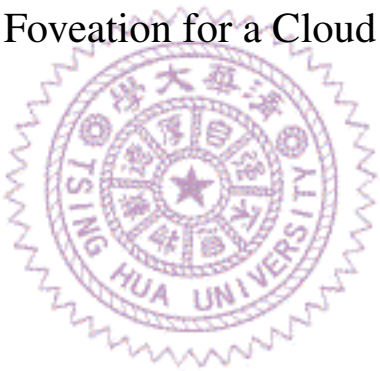
國立清華大學電機資訊學院資訊工程研究所
碩士論文
Department of Computer Science
College of Electrical Engineering and Computer Science
National Tsing Hua University
Master Thesis

最佳化雲端虛擬實境遊戲平台之動態注視點調適功能

Optimizing Dynamic Foveation for a Cloud VR Gaming Platform



方嘉瑋
Jia-Wei Fang

學號：111062563
Student ID:111062563

指導教授：徐正炘 博士
Advisor: Cheng-Hsin Hsu, Ph.D.

中華民國 113 年 3 月
March, 2024

國立清華大學
資訊工程研究所

碩士論文

最佳化雲端虛擬實境遊戲平台之動態注視點調適功能

方嘉瑋

Abstract

Cloud Virtual Reality (VR) gaming offloads the computationally intensive rendering tasks from resource-limited Head-Mounted Displays (HMDs) to cloud servers. Streaming HMD viewports over the Internet consumes a staggering amount of bandwidth for high-quality gaming experiences. One way to cope with such high bandwidth demands is to capitalize on the characteristics of Human Vision Systems (HVS) by allocating a higher bitrate to the foveal region, which is known as foveation. Although foveation was employed by remote VR gaming, existing projects all adopt static foveation, in which the gamer gaze positions are assumed to be fixed at the viewport center. In this article, we explore the potential of dynamic foveation in cloud VR gaming. First, we construct a first dynamic-foveation-enabled cloud VR gaming platform. In our platform, the real-time gaze positions of gamers are streamed from HMD eye trackers to cloud servers, which in turn dynamically adjust the location of the foveal region. Several other foveation parameters, such as foveal region size and peripheral region quality can also be adjusted. Using our platform, we carry out a user study to justify the value of dynamic foveation and derive the optimal foveation parameters for maximizing the gaming Quality of Experience (QoE) in Mean Opinion Score (MOS). Compared to static foveation, dynamic foveation increases MOS in visual quality by 0.60 (on a scale of 1–5) while saving streaming bitrate by 9.81%. Second, we optimize the foveation module in the cloud VR gaming system for even more visually appealing and immersive gaming experiences. By reducing the overhead caused by reconfiguring the foveation parameters, we cut the total latency from 69.15 to 13.36 ms and increase the frame rate from 25.64 to 68.78 Frame Per Second (FPS). Last, another user study on the optimized platform reveals a maximum overall MOS increase of 1.80 and a maximum cybersickness MOS reduction of 1.07, demonstrating its effectiveness and efficiency. Our cloud VR gaming platform is open-source, and could be utilized by researchers and engineers around the globe to further explore the implications of applying foveation to cloud Extended Reality (XR) applications.

中文摘要

雲端虛擬實境（VR）遊戲將計算密集的渲染任務從資源有限的頭戴式顯示器轉移至雲端服務器，以提供高品質的遊戲體驗。然而，串流高畫質遊戲影像需要龐大的頻寬。應對這種高頻寬需求的方法之一是利用人類視覺系統的特性，通過將更高的bitrate分配給頭戴式顯示器視野的中央區域，同時降低周圍區域bitrate的使用，如此一來，能在使用者感受不到畫質降低的情況下，達到減少頻寬的效果，即所謂的視覺焦點（foveation）。雖然視覺焦點已經被應用於遠端虛擬實境遊戲，但現有的雲端遊戲平台都採用靜態視覺焦點（static foveation），假設玩家的凝視位置固定在視野中心。

在本文中，我們首先探討了動態視覺焦點（dynamic foveation），即焦點區域位置會隨著眼睛注視點改變，在雲端虛擬實境遊戲中的可能性。我們建立了第一個支援動態視覺焦點的雲端虛擬實境遊戲平台。在我們的平台中，玩家的眼睛注視點會即時從頭戴式顯示器上的眼動追器傳至雲端服務器，進而調整焦點區域的位置。此外，我們還調整其他視覺焦點參數，如焦點區域大小和周邊區域壓縮率。 透過這個平台，我們進行用戶研究以證明動態視覺焦點的價值，找出最大化遊戲體驗的最佳視覺焦點參數，並以使用者對平台使用體驗的平均評分進行評估。相較於靜態視覺焦點，動態視覺焦點在評分上增加了0.60（區間為1-5分），同時節省了9.81％的bitrate。 接著，我們優化了雲端虛擬實境遊戲平台中的視覺焦點模組，以實現更高品質的遊戲體驗。通過減少重新配置視覺焦點參數造成的運算成本，我們將系統總延遲減少了約50毫秒，實現了接近70幀每秒的幀率。最後，針對優化視覺焦點模塊進行的另一項用戶研究顯示，整體用戶評分增加了1.80，暈眩不適感減少了1.07，展示了優化後平台的效率。 未來，我們計畫開源我們的雲端虛擬實境遊戲平台，供全球的研究人員和工程師使用，進一步探索視覺焦點對雲端擴展實境（XR）應用的影響。

Acknowledgments

I express my sincere gratitude for the successful completion of my master's program over the past two years. During this period, there are many people I would like to thank.

