Multiple Sensor-based Adaptive Foveated Display Control for Enhanced Computational Efficiency of Virtual Reality Devices

Kimleang Kea[1], Youngsun Han[1], and Tae-Kyung Kim[2*]
[1] Department of AI Convergence, Pukyong National University
Busan 48513, South Korea
[e-mails: kimleangkea@pukyong.ac.kr, youngsun@pknu.ac.kr]
[2] Department of Management Information Systems, Chungbuk National University
Chungbuk 28644, South Korea
[e-mail: kimtk@chungbuk.ac.kr]
[*]Corresponding author: Tae-Kyung Kim

Received May 1, 2024; revised August 6, 2024; accepted September 23, 2024; published December 31, 2024

Abstract

The demand for immersive virtual reality (VR) experiences in high resolution has escalated, necessitating the development of advanced hardware (HW) capable of rendering images with high quality. To optimize the computational efficiency of delivering high-quality images, it becomes imperative to align with the constraints of the human visual system. The foveated rendering approach strategically allocates rendering resources by prioritizing the highest quality at the user's gaze point while reducing image quality in the periphery. In this paper, we propose an adaptive foveated display controller algorithm that leverages a grid-based strategy and seamlessly integrates with varying levels of detail to dynamically construct the foveated rendering technique. Our technique dissects images into grid cells while tracking the user's gaze through an eye-tracking sensor. The resolution is highest at the grid cell where the user is looking, gradually decreasing in nearby grid cells. This efficient approach reduces the need for heavy computing and effectively minimizes rendering latency, which is commonly found in non-foveated rendering methods. We conducted experiments using our proposed approach on several hardware. This setup allowed us to gather data from a range of interconnected sensors and run the adaptive foveated display controller algorithm. The results of our experimentation demonstrated a significant reduction in both latency and resource consumption, clearly outperforming the non-foveated rendering approach.

Keywords: adaptive foveated rendering, embedded device, eye-tracking system, sensors.

1. Introduction

In recent years, the demand for high-resolution and pixel-dense displays to meet the diverse needs of various applications has been growing. For example, virtual reality (VR) technology has gained popularity in a wide range of industries such as military [1, 2], manufacturing [3, 4], entertainment [5, 6], and more [7, 8]. Despite its widespread adoption, current VR technology faces several challenges. These include the demand for significant computational hardware (HW) resources and the persistent issue of high rendering latency, particularly when striving to render images to exacting specifications [9]. As VR devices aim for compactness in small size and the necessity for rendering high-quality images intensifies, the requirement for robust computational power in HW becomes indispensable. Unfortunately, such hardware often comes in large sizes, making it unsuitable for mounting on the user's head. This entails employing a rendering technique meticulously crafted to curtail resource usage. Strategies may include presenting images at diminished resolutions and utilizing selective portions of the image. By doing so, the hardware footprint is minimized, as only specific segments demand high processing power. This approach is commonly referred to as foveated rendering.

Foveated rendering is a method designed to improve the efficiency of visual rendering processes by leveraging the unique characteristics and limitations of the human visual system (HVS). This technique concentrates computational resources on rendering the central area of the image in high quality, while simultaneously reducing detail and resolution in peripheral areas. As a result, foveated rendering not only significantly decreases the computational load and latency required for VR experiences but also diminishes the computational demands for rendering while maintaining high visual quality [10]. Consequently, compact and lower computational hardware can be further minimized in size, enhancing overall efficiency. Moreover, this advancement provides an additional layer of natural look and feel within virtual reality applications. However, Wang et al. [8] have noted three primary challenges for the practical adoption of foveated rendering: utilizing a perceptual model of the HVS to guide foveated rendering, applying different levels of rendering quality to different regions, and integrating foveated rendering techniques into established rendering paradigms for enhanced performance. By prioritizing rendering resources on the central area of focus, grid-based foveated rendering can help reduce latency in VR devices, leading to a smoother and more responsive user experience. This reduction in latency is particularly important for applications where real-time interaction is critical, such as gaming or virtual simulations.

In the past decades, interest has been growing in the computer graphics community regarding the use of eye-tracking to enhance the accuracy and efficiency of foveated rendering. Early attempts at foveated rendering faced challenges due to limited hardware capabilities, compromising accuracy and increasing latency in eye-tracking systems [11]. Nowadays, high-precision eye-tracking cameras offer highly accurate real-time tracking of eye movements. Additionally, a range of powerful and energy-efficient hardware is now available, providing more computational power to render high-quality images in real-time [12]. Despite these advances, significant challenges still need to be addressed, such as optimizing hardware design and algorithms for performance and ensuring low latency across the entire system. Acceptable latency depends on multiple factors, including the complexity of the application and user experience in the visual field. Hence, the development of algorithms that efficiently harness limited hardware resources is crucial for enhancing the popularity, compactness, availability, and affordability of VR devices. Furthermore, another potential limitation lies in the adaptability of display techniques across different VR platforms and hardware configurations.

In this study, we present an adaptive foveated rendering algorithm that leverages a grid-based structure to partition the display resolution into smaller grid cells. This approach dynamically constructs and optimizes the foveated rendering technique by integrating variable levels of detail. The findings underscore a significant decrease in computational resource utilization, thereby enabling the design of more compact VR devices. Furthermore, the proposed algorithms are evaluated to yield substantial latency reduction and lower resource consumption. As a result, this capability is applicable across a wide range of applications, particularly those requiring precise object localization within a single grid cell, such as object detection and robotic hand manipulation. With this approach, only the focal point requires rendering at the highest available resolution. Surrounding areas can undergo a radial decrease in resolution. This strategic adaptation empowers the rendering algorithm to significantly reduce the volume of image or video data for processing. The main contributions of our research are as follows:

- We introduce a system that responsively adapts to the size and position of the foveal region in real-time. Our technique allocates rendering resources based on the user's gaze direction, thereby reducing the computational workload.
- Using a level-of-detail technique, we render the foveal region with enhanced detail and resolution while rendering the peripheral regions at lower detail. The display area is segregated into a grid of rectangular cells, each representing a distinct level of detail and resolution.
- We experimentally demonstrate that our foveated rendering approach achieves lower latency than standard rendering techniques. In addition, we investigate the correlation between grid size and latency, offering valuable insights into optimizing the grid size for various use-cases.

The paper is organized as follows: In Section 2, we provide a concise overview of how foveated rendering leverages the HVS, present the general concepts of FPGAs, and review related works in the field. The system architecture of our proposed adaptive foveated rendering technique is presented in Section 3. In Section 4, we delve into the HW design and algorithms implemented on the embedded HW and FPGA, which were used to test our proposed algorithm. Section 5 outlines the experiments conducted to evaluate the system's performance, focusing on latency. Finally, in Section 6, we conclude the paper.

