discrepancy results in substantial redundant rendering since C_1 does not align with the actual foveal region C_N, determined by u_N with a radius $r_{f,N}$. Conversely, when u_1 is sufficiently close to u_N, the overlap between C_1 and C_N increases, and the wasted rendering region gets smaller, as depicted in Figure 8 (b).

Our aim is to **minimize unnecessary rendering computations** by ensuring that the areas rendered based on initial gaze predictions (e.g., C_1) are effectively used in subsequent refinement processes. This approach optimizes the use of computational resources by leveraging the initial rendering output in later, more detailed rendering stages. Specifically, to eliminate the unnecessary rendering in C_1, we must ensure that the sum of the radius of C_1 and the distance between u_1 and the final gaze prediction u_N is less than the foveal region radius $r_{f,N}$ denoted by C_N (Figure 8(c)). This condition guarantees that C_1 is entirely contained within the foveal region C_N. Mathematically, this relationship is formalized in the following theorem:

$$r_{f,i} \leq max(0, r_{f,N} - dist(u_i, u_N)) \quad 1 \leq i \leq N \tag{3}$$

where i is set to 1 shown in Figure 8 (c) and $dist(.)$ denotes the l_2 distance between two points. To ensure efficient utilization of computational resources, the maximum radius of the rendered region should be determined by Equation 4. Once the next predicted gaze location $u_i (1 \leq i \leq N)$, is generated and is closer to u_N, the corresponding radius $r_{f,i}$ can be repeatedly computed using Equation 4, ensuring that only the incremental region is rendered, as shown in green in Figure 8 (d).

3.3.2 Offline Profiling. To optimize rendering settings, equation 4 is based on the known distances $\text{dist}(u_i, u_N)$ between the predicted gaze locations u_i and u_N. In practical applications, we utilize the average distance between gaze locations as determined from eye-tracking training data (e.g. OpenEDS 2020 [52]). Consequently, Equation 4 can be reformulated to incorporate this average distance, providing a more empirical basis for adjusting rendering parameters to enhance accuracy and efficiency in real-time applications.

$$r_{f,i} = max(0, r_{f,N} - \mathbb{E}(dist(u_i, u_N))) \quad 1 \leq i \leq N \tag{4}$$

where $\mathbb{E}(.)$ denotes the expected value across the training data set.

3.3.3 Incremental Rendering under Dynamic Performance Variation. To minimize overall latency, it is crucial that the CPU completes the gaze prediction at the i-th local exit L_i before the rendering of C_{i-1} finishes. If not, C_i will have to wait until u_i is fully processed, resulting in increased overall latency. Additionally, the processing speeds of the GPU and CPU can fluctuate over time due to resource sharing with other programs, making it challenging to develop a fixed scheduling algorithm for minimal processing time.

To address this issue, we've developed an asynchronous incremental rendering scheme that allows the GPU and CPU to operate independently. Under this scheme, the latest predicted gaze u_i is stored in GPU memory, and the incremental rendering process runs concurrently on the GPU. Once the GPU completes rendering the current region $C_j, (j < i)$, it retrieves the most recent gaze prediction from its memory and initiates the next round of rendering based on this updated gaze information, as depicted in Figure 7.

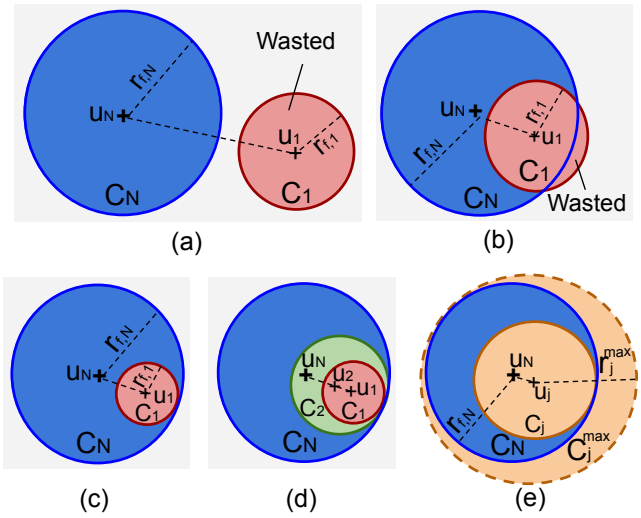

Figure 8: An illustration of the A3FR operation is provided: (a) If the distance between u_1 and u_N is large, the entire region centered at C_1 will be wasted. (b) As u_1 and u_N get closer, the extent of the wasted region decreases. (c) No region will be wasted if the radius $r_{f,1}$ meets the requirements set forth in equation 4. (d) The green region will be rendered as u_2 approaches u_N. (e) If no additional gaze predictions are forthcoming, the rendering process will persist until the rendering of C_j^{max} is complete, indicated by the dotted yellow circle.

However, one issue that arises is when the CPU processing speed significantly lags behind that of the GPU, potentially delaying the availability of the next gaze prediction u_i when C_{i-1} finishes rendering. This situation can prevent the GPU from proceeding with the next round of the rendering process. To address this, we have developed a **speculative incremental rendering** (SIR) method, which allows the rendering process to continue even if no new gaze prediction is available. In this scenario, the incremental rendering process will continue, centered on u_j, until a more accurate gaze prediction is received. If no additional gaze predictions are received, the rendering process will conclude once the region C_j^{max}, defined by the maximum radius r_j^{max}, has been fully rendered. This can be illustrated as follows:

$$r_j^{max} = r_{f,N} + \mathbb{E}[dist(u_j, u_N)] \tag{5}$$

An example is shown in Figure 8 (e). Assume that C_j has been rendered and no further gaze predictions are received. In this case, the rendering process will continue until the region delineated by a dotted circle, with a radius of r_j^{max}, is fully rendered. This ensures coverage of the actual foveal region centered at u_N.

3.4 AMR-based Rendering Strategy of A3FR

Adaptive mesh refinement (AMR) is a technique widely used in scientific computing and finite-element methods [10, 44, 77]. It adaptively reduces/enhances the local tile resolution based on local error estimation in a scientific simulation or graphic rendering. AMR saves the computation workload and memory usage, compared to a uniform resolution, while maintaining the details of various phenomena in intricate simulation and rendering algorithms. Moreover, it allows one to reuse the results from lower quality levels when computing higher levels of refinement. In this paper, we further apply this idea to the practice of 3DGS.