First, my deepest appreciation goes to my advisor, Professor Cheng-Hsin Hsu. He is the most dedicated and earnest professor I have encountered. Every quarter, Professor Hsu consistently assists students in submitting our research to reputable conferences and meticulously refines our English writings. He teaches us how to craft good articles and consistently reminds us of many important things to be careful with when doing research. I am especially grateful for his support in the months leading up to my oral defense. Despite facing numerous challenges and setbacks, Professor Hsu dedicated time every day to review my progress, offer guidance, and steer me in the right research direction, allowing me to complete my research.

Next, I would like to extend my thanks to my lab mate, Kuan-Yu Lee. We collaborated on the NOSSDAV paper with me in the first year, and he generously assisted whenever he had the time. If I struggled to keep up with the workload, he was always willing to help, regardless of the inconvenience. Moreover, as someone prone to anxiety, I found in him an excellent listener when I needed to vent my frustrations. Despite my reluctance to burden him with negativity, he never turned away from my complaints, and I am truly thankful for his understanding and help. I also want to express my gratitude to all my lab mates. Given the experiments of my research requiring numerous participants for VR experiments and feedback, I appreciate everyone's willingness to assist with the experiments, even wearing VR headsets is uncomfortable. Their cooperation has been invaluable, and I am genuinely grateful for their support.

Last, I extend my heartfelt thanks to my family for providing unwavering support, both emotionally and financially, allowing me to focus wholeheartedly on my research. Each return home has been a source of rejuvenation, providing the energy needed to return to school and continue my research.

These two years of graduate studies have not only equipped me with valuable knowledge, skills, and research capabilities in computer science, but have also instilled in me the right attitude and perseverance to handle challenges. I have learned much from Professor Hsu, my peers, and the entire master's program life. While the past two years have been demanding, they have been immensely worthwhile, and I am also grateful for my continuous efforts. The experiences, both positive and negative, will undoubtedly contribute to my overall capabilities as I move forward into the next phase of life.

Contents

List of Figures

List of Tables

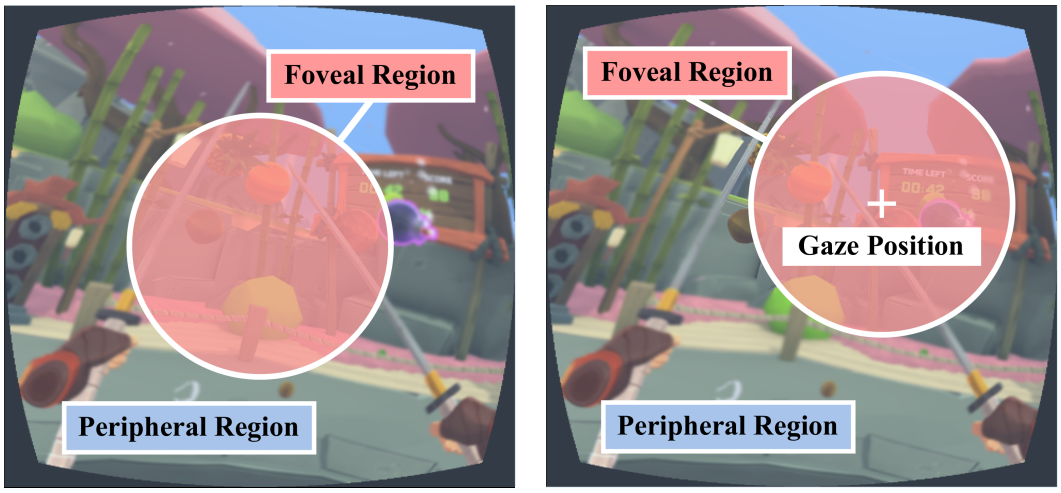
Chapter 1

Introduction

Extended Reality (XR) refers to a wide spectrum of immersive technologies, including Augmented Reality (AR) that enhances the real world by adding digital elements, and Virtual Reality (VR) that immerses users in entirely computer-generated environments. The potential of XR has been well recognized in the industries, as demonstrated by the growing popularity of VR [5] and cloud gaming [7, 51]. In fact, the global cloud gaming market is expected to grow from 691.6 billion USD in 2022 at an annual rate of 45.8% for eight years [46], and the global VR market is anticipated to expand at an annual rate of 15.0% from 21.8 billion USD within the same time duration [47]. Combining the best of both worlds, cloud VR gaming [5, 7] offloads the rendering tasks from Head-Mounted Displays (HMDs) to remote cloud servers, so as to enable a more visually appealing and immersive gaming experience without excessive resource consumption on resource-constrained HMDs.

Providing an immersive cloud VR gaming experience, however, dictates high bandwidth demands, e.g., interactive cloud VR applications require per-user bitrate of 40 and 90 Mbps for acceptable and comfortable experiences, respectively [21]. Service providers, therefore, run into the following dilemma: on the one hand, they have to subscribe to or deploy enough Internet bandwidth to maintain the gaming Quality of Experience (QoE) for customer retention; on the other hand, doing so could hinder their profitability. *Hence, service providers must develop a way to reduce the bitrate consumption while maintaining high gaming QoE.*

One possible solution for service providers is to strategically allocate available bitrate to different regions of HMD viewports in order to leverage the Human Visual System (HVS) that has spatially non-uniform sensitivity. Fig. 1.1 illustrates that each viewport can be divided into: (i) foveal and (ii) peripheral regions. HVS has high acuity in the foveal region, within it the acuity decreases from the gaze position to the periphery. Moreover, HVS gradually becomes less sensitive to color and depth across the peripheral region



(a) (b)

Figure 1.1: Each HMD viewport can be divided into foveal and peripheral regions. Sample screenshots captured from a VR game with: (a) static and (b) dynamic foveation.