2. Background and Related Works

2.1 Foveated Rendering

Rendering refers to a realistic or stylized image generated by a computer in either 2D or 3D format. Producing a digital image requires various details of the scene such as the geometry, viewpoint, texture, lighting, and shading to be processed. The technical specifications of rendering methods depend on the application, level of realism, and computational resources [13]. Foveated rendering involves rendering an image with varying levels of detail based on the user's gaze location. This technique reduces the rendering workload by lowering the image quality in the peripheral region while maintaining high image quality in the center of the image. Eye-tracking technology has become an essential component for enhancing the accuracy and efficiency of foveated rendering. However, challenges remain such as ensuring precise and reliable eye tracking and compatibility with different rendering techniques [14].

2.2 Field-Programmable Gate Arrays

Field-programmable gate arrays (FPGAs) are integrated circuits characterized by high versatility and customizability, and they can be programmed and configured to execute specific tasks. Their HW flexibility, substantial computational throughput, and low energy consumption have contributed to their growing popularity in edge computing [15]. Because their architecture can be adapted to accommodate different algorithms, FPGAs provide a consistent and high computational throughput, which makes them suitable for accelerating various algorithms across a broad spectrum of edge computing applications [16].

2.3 Related Works

Many studies have attempted to realize foveated rendering with low latency using different approaches. These attempts can be divided into two categories: multi-spatial resolution and level of detail (LOD).

2.3.1 Multi-spatial Resolution

Research on multi-spatial resolution can be further divided into two categories: perceptual research on foveated rendering, and rendering images or videos using the foveated rendering techniques. Hsu et al. [17] introduced a regression model to analyze the relationship between image quality and foveated rendering parameters. They found that no single subjective assessment method was definitively superior, which indicates that further observations are needed to ascertain the extent of imperceptibility in foveated rendering. Kaplanyan et al. [18] proposed using generative adversarial neural networks to enhance the quality of images and videos in the peripheral region. Their method allows for real-time processing and compatibility with gaze-contingent head-mounted displays on contemporary HW. Deza et al. [19] examined a visual representation of the HVS and utilized this knowledge to encode features and train a convolutional neural network called Foveation-Nets for scene categorization. Foveation-Nets exhibited a distinct visual representation compared to networks without foveated input, which had a positive effect on its generalization, robustness, and perceptual sensitivity. Surace et al. [20] trained a generative network for foveated image reconstruction with the aim of maintaining perceived image statistics rather than natural ones by penalizing perceptually significant deviations in the output.

2.3.2 Level of Detail

Many researches utilized LOD-based techniques for foveated rendering by optimizing the LOD in different regions of an image depending on the user's gaze position. Swafford et al. [21] conducted a user study to compare images generated by foveated rendering with an eccentric angle of 9° and reference images at full resolution presented in random order. The scene geometry was rendered at three LODs: high, medium, and low, where a lower level corresponded to a less tessellated grid for each tile. Their results showed that users perceived similar visual experiences between the foveated image with medium LOD in the peripheral region and the full-resolution reference image. Young et al. [22] proposed a LOD-based foveated rendering technique that also corrects gaze tracing errors or state parameters by adjusting the size and shape of the foveal region. Stafford et al. [23] employed a selective filtering technique to reduce visual artifacts in the peripheral region caused by lower LOD contrast. They then composited the foveated images for presentation. Lindeberg et al. [24] proposed a gaze-contingent depth-of-field tessellation method that applies tessellation to objects within the focal plane and gradually decreases the tessellation level as the applied blur increases. Spagnolo et al. [25] designed a custom hardware architecture capable of reconstructing high-resolution images by treating the foveal region and peripheral region through accurate and inaccurate operations, respectively. However, one potential limitation is its focus solely on a specific custom hardware architecture, designed specifically for low-power super-resolution processing in virtual reality wearable devices. Spagnolo et al. [26] presented a camera, algorithm, and accelerator co-designed lensless eye tracking system dubbed EyeCoD. To the best of our knowledge, this system is the first to provide a general, front-end eye tracking solution for AR/VR while satisfying the requirements for both high throughput and a smaller form factor.

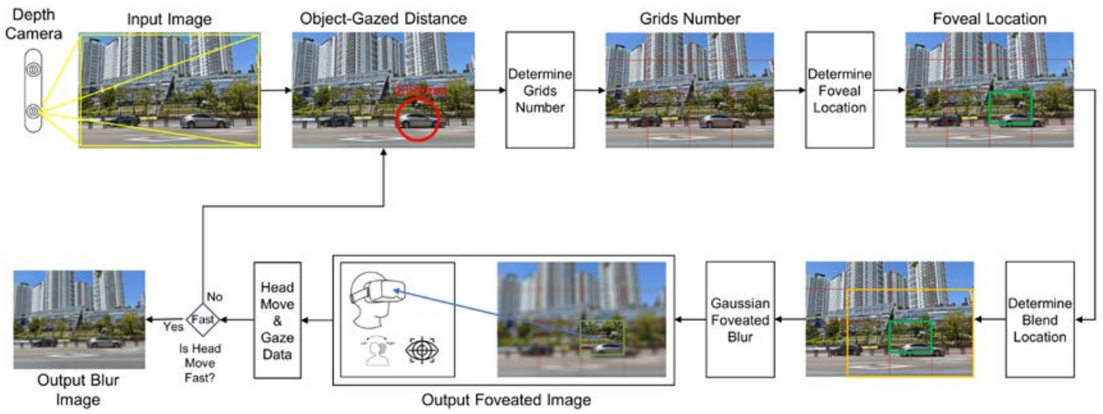


Fig. 1. The overall architecture of the adaptive foveated display control approach leverages sensor data to achieve precise gaze tracking, computation of grid numbers, and dynamic determination of foveal and blend areas. These pieces of information work synergistically to continuously inform and guide the foveated rendering process.

3. Proposed Methodology

In this section, we describe the overall architecture of the proposed adaptive foveated display controller technique and its algorithm in detail.

3.1 Overall Architecture

Fig. 1 illustrates the comprehensive architecture designed to implement the proposed adaptive foveated display controller approach. The proposed method consists of the following six steps: gaze location determination, grid number determination, foveal location identification, blend location identification, gaussian foveated blurring, and head movement speed condition, respectively. Furthermore, these steps are executed iteratively. Initially, a high-resolution depth camera captures an input image. Concurrently, the user's gaze location is determined using an eye-tracking camera, as outlined in Section 3.2. This allows for the precise estimation of the user's focal coordinates within the scene. Second, these gaze coordinates enable us to estimate object distances in the input image from the user's viewpoint. This is crucial for calculating the grid numbers to be displayed, based on the object's distance, as detailed in Section 3.3. Third, the foveal location is assigned to one of these grid cells, a procedure detailed in Section 3.4. Fourth, to enhance the user experience beyond mere foveal location identification, blended areas surrounding the foveal grid cell are determined, as elaborated in Section 3.5. Next, Gaussian foveated blurring is applied to the foveal and blended grid cells, as specified in Section 3.6. Finally, the resultant image is displayed based on the user’s head movement speed. If the head moves rapidly, a fully blurred image is presented, capitalizing on the natural limitations of human visual acuity. Conversely, for stable head movements, the six-step process is executed.