Building on the incremental rendering scheme outlined in Section 3.3.1, we now explore adaptive adjustments to the rendering resolution for 3DGS to achieve computational savings. Specifically, for each block of adjacent 2×2 pixels, depending on the relative distance between the pixel tile and the predicted gaze direction, a subset of the pixels within the current tile will be rendered accordingly. As illustrated in Figure 9, for each block of adjacent 2×2 pixels in the peripheral region, only the top left pixel is rendered. Similarly, in the inter-foveal, near-center, and foveal regions, the number of rendered pixels within the 2×2 tile increases accordingly. The un-rendered pixels simply inherit or interpolate from the rendered pixels. This provides a knob of 4 levels of resolutions, ideally corresponding to 4 regions of eccentricity from gaze center. Our implementation can also be extended to any N^2 levels, by selectively rendering pixels on $N \times N$ block. The user study described in Section 4.7 shows that this adaptive rendering strategy will not cause appreciable visual quality degradation, while greatly saves the computational cost.

3.5 Implementation Details

In practice, to implement incremental rendering, we made several technical changes to the original 3DGS algorithm [29]. First, before rendering, the 3DGS algorithm precomputes several intermediate quantities: 3D Gaussian points projected to screen space, subsets of Gaussian points on each tile, sorted list of Gaussians by depth and etc. To achieve rendering by multiple rounds without redundant precomputing, the intermediate variables computed in round 1 are buffered, and later rounds directly use the buffered variables without the need to access the original 3D Gaussian representations of the scene. Also, to make the best use of the *cooperative group* feature in CUDA, the original 3DGS model uses a tiling of 16×16 pixels per tile, i.e. 256 threads per cooperative group at render time. To achieve adaptive, incremental rendering, we make the following changes: we choose a tiling of 32×32 pixels per tile. The tile is further subsampled to 16×16 blocks of 2×2 pixels. At render time, the tile will raise 4 cooperative groups, corresponding to label $1 \sim 4$ in Figure 9, and each group consists of 256 threads rendering the 16×16 blocks. The tiles are labeled with accuracy levels $1 \sim 4$ according to visual eccentricity from the gaze center. If, for instance, a tile is labeled as level 2, then only 2 out of the 4 groups of 256 threads will run through the 3DGS calculation, while the other 2 will exit immediately after checking, thus saving the computing resources considerably.

3.6 Summary

The A3FR framework seamlessly integrates the techniques discussed in the previous sections. We employ the A3FR-ViT with an early-exit mechanism for eye-tracking, which generates a sequence of increasingly accurate gaze predictions at multiple exit points. The GPU implements 3DGS to perform foveated rendering based on these gaze predictions in an incremental fashion,

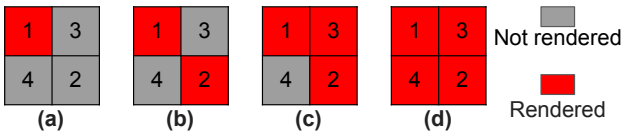

Figure 9: The adaptive rendering strategy of A3FR involves varying the rendering detail based on proximity to the gaze location. For pixel tiles in the peripheral region, only the top left corner is rendered. As the tiles get closer to the gaze location, more pixels within each tile are rendered.

with rendered regions dynamically adjusted as predictions improve. The rendering process is adaptive, meaning the resolution of tiles varies based on the number of gaussians intersecting with each tile. Our AMR strategy enables progressive refinement of the output image over multiple rendering rounds. The foveated rendering aspect is achieved by adjusting rendering resolution according to the eccentricity from the user's gaze point. The incremental rendering approach effectively balances the rendering and eye-tracking workloads between CPU and GPU resources. The complete A3FR framework is outlined in Algorithm 1.

Algorithm 1 A3FR Framework

Input: $V(.)$ ▷ A3FR-ViT function, return series V_l at layer l
Input: $G(.)$ ▷ A3FR-3DGS function, return rasterized pixels
Input: E ▷ User's eye image captured
Input: N ▷ Max layers of ViT
$r \leftarrow$ Profiling(V(dataset)) ▷ Eq. 4: foveal regions
$S \leftarrow 0, (0,0)$ ▷ Init shared variable: layer index, gaze center
Spawn Process 1 and Process 2 in Parallel
Process 1: ▷ Gaze tracking
for $l = 1$ **to** N **do**
  $(x, y) \leftarrow V_l(E)$ ▷ Run ViT forward by 1 layer
  $S \leftarrow l, (x, y)$ ▷ Record gaze tracking result
end for
End Process 1
Process 2: ▷ Rendering
$I \leftarrow 0$ ▷ Init Canvas
$l, (x, y) \leftarrow S$ ▷ Query gaze tracking result
while $l \leq N$ **do**
  $M \leftarrow$ AdaptiveMesh(l, (x, y), r) ▷ AMR
  $I \leftarrow G(M)$ ▷ 3DGS rasterization using mesh M
  $l, (x, y) \leftarrow S$ ▷ Update gaze tracking result
end while
End Process 2

4 Experiments

4.1 Settings

Training. We use the original 3D Gaussian Splatting [28] as our baseline for full-resolution rendering. Following the established methodology, we use L-1 loss and D-SSIM regularization, and incorporate both pruning and densification strategies during the optimization phase as specified in the original work [28]. For our experiments, all scenes undergo consistent training for 30,000 iterations to ensure fair comparison across different rendering approaches.

The A3FR-ViT architecture consists of $N = 6$ transformer layers, with a local exit implemented at the end of each layer. The baseline ViT model maintains the same $N = 6$ layer structure with early exits at each layer ($n = 1$). For training, we compute loss at each exit point using a batch-based maximum loss approach. More specifically, in Equation 2, for training dataset D partitioned into batches B, the loss L_g is calculated as $\sum_{b \in B} \max_{d \in D_b}(||\theta_d - \theta_{gt}||^2)$, where θ_d represents the predicted gaze direction and θ_{gt} represents the ground truth.

To evaluate A3FR-ViT, we compared it with ResNet-34 [6] and DeepVOG [73]. For ResNet-34, we added early exit points after each of its four residual blocks by connecting intermediate features to linear layers for gaze prediction. With DeepVOG, we maintained its encoder-decoder structure and positioned exit points after the encoding stream and after each upsampling layer in the decoding stream. All models were trained using the same loss function to ensure fair comparison.

Datasets. For 3DGS training, we adopt four scenes from Tanks& Temples [31] and Deep Blending [25] datasets. All the eye-tracking models are trained and evaluated on the OpenEDS datasets [17, 20], which were captured by VR device cameras in real time. OpenEDS data are eye images of 640×400 resolution and are grouped into 9160 sequences of continuous eye movement, with each sequence containing 100 images. All of the gaze tracking networks are trained with 25 epochs.