[16, 62]. By allocating relatively more bitrate to the foveal region, we may improve its perceived visual quality. Meanwhile, the degraded visual quality in the peripheral region may not be noticeable, leading to better overall gaming QoE. Such an *unequal rate allocation* approach is referred to as *foveation* [32]. Fig. 1.1 also shows two options when applying foveation to cloud VR gaming: *static foveation*, in which the gaze position (and thus the foveal region) of the user is fixed at the center of the HMD viewport, or *dynamic foveation*, in which the gaze position is determined by eye trackers on the HMD in real-time. Intuitively, dynamic foveation "should" achieve better QoE, although none of the existing cloud VR gaming platforms support that at the time of writing.

1.1 Contributions

The lack of dynamic-foveation-enabled cloud VR gaming platforms is understandable because incorporating dynamic foveation with cloud VR gaming is no easy task. In particular, cloud VR gamers demand the highest possible gaming QoE, which depends on a rich set of influencing factors related to humans, systems, and contexts [41, Ch. 4] that are not well understood by the research community. In this thesis, we set out to answer the following three key Research Questions (RQs) about dynamic foveation in cloud VR gaming:

- **(RQ1) Does dynamic foveation boost cloud VR gaming experience?** Although not in VR games, Illahi et al. [24] reported that gamers' gaze positions are rather fixed at the viewport center in some (such as first-person) games, compared to

other (such as god-view) games. In VR games, would it be worth realizing dynamic foveation for good gaming experiences? Moreover, the utilization of dynamic foveation introduces the need for critical decisions in fine-tuning various foveation parameters, such as foveal region size, to enhance gamers' gaming experience. Ill-selected foveation parameters could back-fire, making the gaming experience intolerable.

- **(RQ2) How to effectively support dynamic foveation in cloud VR gaming?** Updating foveation parameters based on gaze positions in real-time leads to additional, non-trivial overhead to the cloud VR gaming server and client, which may degrade the gaming experience. To minimize latency and maintain an acceptable frame rate, we must streamline the foveation reconfiguration workflow. Furthermore, to make our cloud VR gaming platform flexible, different foveation approaches should be incorporated in a modularized fashion.
- **(RQ3) How much QoE improvement can we achieve after optimizing the platform?** Subjective evaluations on our dynamic-foveation-enabled cloud VR gaming platform are needed to validate the effectiveness of our optimization efforts. Such evaluations involve time-consuming user studies to quantify various QoE aspects such as visual quality, immersive level, and cybersickness in Mean Opinion Score (MOS).

Existing remote VR gaming projects, such as the Nvidia CloudXR [42] or open-source Air Light XR (ALXR) [2], only support static foveation at best. Multiple technical difficulties arise when adding dynamic foveation to cloud VR gaming, such as ensuring per-frame consistency of foveation parameters between the server and client and minimizing foveation reconfiguration overhead, which are largely due to the highly real-time nature of cloud VR gaming. Throughout this thesis, we strive to address the three RQs in the following steps:

- We design and implement a first cloud VR gaming platform with dynamic foveation supports, which involves integrating an eye-tracking module and devising a dynamic foveation mechanism to synchronize foveation parameters between the server and client. *This is a first open-source cloud VR gaming platform [3] that supports dynamic foveation.*
- We conduct subjective evaluations to prove the value of dynamic foveation and make recommendations for optimal foveation parameters. Our findings indicate that adopting dynamic foveation increases the MOS in overall quality by 0.60 (on a scale of 1–5) while reducing the bitrate by 9.81%, compared to static foveation. *In Sec. 5, we answer our RQ1 by validating the value of dynamic foveation with optimal foveation parameters.* Some initial results were reported in our conference

paper [11].

- We optimize the foveation module in our cloud VR gaming platform, proposing better reconfiguration workflow and implementing an alternative foveation approach. The objective measurements show that our optimization efforts lead to a reduction in the total latency from 69.15 to 13.36 ms, and increase the frame rate from 25.64 to 68.78 Frame Per Second (FPS). *In Sec. 6, we answer our RQ2 by presenting our optimization efforts on the dynamic-foveation-enabled cloud VR gaming platform.*
- We conduct further subjective evaluations to assess the performance of the optimized platform. The experimental results reveal that, compared to the original foveation module, the optimized one results in a maximum MOS increase of 1.80 in overall quality and a maximum MOS reduction of 1.07 in cybersickness. *In Sec. 7, we answer our RQ3 by demonstrating significant subjective quality improvement achieved after optimizing our platform.*

1.2 Organizations

The thesis is structured as follows. Sec. 3 provides the related works relevant to our thesis. In Sec. 4, we offer a comprehensive overview of the system and the foveated warping approach we employed. We demonstrate the implementation of an authentic real-time cloud VR gaming system and the corresponding subjective evaluations in Sec. 5. Sec. 6 illustrates the foveation module optimization in the system. This is followed by an additional subjective evaluation assessing the performance of the optimized system in Sec. 7. Last, we conclude this thesis and discuss future work in Sec. 8.

Chapter 2

Background

In this chapter, we introduce the related background knowledge from four aspects: cloud VR gaming, human visual system, foveated streaming, and quality of experience.