3.2 Gaze Location Determination

In this study, our adaptive foveated display controller technique dynamically calculates the number of grids based on the user's gaze location. The gaze data, received from the eye tracking camera, are represented by variables *x* and *y* corresponding to the horizontal and vertical positions of the gaze in the display, respectively. Variables *x* and *y* are mapped to the display frame, enabling accurate determination of the object-gazed distance and precise measurement of the visual fixation point on objects within the scene [27]. An image is segregated into three regions, particularly the foveal, blend, and peripheral regions. The foveal region position (*i* and *j*) is defined as follows:

$$(i,j) = \left(\left[\frac{x}{\frac{width}{m}}\right], \left[\frac{y}{\frac{height}{n}}\right]\right), if \begin{cases} 0 \leq x \leq width \\ 0 \leq y \leq height \end{cases} \tag{1}$$

where *x* and *y* represent the horizontal and vertical axes of the gaze location, respectively, *width* and *height* denote the fixed horizontal and vertical display resolution in pixels, respectively, of the image source, and *m* and *n* are the numbers of grid columns and rows, respectively. For example, suppose the foveal region position is *i* = 1 and *j* = 1 on a 4x4 grid. In that case, the portion of the image visible to the observer can be rendered at high resolution as illustrated in **Fig. 2**. This is a simplistic method aimed at enabling the loading of individual image regions without requiring specialized image rendering support for such functionality. By implementing a grid overlay on an image, it leads to spatial subdivision of foveated rendering.

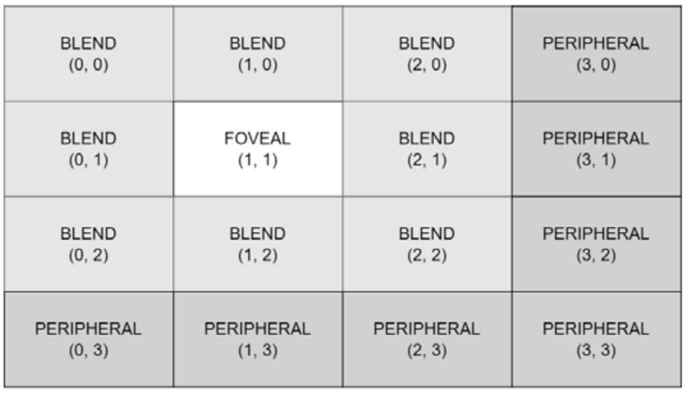


Fig. 2. The foveal region located at the *i* = 1 and *j* = 1, surrounded by blend and peripheral grids.

3.3 Grid Number Determination

The adaptive foveated display controller integrates three pivotal components: the input image source, head motion data, and eye-tracking data. The input image is displayed on the screen, and the user's head movement is monitored to determine its speed. During rapid head motions, the resolution is reduced, as our visual perception struggles to discern fine details in such instances. Conversely, when head movement is gradual, the system precisely identifies the gaze coordinates, pinpointing the object location and distance in the display screen. **Fig. 1** depicts the comprehensive process wherein integrated sensor data is harnessed to detect and compute object-gaze distance, ultimately guiding the selection of the grid for presentation on the screen. The final rendering stage splits the display to multi-resolution rendering in high-, mid-, and low-quality grid cells.

To ensure the adaptability of the foveated rendering, it is crucial to dynamically adjust the number of grids based on the specific requirements of the scene and the user's gaze location. Determining the appropriate number of grids within our approach depends on the distance to the object being focused on. The depth camera sensor is utilized to determine the distance of the objects in the ideal range of 0 to 6000 millimeters. This involves the utilization of predetermined grid configurations, which are systematically aligned with object distances spanning ranges of 0~1000, 2000~3000, 3000~4000, 4000~5000, and 5000~6000 millimeters [28]. Corresponding to these ranges, specific grid sizes of 2x2, 3x3, 4x4, 5x5, 6x6, and 7x7 are implemented, respectively [27, 29]. Considering the visual property where an object appears to decrease in size with increasing distance, applying a greater number of grids becomes strategically beneficial.

```
Input:  S: head movement speed,
        G: gazed point location (x,y),
        D: object distance at gazed point
Output: grids: total number of grids,
        fovealX: position of foveal in X-axis,
        fovealY: position of foveal in Y-axis
 1: procedure determine_number_of_grids(D):
 2:   if D > 0 and D <= 1000 then
 3:     grids = 4
 4:   else if D > 1000 and D <= 2000 then
 5:     grids = 9
 6:   else if D > 2000 and D <= 3000 then
 7:     grids = 16
 8:   else if D > 3000 and D <= 4000 then
 9:     grids = 25
10:   else if D > 4000 and D <= 5000 then
11:     grids = 36
12:   else if D > 5000 and D <= 6000 then
13:     grids = 49
14: end procedure
15: procedure foveal_location(S, G, D):
16:   if S is Fast then
17:     grids = 1
18:     fovealX = 1
19:     fovealY = 1
20:   else
21:     width = 1280
22:     heigh = 720
22:     gazeX, gazeY = G[0], G[1]
23:     grids = determine_number_of_grids(D)
24:     cell_width  = width / sqrt(grids)
25:     cell_height = height / sqrt(grids)
26:     fovealX = gazeX / cell_width
27:     fovealY = gazeY / cell_height
28:   end if
29: end procedure
```
Fig. 3. Algorithm of grid-based foveated rendering.

3.4 Foveal Location Determination

Each grid setup presents diverse rendering levels in terms of detail and resolution. Grid cell sizes are calculated by dividing the fixed display dimensions (e.g., 1280x720 pixels) by the designated number of grids count. For instance, a 4x4 grid entails cells of 320-pixel width

(1280 divided by 4) and 180-pixel height (720 divided by 4). This division ensures each cell uniformly covers a proportionate display segment, guaranteeing a balanced allocation of rendering resources. By utilizing Equation 1, which considers the grid size, the gaze location, and the distance to the object being focused on, we can determine the foveal location. For instance, consider a scenario where the gaze location (x, y) is precisely identified at the pixel coordinates (330, 190) on the screen. By using Equation 1, the result of foveal location is determined at (1, 1). This signifies that the object under fixation precisely aligns with the grid cell positioned at coordinates (1, 1). This portion of the image takes on a high-resolution form, presenting a high-quality image. In another example, consider a 5x5 grid configuration. In this setup, each grid cell measures 256 pixels in width and 144 pixels in height. For example, if the gaze coordination is denoted by (x, y) = (200, 300), the resulting position of the foveal grid cell can be specified by coordinates (i, j) = (0, 2). To provide a step-by-step understanding of the foveal region extraction process, we utilize algorithm in **Fig. 3**, which outlines a procedure for determining the foveal location based on the user's gaze location and the predefined grid configurations. This algorithm encompasses the calculation of grid cell sizes, the determination of the foveal location, and subsequent rendering resource allocation.