Hardware. The experiments were conducted on a machine with 4-core 8-thread Intel Xeon Platinum 8259CL CPU and an Nvidia Tesla T4 16GB GPU (1590 MHz clock, 2560 CUDA cores). The experiments were conducted under two conditions: the default settings and a modified setting with a 915 MHz clock frequency and 1024 CUDA cores. This adjustment was made to mimic the performance of an edge device GPU, specifically the Jetson Orin NX 16GB, which operates at a 918 MHz clock frequency and possesses 1024 CUDA cores [1], and has been frequently used in prior research to model rendering performance in VR devices [22, 24, 53, 58, 75, 81]. Similarly, two settings are applied for CPU clock frequency, 2.5 GHz and 2.2 GHz, where 2.5 GHz is the default CPU frequency, and 2.2 GHz is used to simulate the clock frequency for Jetson Orin NX 16GB.

Baseline comparison. The default A3FR framework loads the 3DGS rendering on GPU and the eye-tracking network on CPU in parallel with communication. In experiments we compare A3FR against two baseline rendering approaches.

- The original 3DGS rendering on GPU without the using the gaze tracking mechanism, denoted as *Full Resolution Rendering (FRR).*
- The traditional TFR framework where the gaze tracking process and the foveated rendering process are executed sequentially, denoted as *Sequential Foveated 3DGS Rendering (SFR).*

For all the algorithms, the tiling size in the 3DGS process is set at 32×32. Evaluations are conducted at three different resolutions: 1280×720, 1920×1080, and 2560×1440 (720p, 1080p, and 1440p).

4.2 Performance on Gaze Tracking

In this section, we assess the performance of gaze tracking by comparing A3FR-ViT with two other gaze tracking neural networks. For A3FR-ViT, we enhance efficiency by applying tokenwise pruning, which eliminates redundant tokens that have low attention scores, thereby reducing the computational cost of gaze tracking. Specifically, we implement two pruning ratios, 10% and 20%, to evaluate the impact of this reduction on the performance of A3FR-ViT.

Table 2 presents a summary of gaze tracking errors on the test set of the OpenEDS dataset, represented by σ_x and σ_y in degrees. It also details the processing latency on the Jetson Orin NX 16GB CPU. For the A3FR-ViT model, the embedding layer consists of a single convolutional layer that transforms the input image into a sequence of tokens. In contrast, for the ResNet-based models, the embedding layer comprises a convolutional layer, a batch normalization layer, and a ReLU layer. For A3FR-ViT, ResNet-based models, and DeepVoG, local exits are introduced to generate the gaze location at the end of each block. Specifically, six local exits are used for A3FR-ViT and DeepVoG, while the ResNet-based models utilize four local exits.

From Table 2, several key observations emerge. First, the gaze tracking error decreases for local exits attached to the deeper stages of the neural network, as more layers contribute to processing the gaze data, resulting in more accurate predictions. Second, A3FR-ViT demonstrates significantly lower gaze tracking latency on the VR CPU compared to the other two baselines, while maintaining comparable accuracy. This improvement is attributed to the implementation of tokenwise pruning and the inherent accuracy advantages of Vision Transformers (ViT) over traditional CNNs. Finally, as the pruning ratio increases, the gaze tracking latency for A3FR-ViT significantly decreases without a substantial loss in accuracy. For instance, with 20% of tokens pruned, A3FR-ViT achieves an end-to-end latency of 26.28ms. Increasing the pruning ratio to 20% further reduces the latency to 21.64ms, demonstrating the effectiveness of pruning in enhancing processing efficiency while retaining performance.

During our offline profiling as Described in Sec. 3.3, both the average tracking error and its distribution are needed. We used the error tested on OpenEDS dataset as the underlying probability distribution for profiling. As an example, Figure 10 shows the 2D probability distribution, marginalized distribution and Gaussian fit of gaze tracking error for A3FR-ViT at layer-3 and layer-6.

The lower two rows for each model in Table 2 report the by-layer and cumulative latencies. We find that pruning gives moderate speedup while slightly influencing tracking accuracy. The ResNet-based model and DeepVOG are more expensive computationally, giving significant delays of latency.

4.3 A3FR Latency Evaluation

In this section, we assess the latency performance of A3FR by comparing it with two baseline methods, FRR and SFR, as described in Section 4.1. Specifically, we evaluate all three solutions on four scenes—"truck," "train," "drjohnson," and "playroom"—sourced from the Tanks & Temples [31] and Deep Blending [25] datasets. The evaluation is conducted at three different resolutions: 1280×720, 1920×1080, and 2560×1440 for each scene, with 100 camera poses per resolution. The average rendering latency across these 100 camera poses is recorded. The experiments are performed using the hardware setup detailed in Section 4.1.

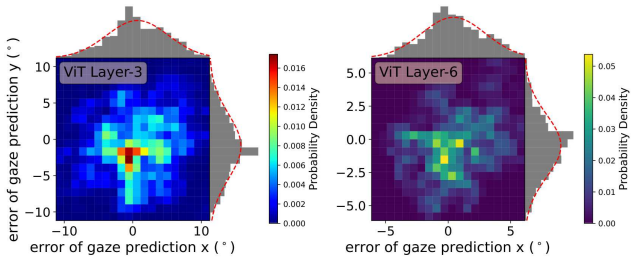

Figure 10: Left: Joint and marginalized distribution of gaze tracking error using the early exit results at 3rd layer of A3FR-ViT. Right: The error distribution of the final gaze prediction at 6th layer.

The results are presented in Figure 11, where Figure 11 (a) illustrates the performance under standard CPU and GPU configurations, while Figure 11 (b) shows the results on the Jetson Orin NX. Overall, A3FR consistently outperforms the baseline methods across all tested scenes, resolutions, and hardware setups. At 1080p and 1440p resolutions, A3FR achieves an average speedup of 20% compared to SFR and 40% compared to FRR. For 720p, the performance gain of SFR over FRR is relatively smaller due to eye-tracking overhead, whereas A3FR still delivers an average speedup of 30%. The results remain consistent across different scenes. This is because A3FR parallelizes eye tracking with the 3DGS process by distributing them between the CPU and GPU, significantly reducing overall computational latency. Additionally, we evaluated rendering performance on the "playroom" scene at 720P resolution under reduced settings equivalent to 16 TOPS. The results demonstrate a of latency of 151.239 ms, 104.328 ms and 76.757 ms for FRR, SFR and A3FR, respectively.