2.1 Cloud VR Gaming

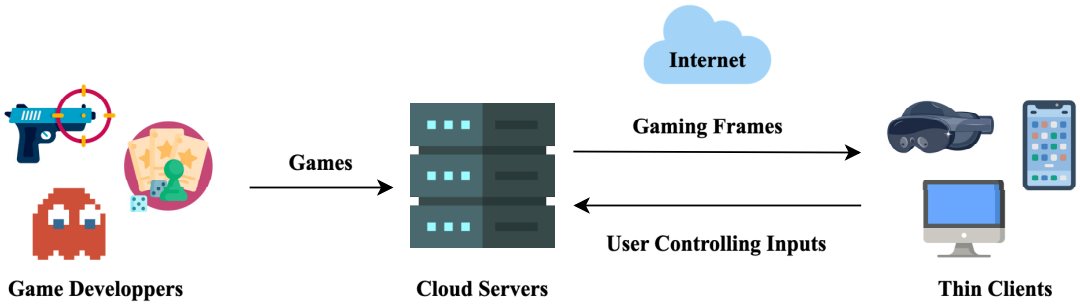


Figure 2.1: A typical cloud gaming system.

Cloud gaming [51], a revolutionary concept in the gaming industry, redefines the gaming experience by leveraging cloud computing technology. This innovative approach splits the gaming system between a cloud server and a local thin client, offering players the flexibility to enjoy their favorite games on various devices. Fig. 2.1 demonstrates the typical cloud gaming system. Generally, the cloud server, hosted on a GPU-equipped cloud or edge server, executes the games from the game developers, captures and encodes gaming frames, and transmits them to the client. The client, situated on low-end devices such as mobile phones, receives, decodes, and renders the gaming frames on the screen. Meanwhile, the gamer's controlling inputs, e.g., head positions and keyboard events, are streamed back to the cloud server. A distinct advantage of cloud gaming is its compatibility with off-the-shelf games, enabling high-quality gaming experiences even on resource-constrained end devices. Various approaches, such as dynamically sharing

rendering load between the server and client [8] or employing novel video object transmission methods [40], further enhance the cloud gaming landscape.

In terms of cloud "VR" gaming, a captivating fusion of virtual reality and cloud gaming technology unfolds. Game developers create immersive VR games, which are carefully curated by cloud VR gaming service providers to operate on cloud servers for VR gamers. These experiences unfold in real-time, with rendered gaming frames seamlessly transmitted to VR gamers' Head-Mounted Displays (HMDs). Simultaneously, gamer interactions, captured through HMDs and controllers, are compressed and streamed back to the server, completing the immersive loop. However, the dynamic and diverse access networks of VR gamers pose challenges for cloud VR gaming service providers in delivering consistent and immersive experiences. Achieving optimal gaming experience demands addressing issues of latency and bandwidth, ensuring that gamers enjoy seamless and high-quality VR experience. As cloud VR gaming continues to evolve, ongoing research and development aim to refine these technologies and unlock new frontiers in the world of VR gaming.

2.2 Human Visual System (HVS)

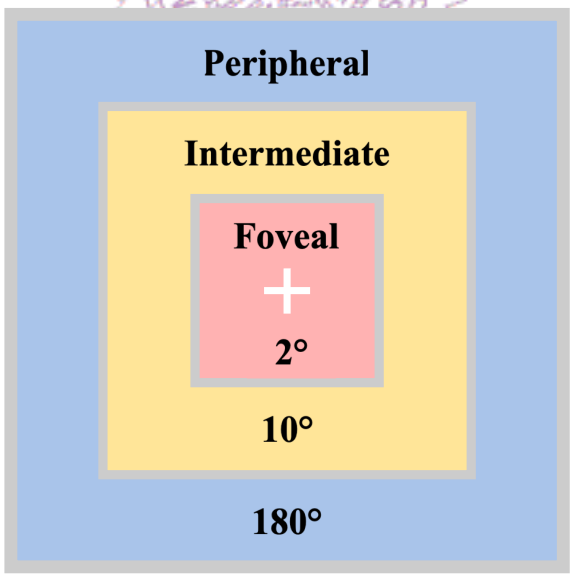


Figure 2.2: Defined regions in the human visual field.

The Human Visual System (HVS) is a marvel of biological engineering, featuring a sophisticated response to visual stimuli driven by the phenomenon of *foveation*. This phenomenon emerges from the intricacies of photoreceptor cell distribution, where rods and cones each contribute to low-illumination and high-illumination vision [62]. An intrinsic characteristic of the HVS lies in the non-uniform distribution of cone density, reaching its

pinnacle at the fovea—a small region within a 2° span in the human visual field [49]. The density undergoes a sharp decline as we move away from the fovea, sculpting the sampling response and exerting a profound influence on the perceived resolution of vision. As shown in Fig. 2.2, these regions are commonly defined as foveal, intermediate (also known as parafoveal), and peripheral regions, corresponding to 2°, 10°, and 180° eccentricity, respectively [59, Ch. 1]. The HVS is a dynamic system that adapts to varying conditions, optimizing its performance based on factors such as spatial frequency, contrast, and luminance [65]. This adaptability allows the visual system to efficiently allocate its resources, focusing on critical details within the foveal region while maintaining a broader awareness of the peripheral surroundings.

Additionally, the HVS incorporates intricate neural mechanisms for visual processing, involving complex pathways and interactions between different regions of the brain [14]. These mechanisms contribute to the brain's ability to interpret visual information, distinguish patterns, and make rapid decisions based on the visual input received. In terms of technological advancements, understanding the meanings of the HVS becomes paramount for designing effective visual experiences. Techniques that leverage the principles of foveation and align with the adaptive nature of the HVS can significantly enhance the efficiency and realism of visual displays, particularly in virtual and augmented reality applications.

2.3 Foveated Streaming

Figure 2.3: Sample of foveated streaming.