By focusing rendering resources solely on a single grid cell, grid-based foveated rendering can significantly reduce the computational load, improving overall performance and enabling smoother experiences, particularly in resource-intensive applications that require high-resolution rendering. In areas where users are most attentive, grid-based foveated rendering can enhance immersion by ensuring that the most critical areas of the scene are rendered with higher fidelity, leading to a more realistic experience. However, grid-based foveated rendering may introduce visual distortions at the boundaries of the foveal region, where the transition between high and low detail areas occurs. This can potentially degrade the overall visual quality and negatively impact user experience, particularly if not implemented carefully.

3.5 Blend Location Determination

In our proposed method, we divide the levels of detail into three regions, simply locating the foveal location is insufficient. It is essential to render the areas surrounding the foveal location with a less refined quality compared to the foveal location itself. To identify the surrounding blend areas around the foveal location, we employ the following equation:

$$b_{i,j} = \{g_{k,l}\} - \{g_{i,j}\}\ for\ all\ k\ and\ l, \tag{2}$$

where $i - 1 \leq k \leq i + 1$ and $j - 1 \leq l \leq j + 1$ and k and l are blended grid cells surrounding the foveal grid.

The $b_{i,j}$ denotes the set of blended cells when the foveal cell $g_{i,j}$ is at a position of i and j. Equation 2 also upholds the following conditions: if $k < 0$, then $k = 0$; if $k \geq m$, it is adjusted to $k = m - 1$. Similarly, l must be 0 and $n - 1$ if $l < 0$ and $l \geq n$, respectively, when m and n are the number of columns and rows of the grid. For instance, we can get the blended cells in a scenario of a 5x5 grid, and where $i = 0$ and $j = 2$, as follows:

$$b_{0,2} = \{g_{0,1}, g_{1,1}, \cancel{g_{0,2}}, g_{1,2}, g_{0,3}, g_{1,3}\} - \{\cancel{g_{0,2}}\} \tag{3}$$

By excluding the foveal location from the blended areas, we can identify the surround blended areas and can dynamically adjust the number of grids and allocate rendering resources based on the user's gaze location. The regions beyond the blended boundaries, known as peripheral, will be rendered with a blur effect. The fovea and blended areas will be integrated together to provide the final rendering output. This adaptive foveated display controller technique optimizes the rendering process, ensuring a high-quality visual experience.

3.6 Gaussian Foveated Blurring

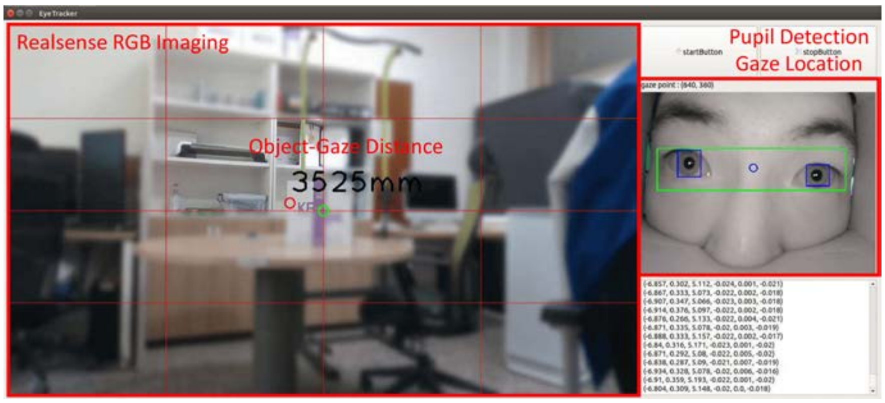
We employed the Gaussian kernel method to selectively introduce blurring across the entire image while preserving sharpness exclusively at the foveal point. Additionally, a slight blur was applied to the blend locations, ensuring a seamless integration of the visual elements within the image. The Gaussian blur technique has been recognized as an invaluable tool for filtering images, particularly when they are characterized by the presence of significant noise [30]. Its efficacy stems from the manner in which it mitigates noise during the filtering process, relying mainly on the variance parameter of the Gaussian kernel. This dependency on the kernel's variance underscores its vital role in shaping the extent and quality of the blur. Moreover, the Gaussian blur technique contributes to a reduction in computational power required for image rendering [31]. This reduction is facilitated by the blurring effect, which imparts a smoother appearance to the image. Consequently, the computational demands are lessened, enhancing the overall efficiency of the rendering process and demonstrating the utility of this approach within our adaptive foveated display control system.

4. Hardware Implementation

To comprehensively evaluate the performance of a HW implementation of the proposed approach, we outline its key components such as software (SW), peripheral devices, and a communication interface. These elements enable the evaluation of our algorithm.

4.1 Software Components

The SW elements encompass a depth camera SDK, a Python-based graphical user interface (GUI) library, and a deep learning model. Within the depth camera SDK, both RGB and depth images are acquired to facilitate foveated rendering and calculate distances for object-gazed interaction. The embedded HW offers high performance and low power consumption for deep learning and computer vision tasks [32]. The GUI application allows user interaction and integrates with the depth camera SDK and deep learning model as shown in **Fig. 4**. Users are required to wear a cardboard-like headset equipped with an eye-tracking camera. During initialization, users need to open their eyes clearly to allow for the capture and tracking of their pupils. Once captured, users can then simply look around the display screen.



```
(-6.857, 0.302, 5.112, -0.024, 0.001, -0.021)
(-6.867, 0.333, 5.073, -0.022, 0.002, -0.018)
(-6.907, 0.347, 5.066, -0.023, 0.003, -0.018)
(-6.914, 0.376, 5.097, -0.022, 0.002, -0.018)
(-6.876, 0.266, 5.133, -0.022, 0.004, -0.021)
(-6.871, 0.335, 5.078, -0.02, 0.003, -0.019)
(-6.888, 0.333, 5.157, -0.022, 0.002, -0.017)
(-6.84, 0.316, 5.171, -0.023, 0.001, -0.02)
(-6.871, 0.292, 5.08, -0.022, 0.005, -0.02)
(-6.838, 0.287, 5.09, -0.021, 0.007, -0.019)
(-6.934, 0.328, 5.078, -0.02, 0.006, -0.016)
(-6.91, 0.359, 5.193, -0.022, 0.001, -0.02)
(-6.804, 0.309, 5.148, -0.02, 0.0, -0.018)
```

Fig. 4. GUI developed by using the PyQT5, intel realsense SDK, and a deep learning model to accurately detect the pupil and gaze locations. The GUI offers real-time visualization of the object being gazed upon and its distance in millimeters.