To analyze the details of latency improvement, we present a breakdown of latency for a sample input frame (Figure 12 (a)). Specifically, we select the "truck" scene at 1080p resolution with 100 camera poses. Let R_i denote the rendering process of C_i, as illustrated in Figure 7, and let L_i represent the computational latency on the CPU at the i-th exit point. It is important to note that the maximum index $i_{\max}$ for L_i and R_i may differ due to variations in processing speed, causing either gaze tracking or 3DGS rendering to complete fewer rounds. We observe that the preprocessing stage of the 3DGS rendering pipeline requires a considerable amount of computation, which can be initiated before the gaze tracking process begins. Within the A3FR framework, the CPU and GPU synchronize to enable parallel processing, effectively reducing overall rendering latency.

4.4 Impact of Gaze-tracking Network

In this section, we analyze the impact of A3FR-ViT on overall rendering latency. Specifically, we replace A3FR-ViT with the ResNet-based gaze tracking model from Table 2 and repeat the latency measurement on the "truck" scene at 1080p resolution over 100

Table 2: Accuracy Evaluation on Gaze Tracking Performance

		Embedding	Layer-1	Layer-2	Layer-3	Layer-4	Layer-5	Layer-6
A3FR-ViT	Error $(\sigma_x, \sigma_y)/°$	–	(8.40, 9.33)	(6.08, 7.05)	(4.50, 4.46)	(3.58, 3.38)	(2.97, 2.85)	(2.05, 2.16)
	Latency/ms	1.00	4.10	4.34	4.46	4.40	3.90	4.06
	Cumulative Latency	1.00	5.11	9.45	13.91	18.31	22.21	26.28
A3FR-ViT (0.1 pruned)	Error $(\sigma_x, \sigma_y)/°$	–	(8.19, 9.22)	(6.10, 6.98)	(4.40, 4.53)	(3.55, 3.76)	(3.01, 2.96)	(2.53, 2.56)
	Latency/ms	0.93	3.73	4.21	3.98	3.76	3.91	3.70
	Cumulative Latency	0.93	4.66	8.88	12.85	16.62	20.52	24.23
A3FR-ViT (0.2 pruned)	Error $(\sigma_x, \sigma_y)/°$	–	(8.17, 9.62)	(6.07, 6.85)	(4.61, 4.42)	(3.74, 3.55)	(3.16, 2.88)	(2.69, 2.39)
	Latency/ms	1.07	3.62	4.56	3.40	3.06	2.99	2.92
	Cumulative Latency	1.07	4.70	9.26	12.66	15.72	18.72	21.64
ResNet-based	Error $(\sigma_x, \sigma_y)/°$	–	(7.96, 10.04)	(5.45, 5.70)	(2.71, 2.38)	(2.24, 1.94)	–	–
	Latency/ms	3.71	6.48	6.86	11.63	8.61	–	–
	Cumulative Latency	3.71	10.19	17.05	28.69	37.30	–	–
DeepVOG	Error $(\sigma_x, \sigma_y)/°$	–	(6.27, 8.24)	(4.34, 5.13)	(3.19, 3.53)	(2.63, 2.88)	(2.24, 2.42)	(1.91, 2.03)
	Latency/ms	–	10.06	6.78	24.77	28.55	35.31	9.13
	Cumulative Latency	–	10.06	16.84	41.61	70.16	105.47	114.6

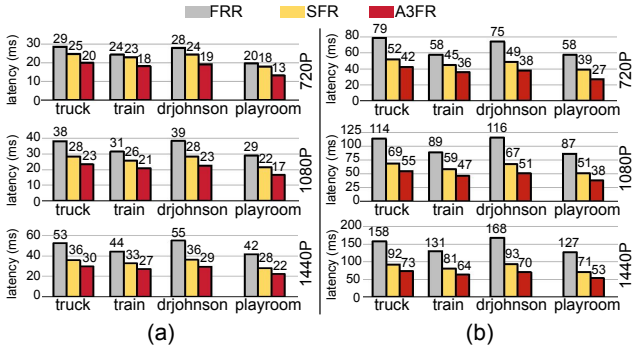

Figure 11: (a) Comparison of rendering latency for A3FR against other baseline models across 4 scenes and 3 resolutions. Gray columns represent FRR with the original 3DGS model. Yellow columns represent SFR where eye-tracking ViT and 3DGS are running on the same GPU in serial. Red columns represent A3FR. Latencies are shown in (ms). (b) Latency evaluation encompasses three approaches, with the GPU clock frequency and CUDA cores adjusted to align with the Jetson NX Orin 16GB.

camera poses. The breakdown is presented in Figure 12 (b). Compared to A3FR-ViT, the ResNet-based model significantly increases the overall TFR processing time. This is primarily because L3 is considerably more time-consuming (~11ms) than a ViT layer (~4ms), while early predictions from L1 and L2 exhibit high error rates, making the intermediate layers of ResNet-based models tend to be either computationally expensive or less accurate.

4.5 Effect of Speculative Incremental Rendering

In this section, we examine the impact of A3FR latency caused by variations in the relative processing speeds of the CPU and GPU. To simulate performance fluctuations, we use *systemd* to consume CPU computational resources, thereby slowing down the processing speed of A3FR-ViT during the TFR process. The CPU utilization rate is alternated between 50% and 100% at 0.1-second intervals. All evaluations are conducted on the "truck" scene at 1080p resolution and repeated over 100 camera poses. We compare the A3FR against a baseline algorithm that did not contain speculative incremental rendering. Namely, if the current rendering round is finished and new gaze tracking results has been received by GPU, it will wait until receive it before rendering further.

The results presented in Figure 12 (c) demonstrate that the A3FR framework remains resilient to performance fluctuations. As shown in Figure 12 (a), a reduced processing speed in A3FR-ViT leads to slower gaze tracking output. However, compared to A3FR without speculative incremental rendering, it still achieves a relatively low overall TFR latency of 26.0ms. In contrast, without speculative incremental rendering, the next rendering round would only begin upon receiving the subsequent gaze tracking result, significantly increasing the overall rendering latency.