Regarding real-time streaming, achieving an optimal balance between video quality and available bandwidth is a perpetual challenge. Traditional adaptive bitrate streaming

has conventionally modulated video quality in response to temporal bandwidth variations. The evolution of streaming technologies welcomes a transformative approach *foveated streaming* that introduces a pioneering layer of spatial rate adaptation within individual frames. As illustrated in Fig. 2.3, this innovative streaming paradigm utilizes the concept of HVS by encoding the highest video quality precisely at the gamer's gaze positions, while allocating lower quality elsewhere in the frame. By doing so, we can conserve more bandwidth usage, while gamers may not readily perceive the quality degradation, as they are less sensitive to changes in the peripheral region. The synergy of spatial and temporal rate adaptation within the foveated streaming framework promises not only significant enhancements in streaming efficiency but also a holistic improvement in the overall Quality of Experience (QoE) for gamers.

While the concept of foveated streaming has intrigued researchers for some time, its widespread implementation has been hindered by the need for individual gaze information. Determining gaze positions can be accomplished through pre-analyzing video content for salient features [28, 64] or utilizing real-time gaze tracking technologies. Current approaches often involve partitioning the video frame into tiles, strategically streaming high-quality tiles at the gaze positions and lower resolution tiles elsewhere [49, 68]. Notably, recent advancements in affordable and non-invasive gaze-tracking solutions have sparked renewed interest in the realization of real-time gaze-based foveated streaming, making it a compelling area for elevating video streaming efficiency and optimizing the gamers' experiences.

2.4 Quality of Experience (QoE)

Figure 2.4: The influencial factors of QoE.

Quality of Experience (QoE) serves as a multifaceted metric encompassing the overall satisfaction and perception of gamers while engaging with a particular system or service. Unlike traditional Quality of Service (QoS), which primarily focuses on technical

performance parameters such as bandwidth and delay, QoE studies the subjective and experiential aspects of gamer interaction. It encapsulates gamers' perceptions of the quality, usability, and enjoyment derived from their interactions with a product, service, or application. QoE is inherently user-centric, emphasizing individual experiences and preferences. It is shaped by various factors as illustrated in Fig. 2.4, including "humans" influence factors that encompass the gamers' demographic and socioeconomic background, physical and mental structure, and emotional state; "systems" influence factors that take into account content-related, media-related, network-related, and device-related aspects; and "contexts" influence factors that consider any situational influences describing the gamers' environments [41, Ch. 4].

Evaluating QoE poses inherent challenges, as it involves conducting user studies for gamers to subjectively rate their experience. This approach differs from objective computational methods like PSNR, SSIM [63], and VMAF [26], which are traditionally used for video quality assessment. However, the current system prioritizes the subjective experience of end-gamers, believing that their feedback provides the most accurate reflection of overall QoE [22]. Numerous studies have explored QoE in terms of foveated streaming [13, 24, 29], emphasizing distinct parameters based on their specific settings. The sizes, resolutions, and coding parameters at different regions of the frames play important roles in reducing bandwidth requirements for high-quality streaming. Notably, Round-Trip Time (RTT), or end-to-end latency, was also identified as a key constraint. Through gamer studies, we can explore various coding parameters and network properties, thus concluding that acceptable QoE could be achieved with proper parameterization.

In terms of cloud VR gaming, QoE considerations extend to factors like perceived visual quality, seamless delivery of VR content, immersion, interaction, and cybersickness [58]. QoE plays a pivotal role in shaping gamer satisfaction, loyalty, and the success of interactive systems, making it a crucial measure in the evaluation and enhancement of user-centric technologies.

Chapter 3

Related Work

In this chapter, we survey cloud streaming systems from four aspects: remote rendering, unequal rate allocation, gaze-driven adaptation, and subjective evaluations with foveation.

3.1 Remote Rendering

Remote rendering, which offloads the rendering tasks from local clients to remote servers, has been considered in various interactive multimedia applications [54]. For example, Shi et al. [55] introduced a proxy-based remote rendering framework designed for efficient 3D video rendering on mobile devices. Shi et al. [56] presented an advanced low-latency remote rendering system that aims to assist mobile devices in rendering interactive 3D graphics in real-time. Remote rendering has been leveraged by cloud gaming, e.g., Huang et al. [20] proposed the very first cloud gaming platform called *GamingAnywhere*. Cai et al. [8] implemented a component-based cloud gaming system that dynamically allocates rendering loads between the server and client. Kämäräinen et al. [31] built a mobile cloud gaming prototype to dissect the delays in mobile devices and various network conditions. Lee et al. [33] built a cloud VR gaming testbed to conduct user studies for QoE modeling. Readers who are interested in cloud gaming literature are referred to relevant surveys [6, 9, 39]. *Although the above-mentioned remote rendering systems [8, 20, 55, 56] achieved high performance without increasing network loads, they did not employ foveation.*

Foveation has been applied to some remote rendering systems, e.g., Illahi et al. [23,24] proposed a cloud gaming foveation prototype that requires no modifications to the underlying game engine. Ryoo et al. [49] developed a streaming system employing foveated encoding driven by gaze positions. *Although the above works [23, 24, 49] constructed cloud streaming systems enhanced by foveation, they focused on traditional, non-VR content.* Foveation has also been adopted in a few VR projects. For example, Romero-Rondón et al. [48] presented a foveated encoding system that streams 360° videos to HMDs. More-

over, Nivida has released an SDK, called CloudXR [42], for streaming XR contents from remote servers, which supports static foveation. However, *none of these studies [42, 48] realized a real-time remote rendering VR system with dynamic-foveation supports.*

3.2 Unequal Rate Allocation

Unequal rate allocation based on diverse definitions of Region-of-interest (ROI) has been done in the literature. Relevant studies can be classified into: *content-aware*, *object-aware*, and *foveation* based on how they defined ROI. Hegazy et al. [17] proposed a content-aware video encoding method to allocate different amounts of bitrate to multiple regions leveraging in-game ROIs. Mohammadi et al. [40] introduced an object-aware method based on gamer visual attention in cloud gaming. Zou et al. [69] proposed an object-aware video encoding method, which sets the quantization parameters based on objects intersecting with HMD gaze positions. *Both content- and object-aware methods [17, 40, 69] dictate augmenting the source code of VR applications to retrieve ROIs. Such a* white-box *approach incurs higher engineering overhead and may drive service providers away.*