4.2 Hardware Components

The approach employs sensors such as a depth camera, eye-tracking camera, and gyroscope, linked to embedded HW for data collection and preprocessing. The depth camera measures distances at the gazed location in the display, enhancing accuracy with an infrared projector, whereas the eye-tracking camera utilizes a pre-trained deep learning model to locate the user's gaze in high-quality video [33]. Precision head movement speed detection is achieved with the gyroscope. Moreover, the FPGA development board combines programmable logic and an integrated ARM processor for custom HW acceleration to further process the data. These devices collectively ensure accurate and reliable data analysis. **Fig. 5** shows the HW components and their connections. The Jetson Nano, powered by CUDA architecture, offers high-performance GPU computing suitable for image processing tasks. By offloading computationally intensive workloads from the Jetson Nano to the FPGA, it can improve the overall system performance and efficiency.

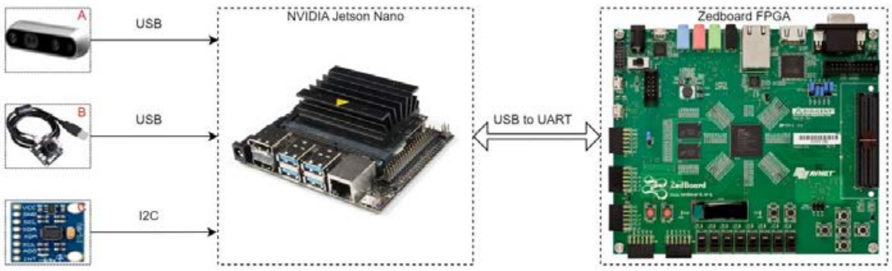


Fig. 5. Connections among embedded HW: (a) Intel Realsense D435 sensor that captures images at a resolution of 1280×720 pixels. (b) A Day-and-night camera that captures high-quality images for eye location detection. (c) An MPU6050 gyroscope sensor that measures the angular velocity around an axis to capture head movement speed.

4.3 Communication Protocols

A communication protocol, such as the universal asynchronous receiver-transmitter (UART), facilitates data transmission between HW entities, enabling serially transmission of data one bit at a time. UART operates in simplex, half-duplex, and full duplex modes. AXI controller offers AXI4-lite for low-latency and AXI4-stream for high-speed data transfer [34]. Utilizing AXI, direct data transfer between device and memory is possible without CPU intervention. This enables efficient communication between PS and PL modules.

5. Performance Evaluation

In our study, we conducted an experiment to evaluate the efficacy of our proposed adaptive foveated display controller technique. This technique was rigorously evaluated across key parameters, including latency reduction, grid size impact, and HW resource consumption.

5.1 Experimental Tools

In our proposed approach, the experimental tools consist of components as follows:

- **FPGA platform:** We utilize the Zedboard FPGA development board to implement algorithm in **Fig. 3**. The proposed algorithm is written in the Verilog language, where it is implemented in the PL of the FPGA. Then, it is packaged as intellectual property (IP) for the PS to connect to using the AXI interface.

- **Embedded HW:** The Jetson Nano is employed and serves as a middleware, connecting all sensors and facilitating foveated rendering on the display monitor.
- **Realsense Camera:** To capture both color and depth frames, we utilize an Intel RealSense D435 sensor. Operating at a resolution of 1280x720 pixels, this sensor serves as the visual input source and provides distance measurements through depth frames. The resulting data is then visualized through a GUI, offering users visuals to examine and rendering a foveated display.
- **Eye-tracking Camera:** This camera is employed for capturing the user's pupils and pinpointing their gaze location within the rendering viewport of the Realsense camera display. Utilizing a deep learning model, SSD-MobilenetV2, enables the detection of the pupil and computation of its location.
- **Gyroscope Sensor:** A combination of a three-axis accelerometer and a three-axis gyroscope was employed for accurate speed computation of head movements. This setup is integrated with the eye-tracking camera to facilitate data acquisition and precise calculation.

The selection of these HW implementations is instrumental in evaluating the performance of our proposed algorithm. It is important to note that the chosen HW implementations are strictly used for evaluation purposes. This decision facilitates a thorough evaluation of its efficacy regarding data acquisition, latency, and resource consumption.

5.2 Experimental Workflow

Fig. 5 visually demonstrates the connection and communication protocol necessary for extracting and manipulating data from each sensor. To execute the grid-based foveated rendering algorithm within the PS/PL of FPGA, we followed a well-defined workflow for the integrated sensors, as detailed below:

- All sensors data are acquired by the Jetson Nano through various connections, including USB and pin interfaces.
- The Jetson Nano transmits sensor data via a USB-UART connection, including head movement, gaze location, and object distance data formatted in serial. This data is then received by the PS of the FPGA. Subsequently, the PS component of the FPGA extracts this data and transmits it to the PL using the AXI interface for further processing.
- Head movement data is utilized to determine the speed of its movement. When classified as fast, the grid value changes to one, resulting in a blurred representation of the associated image. This occurs because, as humans, our vision naturally blurs when our heads move rapidly, making it challenging to perceive details clearly.
- Object distance primarily determines the total grid count, with fixed intervals ranging from 0 to 6000 mm as conditions, influencing the number of grids accordingly.
- Once the total number of grids, the dimensions of each grid cell are computed in accordance with the image width and height. This enables precise localization of the foveal grid cell within the visual field.
- Gaze positions are determined using pixel coordinates to precisely pinpoint the foveal grid cell within a specified set of grids. Subsequently, the results are transmitted back to the Jetson Nano for processing and rendering.

5.3 Experimental Results

5.3.1 Latency Reduction

Here, we conducted a comparative analysis between our approach and the conventional non-foveated rendering (non-FR) technique. Non-FR involves rendering at full resolution without using foveated rendering. Our proposed method showcased a notable decrease in latency, highlighting its effectiveness over non-FR approaches. **Table 1** compares the average latency values across three stages: frame capture, foveated image construction, and foveated rendering in the GUI. The most time-consuming stage was foveated image construction, which required a significant amount of processing to construct images in real-time. This posed a significant challenge on the embedded HW with constrained computing resources. The data retrieval from the eye-tracking camera and gyroscope was equally demanding, further burdening the computational capacity of the embedded system. Our proposed approach markedly enhances efficiency, reducing computational load and end-to-end latency by 15% compared to the non-FR approach. In the context of frame capture, it is crucial to understand that images are composed of pixels acquired by the depth camera sensor. The average latency per 30 frame captures was 23.7ms. This duration provides a comprehensive assessment of system performance over a reasonable timeframe, while minimizing the impact of outliers. Regarding the foveated image construction latency, the amount of latency depends on the quantity of data involved. The non-FR approach requires processing the entire frame to achieve a high image quality; this takes longer than foveated rendering, which only generates high-quality pixels in the foveal region and reduces the quality in other areas. About foveated rendering latency, the constructed frames are then utilized to render a video in a GUI. This stage involves low latency due to the data are displayed on the screen immediately upon construction.

Table 1. Comparison of end-to-end latency of our proposed approach with non-FR.