4.6 Impact of AMR

Finally, we evaluate the impact of AMR on reducing TFR rendering latency. To achieve this, we remove AMR from the A3FR framework and repeat the latency evaluation across four scenes and three different resolutions, using the same experimental settings as described in Section 4.3. The results are presented in Figure 13. We observe that AMR contributes to approximately a 10% improvement in rendering latency at 1440p resolution, while its impact is less pronounced at lower resolutions. This is primarily due to the additional computational overhead introduced by AMR. Specifically, during preprocessing, tiles must be sorted based on the number of Gaussians, and their resolutions adjusted accordingly. When the rendering load is relatively low, such as in the 720p case, this additional processing cost becomes comparable to the potential savings, reducing the overall benefit of AMR.

4.7 User Study

To evaluate the practical effectiveness of our A3FR approach, we conducted a comprehensive user study to assess the visibility of

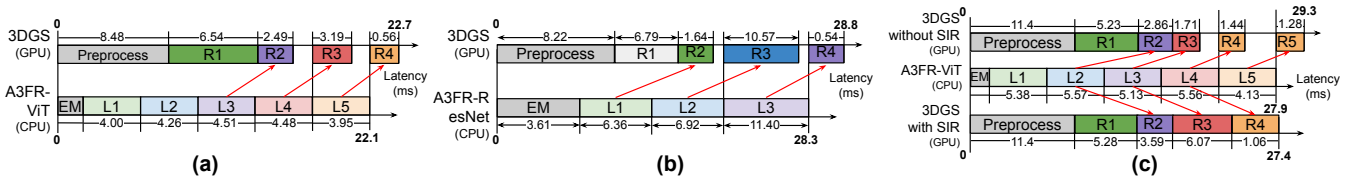

Figure 12: (a) Latency breakdown of A3FR framework. For 3DGS, the gray bar represents the preprocessing latency, while the colored bars labeled R1 to R4 indicate the latencies for each rendering round. Both individual and cumulative latencies are displayed. For ViT, the gray bar represents the patch embedding latency, and the colored bars labeled L1 to L5 correspond to the processing latency at each local exit point. Red arrows illustrate the dependency of each 3DGS rendering round on the results from the corresponding local exit of A3FR-ViT. (b) Latency breakdown with ResNet-based Model as the gaze tracking network. (c) Impact of speculative incremental rendering on the overall latency.

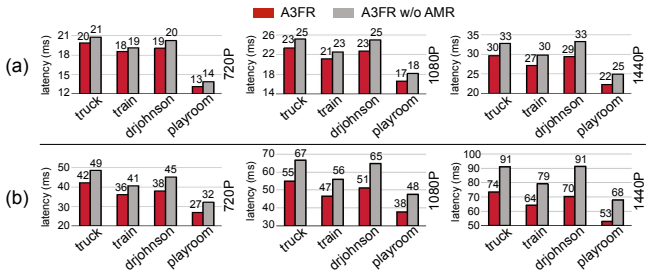

Figure 13: (a) Ablation study of AMR strategies. A3FR rendering latencies are shown in red and those without AMR are shown in grey. (b) Same as (a), but run on the machine with reduced GPU clock frequency and CUDA cores to match Jetson NX Orin 16GB.

Figure 14: Sample images from 20 views of 4 scenes.

artifacts in foveated 3D Gaussian Splatting rendering. The primary objective of the study was to quantitatively assess the rendering quality achieved by our A3FR framework (Sections 3.3 and 3.4) and to validate its potential to deliver high-quality visual experiences under gaze-tracked conditions. This objective was met by demonstrating that participants could not reliably distinguish between the A3FR and full-resolution rendering methods, confirming that the overall perceptual quality remains uncompromised.

A total of eight participants were recruited for the experiment, ensuring a representative sample for evaluating the method. As shown in Figure 15, during the experiment, each participants remained seated and observed the stimuli via a HMD, the Meta Quest Pro [56]. Interaction during the study was facilitated through a standard keyboard interface, providing a uniform and straightforward means of navigating between stimuli. The stimuli comprised images rendered from various scenes using 3DGS. Participants were instructed to perform a two-interval forced-choice (2IFC) task [72], a widely used methodology for assessing visual preferences. In each trial, participants were presented with rendered images applied to the same scene: (1) the reference image, from the ground truth dataset of 3DGS, and (2) two test images, labeled t1 and t2, rendered using different methods. One of the test images is generated using our A3FR approach with AMR, which incorporates a pre-defined fixed point to simulate the gaze direction. The other test image depicts a fully rendered scene using 3DGS without gaze-tracking. These two rendering methods, referred to as m1 (A3FR) and m2 (FRR), provides a basis for evaluating the impact of TFR on perceived visual quality.

During each trial, t1 and t2, which depicted the same visual scene, were randomly paired with m1 and m2 to eliminate order effects or potential bias. Participants were allowed to freely switch among t1, t2, and the reference image using the keyboard, enabling them to carefully assess and compare the visual quality of the test images.

To simulate realistic gaze-tracking conditions, participants were instructed to focus on the marked pre-defined fixed point (as shown in Figure 14) throughout the comparison and were required to make their judgments within a 10-second time frame. As noted in Section 2.1, the typical duration of an eye fixation ranges from tens of milliseconds to several seconds [16]. To accommodate the brief delay caused by switching between image pairs and the subsequent visual recovery period, we extended the fixation duration to 10 seconds as the trial time. These constraints were designed to ensure that the decision-making process closely mirrored the time-sensitive dynamics of real-time gaze-tracked rendering systems.

After observing both test images at least once, participants were asked to select the test image they perceived as having higher visual quality. A total of 20 distinct image pairs (sample images shown in Figure 14), rendered from 4 different scenes, were presented to each participant, resulting in 20 trials per participant. This design provided a robust dataset for analyzing user preferences and visual performance across various scenarios. The results of the user

Figure 15: Participants are conducting the user study on the Quest Pro.

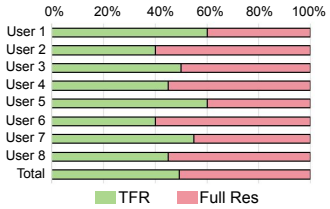

Figure 16: Selection results from the participants.

study are illustrated in Figure 16. Across all participants, the TFR within our A3FR framework was selected 49.4%±8.2% of the time compared to the baseline rendering method, demonstrating that our rendering method maintains visual quality and user experience without significant degradation compared to the baseline method, effectively avoiding perceivable decline in image quality or overall visual satisfaction. This outcome underscores the practicality of the A3FR framework for real-world applications in foveated rendering systems.