In contrast, foveation could be realized in a *black-box* approach. For example, Shen et al. [53] introduced a new image transmission scheme that leverages the foveation characteristic and analog coding techniques to achieve higher perceptual visual quality. Wang et al. [64] proposed a foveation scalable video coding algorithm that offers both good quality-compression performance and effective rate scalability. Illahi et al. [24] assessed the impact of different foveated encoding parameters on visual quality and total latency using their cloud gaming system. The results suggested that it is feasible to identify a "sweet spot" for the encoding parameters, where users may hardly notice the effect of foveated encoding under reduced bandwidth requirements. Illahi et al. [25] took a step further and analyzed the performance of different foveation approachesdriven by eye trackers. *Compared to our work, these foveation systems [24, 25, 53, 64] were built for traditional contents on desktop computers, rather than VR contents in HMDs.*

Illahi et al. [25] identified three foveation approachessuitable for cloud VR systems:

- *Foveated rendering* reduces the rendering quality of the peripheral region by, e.g., downsampling the 3D meshes therein.
- *Foveated encoding* adjusts the video encoding parameters, such as quantization parameters, of the peripheral region.
- *Foveated warping* downsamples the peripheral region of HMD viewports before encoding the whole viewports, so that a higher bitrate is used by the foveal region.

Among these three methods, foveated rendering aims to reduce computational complex-

ity, while foveated encoding and foveated warping intend to decrease bandwidth consumption. Recently, foveated warping has been found to be more efficient due to fewer pixels, which results in smaller resolution and shorter latency for a given rendered resolution [25]. *Hence, we adopt foveated warping throughout this thesis.*

3.3 Gaze-driven Adaptation

Gaze-driven adaptation of unequal rate allocation has been under active research. Patil et al. [43] proposed a gaze-aware video streaming system for mobile devices, which selectively streams audio/video content based on gaze positions. Frieß et al. [13] adjusted the encoding parameters of individual macro-blocks dynamically based on gaze positions on a large display, which demonstrates a reduction of required bandwidth and an increase of visual quality in foveal regions. Zare et al. [68] stored the video content at two different resolutions, each divided into HEVC-compliant tiles that can be encoded and decoded independently. According to the current viewport, a set of tiles is transmitted in high resolution, while the remaining tiles are transmitted in low resolution. Similarly, Ryoo et al. [49] developed a foveated streaming approach employing multi-resolution video coding to encode the video in multiple copies for different resolutions. Based on real-time webcam-based gaze tracking, they stream tiles around the gaze region in high resolution only.

Besides traditional video streaming, this technique has also been applied to VR content. Firdose et al. [12] demonstrated a 360° video streaming method that utilizes eye trackers to deliver high-quality VR content around gaze positions only. Chen et al. [10] assessed perceptual importance in 2D image space by analyzing gaze behaviors. They then mapped the importance of 3D object space to determine streaming priorities for rendering. Lungaro et al. [35] presented an innovative content delivery approach for VR video streaming that leverages gaze positions for eye-tracking-capable HMDs. Compared to traditional approaches, they reduced the consumed bandwidth yet delivered high QoE levels to HMD users. *While the aforementioned works [10, 12, 13, 35, 43, 49, 68] proposed solutions leveraging gaze positions to realize unequal rate allocation, our work takes a step forward by realizing an interactive cloud VR platform with optimal dynamic foveation, which is much more challenging than one-way video streaming.*

3.4 Subjective Evaluations with Foveation

Subjective evaluations with foveation have been carried out in the literature. For example, Hsu et al. [19] proposed a framework to compare different subjective metrics in remote

foveated rendering systems. Moreover, they proposed a regression model to characterize the relationship between the human perceived quality and foveated rendering parameters. Illahi et al. [25] performed a small-scale user study to evaluate the subjective video quality of foveated encoding and warping. Lungaro et al. [36] conducted user studies to investigate the design space for foveated content provision under different network Round-Trip Times (RTTs), resolutions, and foveated radii in video streaming services. Ryoo et al. [49] presented a comprehensive user study on their multi-resolution video coding approach, demonstrating a bandwidth reduction of a factor of 2 while maintaining the same level of user satisfaction in their foveated video streaming system. Jin et al. [29] compiled a 2D and 3D video compression dataset for quality assessment research on foveated encoding. They varied two parameters: the quantization parameter and foveal region size and considered different objective assessment methods. Patney et al. [44] designed a user study on a foveated rendering system using eye-tracking-capable HMDs and monitors to evaluate the HVS acuity. Hsiao et al. [18] conducted a user study on their foveated encoding system to investigate the implications of total latency. They found that if the total latency is smaller than 15 ms, the same QoE can be achieved with one-fifth of bandwidth consumption. Albert et al. [4] performed user studies to assess the detectability of visual artifacts across different foveated methods and different radii of the foveal region. They reported that a total latency of 50 to 70 ms is tolerable for foveated rendering in VR applications. *Different from the current work, none of the aforementioned studies [4, 18, 19, 25, 29, 36, 44, 49] conducted subjective evaluations with HMDs equipped with eye trackers to assess the performance of dynamic foveation in cloud VR gaming.*

Chapter 4

System Overview

In this chapter, we first discuss the unique properties of cloud VR gaming platforms. Then, we present the fundamental components of a cloud VR gaming platform and explain their interplay. This is followed by the introduction of the foveated warping approach employed in an open-source cloud VR gaming system [2].