Stage	Our Approach	Non-FR
Frame Capturing	23.7	23.7
Foveated Construction	117.2	140.7
Frame Rendering	15.5	17.3

We evaluated the latency of our proposed approach by comparing it with the study described in [35], known as FoReCast. The end-to-end latency in FoReCast is comprised of several sub-components within the framework: (1) at the remote site, which includes the processes of data acquisition, ray-casting, conversion, sampling, and encoding; and (2) at the user site, encompassing the processes of decoding, conversion, and rendering. Within the FoReCast methodology, end-to-end latency was calculated for both office and living room scenarios, resulting in average latencies of 287.2ms and 322.8ms, respectively. In contrast, our approach, which utilizes predefined grid numbers, significantly reduced the average latency to 155.25ms. This marked improvement highlights the superiority of our method, showcasing the effectiveness of our adaptive foveated display control approach. **Table 2** illustrates that even though FoReCast employs significantly superior technology, the latency remains quite unacceptable when compared to our approach, which utilizes simpler SW/HW components.

Table 2. Framework implementation of our approach and FoReCast.

SW/HW Components	Our Approach	FoReCast
Eye Tracking	USB Webcam	Tobii Eye Tracking VR Headset
RGB Imaging	Intel Realsense Camera	ZED Stereoscopic Camera
User OS and GPU	Jetson Nano and FPGA	Windows and GTX 1080
Server OS and GPU	N/A	Ubuntu and Nvidia GP104M
Communication	USB, UART, and AXI	Ethernet LAN with 10Gbit/s Switch

5.3.2 Grid Size Effect

The total number of grids was found to have a negligible effect on the end-to-end latency of the system. Our experiments showed that a smaller grid number resulted in lower latency because larger grids require more processing to extract and blur the relevant portions of the video frame. We tested square grids of 2x2, 3x3, 4x4, 5x5, 6x6, and 7x7 to ensure equal numbers of rows and columns. As an example for the 5x5 grid, we segmented the video frame into 25 distinct portions and classified each as foveal, blended, or other areas. We then applied clear and less blur to the foveal and blended areas, respectively, while blurring other areas more. **Fig. 6** visualizes the influence of the grid size on the overall end-to-end system latency.

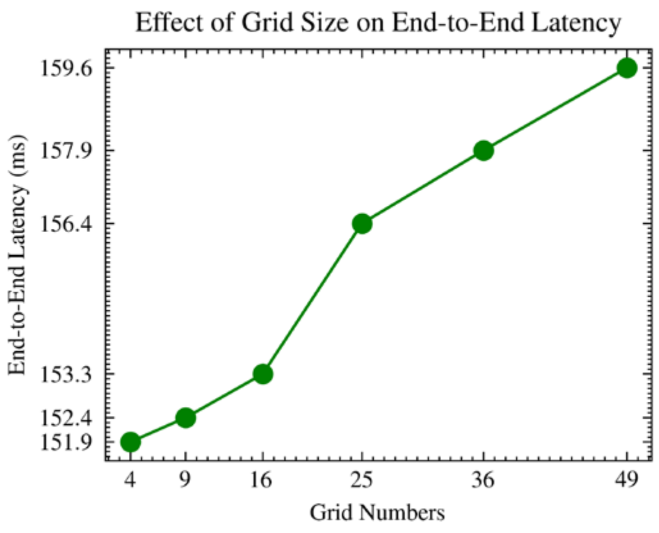


Fig. 6. The plot displays the latency values measured for six different grid sizes: 4, 9, 16, 25, 36, and 49. The x-axis represents the grid size while the y-axis represents the measured latency. The markers on the plot indicate the recorded values.

Fig. 7 shows representative images with different grid sizes. Each image shows the gaze location and distance to the object. These representations allow for better understanding of the effects of different grid numbers on the foveated rendering process.

5.3.3 Resource Usage

We conducted experiments to evaluate the resource consumption of normal and foveated rendering techniques using Jetson Nano. We utilized Jetson Stats to provide real-time information on the CPU, GPU, and RAM consumption of the Jetson Nano platform, which is displayed on the monitor. This feature enabled efficient monitoring and optimization of the system performance.

Table 3. Comparison of resource consumption of our approach and non-FR.

Components	Our Approach	Non-FR
CPU (%)	60.9	61.1
GPU (%)	64.8	73.5
RAM (GB)	2.8	2.9

As shown in **Table 3**, there are slight differences in CPU and RAM usage between foveated rendering and normal rendering. Foveated rendering consumes 60.9% of CPU and 2.8GB of RAM, compared to 61.1% and 2.9GB, respectively, for normal rendering. More notable disparities, however, are evident in GPU and power consumption. Foveated rendering utilizes 64.8% of GPU resources, marking a decrease of approximately 11.8% compared to the 73.5% consumed by non-FR. Although foveated rendering has a minimal impact on CPU and RAM usage, it significantly optimizes GPU and power consumption, highlighting its superior efficiency in resource utilization.

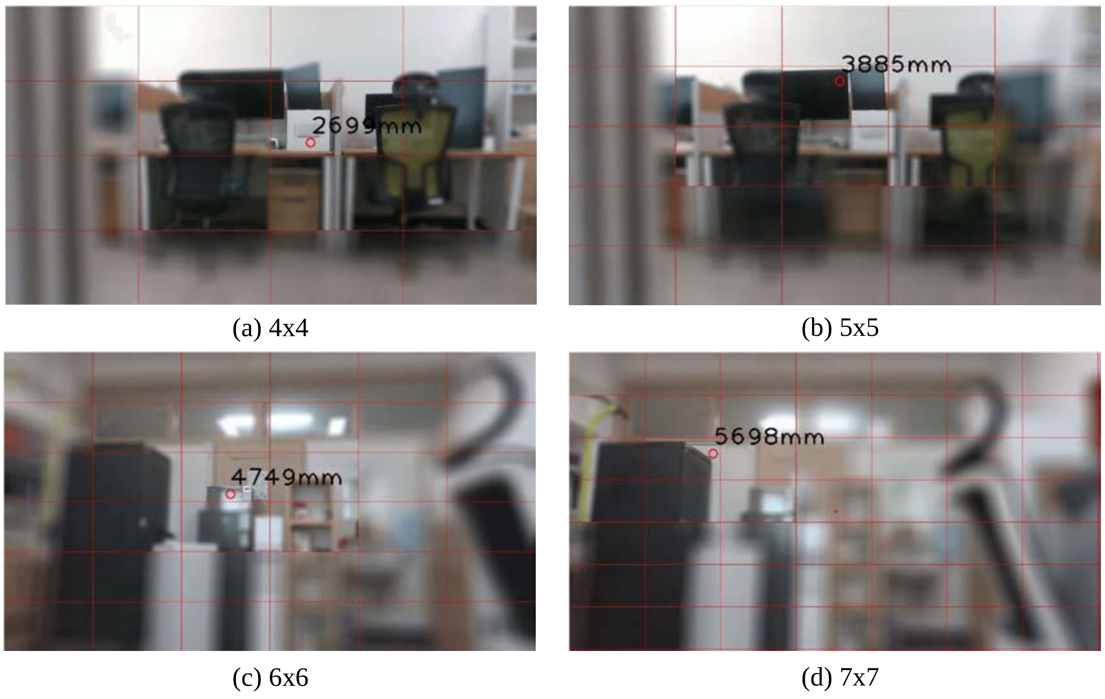


Fig. 7. Application of the proposed foveated rendering technique using different grid sizes to determine the level of detail. The system dynamically adjusts the level of detail in different image regions based on the user's gaze, which results in improved visual quality and more efficient processing. The red dot on the screen indicates where the user is looking, and the value in millimeters indicates the distance between the user's gaze and the object being viewed.