5 Conclusions

In this work, we propose a collaborative execution framework, A3FR, that performs foveated rendering and gaze tracking in parallel to reduce the overall processing latency in AR/VR systems. We further introduce A3FR-ViT, an efficient gaze-tracking neural network that enables early estimation of gaze direction and facilitates parallel processing with 3DGS rendering. A3FR significantly reduces rendering latency while maintaining equal visual quality validated by our user study. Experimental results demonstrate its performance across various software and hardware settings. This framework paves the way for more efficient and responsive rendering systems in VR applications.

References

[1] [n. d.]. NVIDIA Jetson Orin. https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/.

[2] [n. d.]. What is Mobile Industry Processor Interface (MIPI) Protocol? https://www.synopsys.com/blogs/chip-design/what-is-mobile-industry-processor-interface-protocol.html

[3] Abdullah M Al-Ansi, Mohammed Jaboob, Askar Garad, and Ahmed Al-Ansi. 2023. Analyzing augmented reality (AR) and virtual reality (VR) recent development in education. *Social Sciences & Humanities Open* 8, 1 (2023), 100532.

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

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

[6] Rishi Athavale, Lakshmi Sritan Motati, and Rohan Kalahasty. 2022. One Eye is All You Need: Lightweight Ensembles for Gaze Estimation with Single Encoders. arXiv:2211.11936 [cs.CV] https://arxiv.org/abs/2211.11936

[7] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. 2021. Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields. In *ICCV*. 5835–5844.

[8] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. 2022. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. In *CVPR*. 5460–5469.

[9] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. 2023. Zip-NeRF: Anti-Aliased Grid-Based Neural Radiance Fields. In *ICCV*. 19697–19705.

[10] Marsha J. Berger and Joseph Oliger. 1984. Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations. *J. Comput. Phys.* 53 (1984), 484. https://doi.org/10.1016/0021-9991(84)90073-1

[11] Fergus W Campbell and Robert H Wurtz. 1978. Saccadic omission: why we do not see a grey-out during a saccadic eye movement. *Vision research* 18, 10 (1978), 1297–1303.

[12] Aayush K. Chaudhary, Rakshit Kothari, Manoj Acharya, Shusil Dangi, Nitinraj Nair, Reynold Bailey, Christopher Kanan, Gabriel Diaz, and Jeff B. Pelz. 2019. RITnet: Real-time Semantic Segmentation of the Eye for Gaze Tracking. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. IEEE. https://doi.org/10.1109/iccvw.2019.00568

[13] Rajeswari Chengoden, Nancy Victor, Thien Huynh-The, Gokul Yenduri, Rutvij H Jhaveri, Mamoun Alazab, Sweta Bhattacharya, Pawan Hegde, Praveen Kumar Reddy Maddikunta, and Thippa Reddy Gadekallu. 2023. Metaverse for healthcare: a survey on potential applications, challenges and future directions. *IEEE Access* 11 (2023), 12765–12795.

[14] Woranipit Chidsin, Yanlei Gu, and Igor Goncharenko. 2021. AR-based navigation using RGB-D camera and hybrid map. *Sustainability* 13, 10 (2021), 5585.

[15] Alexey Dosovitskiy. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

[16] Andrew T Duchowski and Andrew T Duchowski. 2017. *Eye tracking methodology: Theory and practice*. Springer.

[17] Kara J Emery, Marina Zannoli, James Warren, Lei Xiao, and Sachin S Talathi. 2021. OpenNEEDS: A dataset of gaze, head, hand, and scene signals during exploration in open-ended VR environments. In *ACM Symposium on Eye Tracking Research and Applications*. 1–7.

[18] Jasper H Fabius, Alessio Fracasso, Tanja CW Nijboer, and Stefan Van der Stigchel. 2019. Time course of spatiotopic updating across saccades. *Proceedings of the National Academy of Sciences* 116, 6 (2019), 2027–2032.

[19] Yu Feng, Nathan Goulding-Hotta, Asif Khan, Hans Reyserhove, and Yuhao Zhu. 2022. Real-Time Gaze Tracking with Event-Driven Eye Segmentation. In *2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. 399–408. https://doi.org/10.1109/VR51125.2022.00059

[20] Stephan J Garbin, Yiru Shen, Immo Schuetz, Robert Cavin, Gregory Hughes, and Sachin S Talathi. 2019. Openeds: Open eye dataset. *arXiv preprint arXiv:1905.03702* (2019).

[21] Jaris Gerup, Camilla B Soerensen, and Peter Dieckmann. 2020. Augmented reality and mixed reality for healthcare education beyond surgery: an integrative review. *International journal of medical education* 11 (2020), 1.

[22] Antonin Gilles, Pierre Le Gargasson, Grégory Hocquet, and Patrick Gioia. 2023. Holographic near-eye display with real-time embedded rendering. In *SIGGRAPH Asia 2023 Conference Papers*. 1–10.

[23] Dan Witzner Hansen and Qiang Ji. 2010. In the Eye of the Beholder: A Survey of Models for Eyes and Gaze. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32, 3 (2010), 478–500. https://doi.org/10.1109/TPAMI.2009.30

[24] Tairan He, Zhengyi Luo, Xialin He, Wenli Xiao, Chong Zhang, Weinan Zhang, Kris Kitani, Changliu Liu, and Guanya Shi. 2024. OmniH2O: Universal and Dexterous Human-to-Humanoid Whole-Body Teleoperation and Learning. *arXiv preprint arXiv:2406.08858* (2024).

[25] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. 2018. Deep blending for free-viewpoint image-based rendering. *ACM Trans. Graph.* 37, 6, Article 257 (Dec. 2018), 15 pages. https://doi.org/10.1145/3272127.3275084

[26] Meta Platform Inc. 2022. Gaze-tracked Foveated Rendering in Meta Quest Pro. https://developers.meta.com/horizon/blog/save-gpu-with-eye-tracked-foveated-rendering/.

[27] Ye-Joon Jo, Jun-Seok Choi, Jin Kim, Hyo-Joon Kim, and Seong-Yong Moon. 2021. Virtual reality (VR) simulation and augmented reality (AR) navigation in orthognathic surgery: a case report. *Applied Sciences* 11, 12 (2021), 5673.

[28] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.* 42, 4 (2023), 139–1.

[29] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM TOG* 42, 4 (7 2023).