4.1 Cloud VR Gaming Systems

Table 4.1: Systems Related to Cloud VR Gaming

Property / System	Bidirectional	Cloud Rendering	Extended Reality	Quality Sensitive	Latency Sensitive	Bandwidth Sensitive
360° Video Streaming	○	○	●	◐	○	◐
VR Teleconferencing	●	○	●	◐	◐	◐
AR Rendering	○	●	●	○	●	○
Cloud Gaming	●	●	○	◐	●	◐
Cloud VR Gaming	●	●	●	●	●	●

Table 4.2: Notations of Table 4.1

○	◐	●
No	Partially Yes	Yes

Table 4.1 compares relevant to cloud VR gaming, where ●, ◐, and ○ represent *yes*, *partially yes*, and *no*, as defined in Table 4.2. 360° video streaming utilize cloud infrastructure to offer Video-on-Demand services. As it is one-way streaming and spherical 360° videos are easier to compress compared to 3D content, 360° video streaming is not latency-sensitive and relatively less quality- and bandwidth-sensitive. VR teleconferencing and AR rendering leverage cloud resources for bidirectional communications

and rendering, while both incorporating XR technologies. VR teleconferencing has balanced demands for quality, latency, and bandwidth, and AR rendering dictates very low latency for immediate application responsiveness. Like cloud VR gaming, (traditional) cloud gaming relies on cloud infrastructure, bidirectional interactions, and thus is highly sensitive to latency. It, however, is less quality- and bandwidth-sensitive than cloud VR gaming consumed by HMDs. Among these systems, cloud VR gaming exhibits the highest demands on quality, latency, and bandwidth. In this article, we study the most challenging cloud VR gaming systems. Adding to that, we aim to enable dynamic foveation in cloud VR gaming systems, which impose even more stringent requirements. Through solving various challenges in such systems, our developed solutions may also be applied to less-demanding systems in Table 4.1.

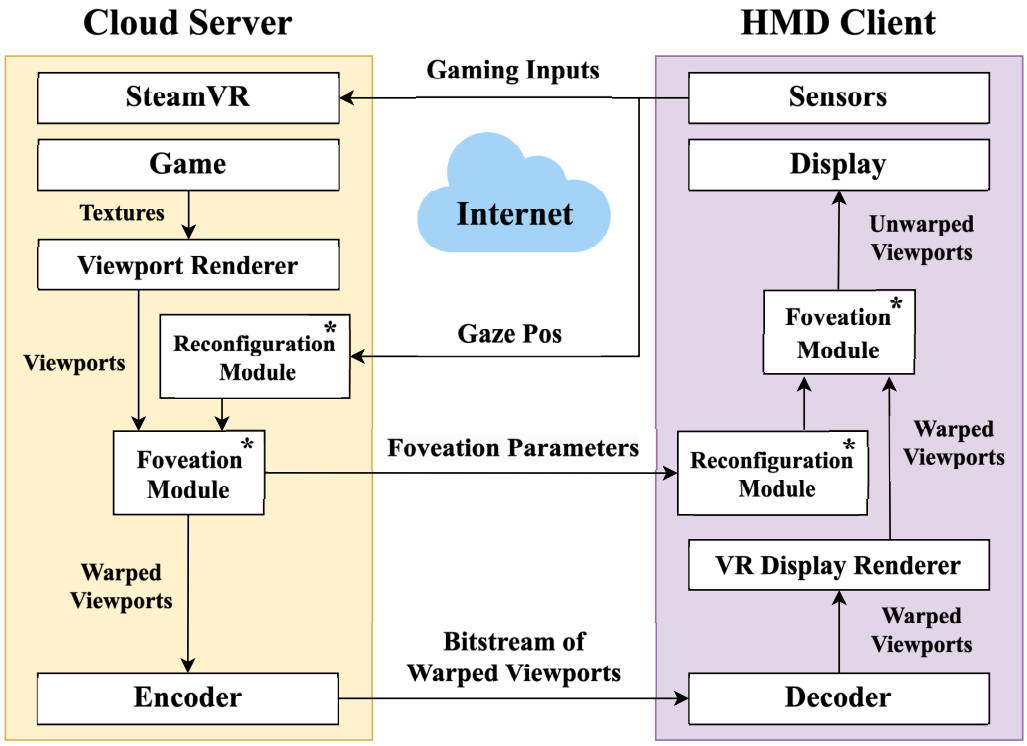


Figure 4.1: Our cloud VR gaming platform that supports dynamic foveation.

Fig. 4.1 gives a block diagram of our cloud VR gaming platform. The platform consists of a cloud server and an HMD client. The cloud server sends a sequence of warped viewport frames along with corresponding foveation parameters to the HMD client. The HMD client in turn sends a sequence of gaming inputs, including head positions/orientations, controller keystrokes, and gaze positions back to the server. In addition to SteamVR [61] engine and VR game, the server also encompasses components including: (i) a viewport renderer that renders the textures captured from SteamVR engine, (ii) a reconfiguration module that takes new gaze positions to initialize/adjust the foveation module and sends the new foveation parameters to the client, (iii) a foveation module

that warps the viewport frames from the renderer, and (iv) an encoder that encodes the warped viewport frames into a bitstream. The client consists of similar components in the opposite direction including: (i) a decoder that decodes the bitstream back to warped viewport frames, (ii) a VR display renderer that renders the warped viewport frames, (iii) a reconfiguration module that takes the corresponding foveation parameters received from the server as input to initialize/adjust the foveation module, and (iv) a foveation module that unwarps the viewport frames from the VR display renderer.