6. Conclusion

We proposed an adaptive foveated display controller technique that dynamically modulates the image details based on the user’s gaze location. This technique effectively reduces the computational demands of high-resolution displays. Our method, which partitions the image into grids and selectively renders the high-resolution details in the grid cell that aligns with the gaze location, reduces the latency by 15% from those of traditional non-FR rendering methods and optimizes the resource consumption. Moreover, our approach is superior compared to FoReCast method in terms of end-to-end latency. The evaluation was conducted on the embedded HW utilizing FPGA technology. The results demonstrate a substantial reduction in the computational resources required by the embedded HW. By implementing our proposed approach, the rendering process demonstrates a remarkable improvement of around 11.8% in HW resource consumption, when both the embedded HW and FPGA execute the corresponding application concurrently. As a result, this capability is applicable across a wide range of applications, particularly those requiring precise object localization within a single grid cell, such as object detection and robotic hand manipulation. Despite these promising results, further advancements should be made. One potential limitation lies in the adaptability of our technique across different HW configurations. While we have demonstrated its efficacy within controlled settings, variations in device capabilities, display resolutions, and tracking systems may affect its performance in practical applications. In future research, we will integrate our algorithm into more powerful and diverse embedded HW systems, which will potentially benefit VR applications, advanced telecommunication systems, and other sectors reliant on immersive technologies. However, we understand that the effectiveness of our technique may vary depending on the specific application context. While we have demonstrated its utility in certain scenarios, there may be constraints or requirements in other domains that limit its practicality or efficacy. With continuous refinement and application, foveated rendering techniques such as the proposed technique can potentially enhance display-system performance and resource utilization, increasing the accessibility and efficiency of advanced visual experiences.

Acknowledgement

This material is based upon work supported by the Ministry of Trade, Industry & Energy (MOTIE, Korea) under Industrial Technology Innovation Program. No. 20016150, Development of low latency multi-functional Microdisplay controller for eye-wear devices.

References

[1] Rizzo, A., Morie, J. F., Williams, J., Pair, J., Buckwalter, J. G., “Human Emotional State and its Relevance for Military VR Training,” in *Proc. of the 11th International Conference on Human Computer Interaction*, 2005. Article (CrossRef Link)

[2] Lele, A., “Virtual reality and its military utility,” *Journal of Ambient Intelligence and Humanized Computing*, vol.4, pp.17-26, 2013. Article (CrossRef Link)

[3] Nee, A.Y.C., Ong, S.K., “Virtual and Augmented Reality Applications in Manufacturing,” *IFAC Proceedings Volumes*, vol.46, pp.15-26, 2013. Article (CrossRef Link)

[4] Choi, S., Jung, K., Noh, S. D., “Virtual reality applications in manufacturing industries: Past research, present findings, and future directions,” *Concurrent Engineering*, vol.23, no.1, pp.40-63, 2015. Article (CrossRef Link)

[5] Avila, L., Bailey, M., “Virtual Reality for the Masses,” *IEEE Computer Graphics and Applications*, vol.34, pp.103-104, 2014. Article (CrossRef Link)

[6] Bialkova, S., Van Gisbergen, M. S., “When sound modulates vision: VR applications for art and entertainment,” in *Proc. of the 2017 IEEE 3rd Workshop on Everyday Virtual Reality (WEVR)*, pp.1-6, 2017. Article (CrossRef Link)

[7] Ferdani, D., Fanini, B., Piccioli, M. C., Carboni, F., Vigliarolo, P., “3D reconstruction and validation of historical background for immersive VR applications and games: The case study of the Forum of Augustus in Rome,” *Journal of Cultural Heritage*, vol.43, pp.129-143, 2020. Article (CrossRef Link)

[8] Wang, L., Shi, X., Liu, Y., “Foveated rendering: A state-of-the-art survey,” *Computational Visual Media*, vol.9, pp.195-228, 2023. Article (CrossRef Link)

[9] Albert, R., Patney, A., Luebke, D., Kim, J., “Latency Requirements for Foveated Rendering in Virtual Reality,” *ACM Transactions on Applied Perception (TAP)*, vol.14, no.4, pp.1-13, 2017. Article (CrossRef Link)

[10] Mohanto, B., Islam, A. T., Gobbetti, E., Staadt, O., “An integrative view of foveated rendering,” *Computers & Graphics*, vol.102, pp.474-501, 2022. Article (CrossRef Link)

[11] Roth, T., Weier, M., Hinkenjann, A., Li, Y., Slusallek, P., “A Quality-Centered Analysis of Eye Tracking Data in Foveated Rendering,” *Journal of Eye Movement Research*, vol.10, no.5, 2017. Article (CrossRef Link)

[12] Lee, D., Lee, E., Hwang, Y., “Lossless Reconstruction of Convolutional Neural Network for Channel-Based Network Pruning,” *Sensors*, vol.23, no.4, 2023. Article (CrossRef Link)

[13] Tewari, A., Thies, J., Mildenhall, B., Srinivasan, P., Tretschk, E., Yifan, W., Lassner, C., Sitzmann, V., Martin-Brualla, R., Lombardi, S. et al., “Advances in Neural Rendering,” *Computer Graphics Forum*, vol.41, no.2, pp.703-735, 2022. Article (CrossRef Link)

[14] Jabbireddy, S., Sun, X., Meng, X., Varshney, A., “Foveated Rendering: Motivation, Taxonomy, and Research Directions,” *arXiv preprint*, arXiv:2205.04529, 2022. Article (CrossRef Link)

[15] Biookaghazadeh, S., Zhao, M., Ren, F., “Are FPGAs Suitable for Edge Computing?,” in *Proc. of the USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18)*, 2018. Article (CrossRef Link)

[16] Quraishi, M. H., Tavakoli, E. B., Ren, F., “A Survey of System Architectures and Techniques for FPGA Virtualization,” *IEEE Transactions on Parallel and Distributed Systems*, vol.32, no.9, pp.2216-2230, 2021. Article (CrossRef Link)