[30] Joohwan Kim, Michael Stengel, Alexander Majercik, Shalini De Mello, David Dunn, Samuli Laine, Morgan McGuire, and David Luebke. 2019. 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)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300780

[31] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. 2017. Tanks and temples: benchmarking large-scale scene reconstruction. *ACM Trans. Graph.* 36, 4, Article 78 (July 2017), 13 pages. https://doi.org/10.1145/3072959.3073599

[32] 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. 2022. SPViT: Enabling Faster Vision Transformers via Soft Token Pruning. arXiv:2112.13890 [cs.CV] https://arxiv.org/abs/2112.13890

[33] Eileen Kowler. 2011. Eye movements: The past 25 years. *Vision research* 51, 13 (2011), 1457–1483.

[34] Bernard C Kress. 2020. Optical architectures for augmented-, virtual-, and mixed-reality headsets. *(No Title)* (2020).

[35] Yuna Kwak, Eric Penner, Xuan Wang, Mohammad R Saeedpour-Parizi, Olivier Mercier, Xiuyun Wu, Scott Murdison, and Phillip Guan. 2024. Saccade-Contingent Rendering. In *ACM SIGGRAPH 2024 Conference Papers*. 1–9.

[36] Philippe Lancheres and Mohamed Hafed. 2019. The MIPI C-PHY standard: A generalized multiconductor signaling scheme. *IEEE Solid-State Circuits Magazine* 11, 2 (2019), 69–77.

[37] Steven M LaValle, Anna Yershova, Max Katsev, and Michael Antonov. 2014. Head tracking for the Oculus Rift. In *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 187–194.

[38] Pil-Ho Lee and Young-Chan Jang. 2019. 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 (2019), 2672–2676.

[39] Bin Li, Hong Fu, Desheng Wen, and WaiLun LO. 2018. Etracker: A Mobile Gaze-Tracking System with Near-Eye Display Based on a Combined Gaze-Tracking Algorithm. *Sensors* 18, 5 (2018). https://doi.org/10.3390/s18051626

[40] Weikai Lin, Yu Feng, and Yuhao Zhu. 2024. MetaSapiens: Real-Time Neural Rendering with Efficiency-Aware Pruning and Accelerated Foveated Rendering. https://doi.org/10.1145/3669940.3707227 arXiv:2407.00435 [cs.GR]

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

[42] Wenxuan Liu, Budmonde Duinkharjav, Qi Sun, and Sai Qian Zhang. 2025. Fovealnet: Advancing ai-driven gaze tracking solutions for efficient foveated rendering in virtual reality. *IEEE Transactions on Visualization and Computer Graphics* (2025).

[43] Wenxuan Liu, Budmonde Duinkharjav, Qi Sun, and Sai Qian Zhang. 2025. FovealNet: Advancing AI-Driven Gaze Tracking Solutions for Optimized Foveated Rendering System Performance in Virtual Reality. *IEEE Transactions on Visualization and Computer Graphics (IEEE VR)* (2025).

[44] Frank Löffler, Joshua Faber, Eloisa Bentivegna, Tanja Bode, Peter Diener, Roland Haas, Ian Hinder, Bruno C Mundim, Christian D Ott, Erik Schnetter, et al. 2012. The Einstein Toolkit: a community computational infrastructure for relativistic astrophysics. *Classical and Quantum Gravity* 29, 11 (2012), 115001.

[45] Conny Lu, Praneeth Chakravarthula, Kaihao Liu, Xixiang Liu, Siyuan Li, and Henry Fuchs. 2022. 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)*. 375–383. https://doi.org/10.1109/ISMAR55827.2022.00053

[46] Feng Lu, Takahiro Okabe, Yusuke Sugano, and Yoichi Sato. 2011. A Head Pose-free Approach for Appearance-based Gaze Estimation. In *British Machine Vision Conference*. https://api.semanticscholar.org/CorpusID:7733236

[47] Henna Mäkinen, Elina Haavisto, Sara Havola, and Jaana-Maija Koivisto. 2022. User experiences of virtual reality technologies for healthcare in learning: an integrative review. *Behaviour & Information Technology* 41, 1 (2022), 1–17.

[48] Rados Mantiuk, Bartosz Bazyluk, and Anna Tomaszewska. 2011. Gaze-Dependent depth-of-field effect rendering in virtual environments. In *Proceedings of the Second International Conference on Serious Games Development and Applications*. Springer-Verlag, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-23834-5_1

[49] Ethel Matin. 1974. Saccadic suppression: a review and an analysis. *Psychological bulletin* 81, 12 (1974), 899.

[50] Pier Luigi Mazzeo, Dilan D'Amico, Paolo Spagnolo, and Cosimo Distante. 2021. Deep Learning based Eye gaze estimation and prediction. In *2021 6th International Conference on Smart and Sustainable Technologies (SpliTech)*. 1–6. https://doi.org/10.23919/SpliTech52315.2021.9566413

[51] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2020. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*. 405–421.

[52] Cristina Palmero, Abhishek Sharma, Karsten Behrendt, Kapil Krishnakumar, Oleg V. Komogortsev, and Sachin S. Talathi. 2020. OpenEDS2020: Open Eyes Dataset. arXiv:2005.03876 [cs.CV] https://arxiv.org/abs/2005.03876

[53] Junrui Pan and Timothy G Rogers. [n. d.]. CRISP: Concurrent Rendering and Compute Simulation Platform for GPUs. ([n. d.]).

[54] Anjul Patney, Joohwan Kim, Marco Salvi, Anton Kaplanyan, Chris Wyman, Nir Benty, Aaron Lefohn, and David Luebke. 2016. Perceptually-based foveated virtual reality. In *ACM SIGGRAPH 2016 Emerging Technologies* (Anaheim, California) *(SIGGRAPH '16)*. Association for Computing Machinery, New York, NY, USA, Article 17, 2 pages. https://doi.org/10.1145/2929464.2929472

[55] Anjul Patney, Marco Salvi, Joohwan Kim, Anton Kaplanyan, Chris Wyman, Nir Benty, David Luebke, and Aaron Lefohn. 2016. Towards foveated rendering for gaze-tracked virtual reality. *ACM Trans. Graph.* 35, 6 (2016). https://doi.org/10.1145/2980179.2980246

[56] Meta Quest Pro. 2022. https://www.meta.com/quest/quest-pro/tech-specs/#tech-specs.

[57] DA Robinson. 1964. The mechanics of human saccadic eye movement. *The Journal of physiology* 174, 2 (1964), 245.

[58] Mohammadreza Saed, Yuan Hsi Chou, Lufei Liu, Tyler Nowicki, and Tor M. Aamodt. 2022. Vulkan-Sim: A GPU Architecture Simulator for Ray Tracing. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 263–281. https://doi.org/10.1109/MICRO56248.2022.00027

[59] Rahul Singh, Muhammad Huzaifa, Jeffrey Liu, Anjul Patney, Hashim Sharif, Yifan Zhao, and Sarita Adve. 2023. Power, Performance, and Image Quality Tradeoffs in Foveated Rendering. In *2023 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*. 205–214. https://doi.org/10.1109/VR55154.2023.00036

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

[61] Yusuke Sugano, Yasuyuki Matsushita, and Yoichi Sato. 2014. Learning-by-Synthesis for Appearance-Based 3D Gaze Estimation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 1821–1828. https://doi.org/10.1109/CVPR.2014.235

[62] Xiaoyu Sun, Xiaochen Peng, Sai Zhang, Jorge Gomez, Win-San Khwa, Syed Sarwar, Ziyun Li, Weidong Cao, Zhao Wang, Chiao Liu, et al. 2024. 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).

[63] Lech Świrski and Neil A. Dodgson. 2013. A fully-automatic, temporal approach to single camera, glint-free 3D eye model fitting [Abstract]. In *Proceedings of ECEM 2013* (Lund, Sweden). http://www.cl.cam.ac.uk/research/rainbow/projects/eyemodelfit/

[64] Khaled Takrouri, Edward Causton, and Benjamin Simpson. 2022. AR technologies in engineering education: Applications, potential, and limitations. *Digital* 2, 2 (2022), 171–190.

[65] Haiyu Wang, Wenxuan Liu, and Sai Qian Zhang. [n. d.]. Hardware and Algorithm Codesign for Efficient Gaze Tracking in Virtual Reality System. ([n. d.]).

[66] Kang Wang and Qiang Ji. 2017. Real Time Eye Gaze Tracking with 3D Deformable Eye-Face Model. In *2017 IEEE International Conference on Computer Vision (ICCV)*. 1003–1011. https://doi.org/10.1109/ICCV.2017.114

[67] Thomas Westin, José Neves, Peter Mozelius, Carla Sousa, and Lara Mantovan. 2022. Inclusive AR-games for education of deaf children: Challenges and opportunities. In *European Conference on Games Based Learning*, Vol. 16. 597–604.

[68] Erroll Wood, Tadas Baltrušaitis, Louis-Philippe Morency, Peter Robinson, and Andreas Bulling. 2016. A 3D Morphable Eye Region Model for Gaze Estimation. In *Computer Vision – ECCV 2016*, Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling (Eds.). Springer International Publishing, Cham, 297–313.

[69] Erroll Wood, Tadas Baltrušaitis, Louis-Philippe Morency, Peter Robinson, and Andreas Bulling. 2016. 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)*. 131–138. https://doi.org/10.

1145/2857491.2857492

[70] Zhiwen Yan, Chen Li, and Gim Hee Lee. 2023. NeRF-DS: Neural Radiance Fields for Dynamic Specular Objects. In *CVPR*. 8285–8295.

[71] Jiannan Ye, Anqi Xie, Susmija Jabbireddy, Yunchuan Li, Xubo Yang, and Xiaoxu Meng. 2022. Rectangular Mapping-based Foveated Rendering. In *2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. 756–764. https://doi.org/10.1109/VR51125.2022.00097

[72] Yaffa Yeshurun, Marisa Carrasco, and Laurence T Maloney. 2008. Bias and sensitivity in two-interval forced choice procedures: Tests of the difference model. *Vision research* 48, 17 (2008), 1837–1851.

[73] Yuk-Hoi Yiu, Moustafa Aboulatta, Theresa Raiser, Leoni Ophey, Virginia L. Flanagin, Peter zu Eulenburg, and Seyed-Ahmad Ahmadi. 2019. DeepVOG: Open-source pupil segmentation and gaze estimation in neuroscience using deep learning. *Journal of Neuroscience Methods* 324 (2019), 108307. https://doi.org/10.1016/j.jneumeth.2019.05.016

[74] Haoran You, Yang Zhao, Cheng Wan, Zhongzhi Yu, Yonggan Fu, Jiayi Yuan, Shang Wu, Shunyao Zhang, Yongan Zhang, Chaojian Li, et al. 2023. EyeCoD: Eye Tracking System Acceleration via FlatCam-Based Algorithm and Hardware Co-Design. *IEEE Micro* 43, 4 (2023), 88–97.

[75] Baoheng Zhang, Yizhao Gao, Jingyuan Li, and Hayden Kwok-Hay So. 2024. Co-designing a Sub-millisecond Latency Event-based Eye Tracking System with Submanifold Sparse CNN. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5771–5779.

[76] Tongyu Zhang, Yiran Shen, Guangrong Zhao, Lin Wang, Xiaoming Chen, Lu Bai, and Yuanfeng Zhou. 2024. 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 (2024), 2077–2086. https://doi.org/10.1109/TVCG.2024.3372039

[77] Weiqun Zhang, Andrew Myers, Kevin Gott, Ann Almgren, and John Bell. 2021. AMReX: Block-structured adaptive mesh refinement for multiphysics applications. *The International Journal of High Performance Computing Applications* 35, 6 (2021), 508–526.

[78] Xucong Zhang, Yusuke Sugano, and Andreas Bulling. 2019. 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. https://doi.org/10.1145/3290605.3300646

[79] Xucong Zhang, Yusuke Sugano, Mario Fritz, and Andreas Bulling. 2015. Appearance-based gaze estimation in the wild. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 4511–4520. https://doi.org/10.1109/CVPR.2015.7299081

[80] Xucong Zhang, Yusuke Sugano, Mario Fritz, and Andreas Bulling. 2017. MPIIGaze: Real-World Dataset and Deep Appearance-Based Gaze Estimation. arXiv:1711.09017 [cs.CV] https://arxiv.org/abs/1711.09017

[81] Ziliang Zhang, Zexin Li, Hyoseung Kim, and Cong Liu. 2024. BOXR: Body and head motion Optimization framework for eXtended Reality. *arXiv preprint arXiv:2410.13084* (2024).

[82] Yiwei Zhao, Ziyun Li, Win-San Khwa, Xiaoyu Sun, Sai Qian Zhang, Syed Shakib Sarwar, Kleber Hugo Stangherlin, Yi-Lun Lu, Jorge Tomas Gomez, Jae-Sun Seo, et al. 2024. Neural Architecture Search of Hybrid Models for NPU-CIM Heterogeneous AR/VR Devices. *arXiv preprint arXiv:2410.08326* (2024).

Received 28 February 2025; accepted 8 April 2025