[17] Hsu, C.-F., Chen, A., Hsu, C.-H., Huang, C.-Y., Lei, C.-L., Chen, K.-T., “Is Foveated Rendering Perceivable in Virtual Reality?: Exploring the Efficiency and Consistency of Quality Assessment Methods,” in *Proc. of the MM '17: Proceedings of the 25th ACM international conference on Multimedia*, pp.55-63, 2017. Article (CrossRef Link)

[18] Kaplanyan, A. S., Sochenov, A., Leimkühler, T., Okunev, M., Goodall, T., Rufo, G., “DeepFovea: neural reconstruction for foveated rendering and video compression using learned statistics of natural videos,” *ACM Transactions on Graphics (TOG)*, vol.38, no.6, pp.1-13, 2019. Article (CrossRef Link)

[19] Deza, A., Konkle, T., “Emergent Properties of Foveated Perceptual Systems,” *arXiv preprint*, arXiv:2006.07991, 2020. Article (CrossRef Link)

[20] Surace, L., Wernikowski, M., Tursun, O. T., Myszkowski, K., Mantiuk, R., Didyk, P., “Learning Foveated Reconstruction to Preserve Perceived Image Statistics,” *arXiv preprint*, arXiv:2108.03499, 2021. Article (CrossRef Link)

[21] Chaitanya, C. R. A., Kaplanyan, A. S., Schied, C., Salvi, M., Lefohn, A., Nowrouzezahrai, D., Aila, T., “Interactive reconstruction of Monte Carlo image sequences using a recurrent denoising autoencoder,” *ACM Transactions on Graphics (TOG)*, vol.36, no.4, pp.1-12, 2017. Article (CrossRef Link)

[22] Swafford, N. T., Iglesias-Guitian, J. A., Koniaris, C., Moon, B., Cosker, D., Mitchell, K., “User, metric, and computational evaluation of foveated rendering methods,” in *Proc. of the SAP '16: Proceedings of the ACM Symposium on Applied Perception*, pp.7-14, 2016. Article (CrossRef Link)

[23] Young, A., Stafford, J. R., Real-time user adaptive foveated rendering, US Patent, US10192528B2, 2019. Article (CrossRef Link)
[24] Stafford, J. R., Young, A., Selective peripheral vision filtering in a foveated rendering system, US Patent, US10169846B2, 2019. Article (CrossRef Link)
[25] Lindeberg, T., "Concealing rendering simplifications using gazecontingent depth of field," *Dissertation*, 2016. Article (CrossRef Link)
[26] Spagnolo, F., Corsonello, P., Frustaci, F., Perri, S., "Design of a Low-Power Super-Resolution Architecture for Virtual Reality Wearable Devices," *IEEE Sensors Journal*, vol.23, no.8, pp.9009-9016, 2023. Article (CrossRef Link)
[27] You, H., Zhao, Y., Wan, C., Yu, Z., Fu, Y., Yuan, J., Wu, S., Zhang, S., Zhang, Y., Li, C. et al., "EyeCoD: Eye Tracking System Acceleration via FlatCam-Based Algorithm and Hardware Co-Design," *IEEE Micro*, vol.43, no.4, 2023. Article (CrossRef Link)
[28] Kim, H. W., Yang, J. W., Yang, J. Y., Jang, J. H., Park, W. C., "MPEG-DASH SRD based 360 VR Tiled Streaming System for Foveated Rendering," in *Proc. of the 2018 International Conference on Information and Communication Technology Convergence (ICTC)*, pp.587-591, 2018. Article (CrossRef Link)
[29] Wang, H., Lin, Y., Xu, X., Chen, Z., Wu, Z., Tang, Y., "A Study on Long-Close Distance Coordination Control Strategy for Litchi Picking," *Agronomy*, vol.12, no.7, 2022. Article (CrossRef Link)
[30] Campos, R., Quintana, J., Garcia, R., Schmitt, T., Spoelstra, G., M. A Schaap, D., "3D Simplification Methods and Large Scale Terrain Tiling," *Remote Sensing*, vol.12, no.3, 2020. Article (CrossRef Link)
[31] Kostková, J., Flusser, J., Lébl, M., Pedone, M., "Handling Gaussian blur without deconvolution," *Pattern Recognition*, vol.103, 2020. Article (CrossRef Link)
[32] Tian, X., Günther, T., "A Survey of Smooth Vector Graphics: Recent Advances in Repr esentation, Creation, Rasterization, and Image Vectorization," *IEEE Transactions on Visualization and Computer Graphic*s, vol.30, no.3, pp.1652-1671, 2024. Article (CrossRef Link)
[33] Mokatren, M., Kuflik, T., Shimshoni, I., "3D Gaze Estimation Using RGB-IR Cameras," *Sensors*, vol.23, no.1, 2022. Article (CrossRef Link)
[34] Rakhmatulin, I., Duchowski, A. T., "Deep Neural Networks for Low-Cost Eye Tracking," *Procedia Computer Science*, vol.176, pp.685-694, 2020. Article (CrossRef Link)
[35] Jeevan, B., Sahithi, P., Samskruthi, P., Sivani, K., "Simulation and synthesis of UART through FPGA Zedboard for IoT applications," in *Proc. of the 2022 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI)*, pp.1-7, 2022. Article (CrossRef Link)
[36] Tefera, Y. T., Mazzanti, D., Anastasi, S., Caldwell, D. G., Fiorini, P., Deshpande, N., "FoReCast: Real-time Foveated Rendering and Unicasting for Immersive Remote Telepresence," in *Proc. of the International Conference on Artificial Reality and Telexistence Eurographics Symposium on Virtual Environments*, 2022. Article (CrossRef Link)

Kimleang Kea received his bachelor's degree in computer science and engineering from the Royal University of Phnom Penh, Phnom Penh, Cambodia, in 2020. He is currently working towards an M.S. degree in the Department of AI Convergence at Pukyong National University, Busan, South Korea. His research interests include deep learning, embedded systems, and quantum computing.

Youngsun Han received his B.S. and Ph.D. degrees in Electrical Engineering from Korea University, Seoul, South Korea, in 2003 and 2009, respectively. He was a senior engineer at the System LSI, Samsung Electronics, Suwon, South Korea, from 2009 to 2011. He was an assistant/associate professor with the Department of Electronic Engineering at Kyungil University, Gyeongsan-si, South Korea, from 2011 to 2019. He is currently a professor in the Department of Computer Engineering at Pukyong National University, Busan, South Korea. His research interests include Quantum computing, compiler construction, microarchitecture, and high-performance computing.

Tae-Kyung Kim earned his Master of Engineering and Ph.D. degrees from the Department of Information Industry Engineering at Chungbuk National University in Cheongju, South Korea, in 2005 and 2010, respectively. From 2013 to 2019, he served as a director at the Korea Software HRD Center in Phnom Penh, Cambodia. He is currently an Assistant Professor in the Department of Management Information Systems at Chungbuk National University. His research interests include big data, artificial intelligence, and software education.