The server and client interact as follows. A session is initiated once a client connects to a server, which launches a VR game supported by SteamVR engine. Gaming inputs are then captured by the client and streamed back to the server. The server then forwards these gaming inputs to the SteamVR game. If dynamic foveation is enabled, the client also needs to capture and stream the gaze positions to the server during the gamer's playing time. Depending on the gaming inputs from the client, the server first reconfigures the foveation module to adopt the new position of the foveal region. After the reconfiguration, the server proceeds to render, warp, encode, and stream the viewport frames to the client. Meanwhile, upon receiving the viewport frames from the server, the client follows a similar but reversed procedure with consistent foveation parameters to execute the corresponding decoding, reconfiguring, rendering, and unwarping, and finally displays the viewport frames to the gamers.

Following Fig. 4.1, we have implemented our dynamic-foveation-enabled cloud VR gaming platform [3] on top of the open-source project ALXR [2]. ALXR server only discovers clients on the same LAN, preventing it from being used as a cloud VR gaming system over the Internet. We modified the ALXR, so that a client initiates a session with a user-specified cloud server IP address. More importantly, we enhance ALXR by adding optimization modules: foveation and reconfiguration modules, which are annotated with asterisks in Fig. 4.1. We describe the foveated warping approach adopted by vanilla ALXR [2] in the next section.

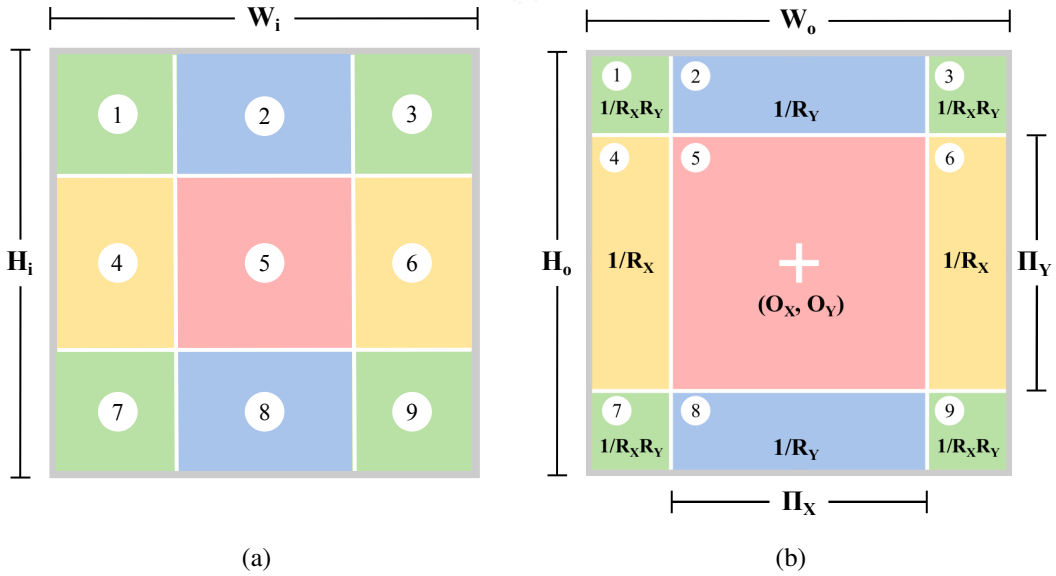
4.2 Foveated Warping

Foveated warping aims to reduce the number of pixels only in the peripheral region of each frame, so that more bits can be used to compress the foveal region for higher quality. Fig. 4.2 shows sample original, warped, and unwarped viewport frames with zoom-in boxes, demonstrating the warping shape distortion due to a decrease in resolution of the peripheral region compared to the original frame. We first present a popular foveated warping approach, called *Axis-Aligned Distortion Transmission (AADT) [1]*, which is a post-processing operation used in Oculus Link [37] after a frame is rendered. Warping

(a) (b) (c)

Figure 4.2: Sample: (a) original, (b) warped, and (c) unwarped viewport frames.

originates from the operation of general inverse barrel distortion in order to compensate for the optical-pincushion distortion [67] caused by lenses used by HMDs. Earlier Oculus HMDs are less computationally capable and require such warping operations to be done by Oculus PC Runtime running on tethered PCs. While more recent Oculus HMDs could perform such compensation themselves, the warping operation in Oculus PC Runtime can be leveraged for implementing AADT for foveated warping to unequally allocate bitrate. The AADT-warped, instead of the original rectilinear frames, are transmitted from a PC to the tethered HMD. Upon receiving the AADT-warped frames, the HMD unwarps them back to the original ones.



(a) (b)

Figure 4.3: AADT Warp: (a) original and (b) warped viewport frames.

We note that there exist several variants of AADT warping approaches in the literature [30]. For the sake of discussion, we describe the version implemented by ALXR and

refer to it as *AADT Warp* throughout the article. Fig. 4.3 presents the viewport frames before and after AADT warping. The original viewport frame is divided into a rectangular foveal region and nine peripheral (sub)regions, enumerated in 1–9. The core warping idea is to "compress" the pixel density in the peripheral region in a non-linear fashion [50], so that the pixel density drops increasingly more when approaching the viewport edge. We let W_i and H_i be the width and height of the original viewport frame, and W_o and H_o be those of the warped viewport frame. This foveated warping approach is parameterized as follows:

- *Foveal region size* is specified by the width and height of a rectangular foveal region. Its width Π_X and height Π_Y are normalized to those of the whole viewport frame, i.e., $\Pi_X, \Pi_Y \in [0, 1]$.
- *Compression ratios* R_X and R_Y control the degree of "compression" along the x- and y-axes, where R_X, $R_Y \in [1, 10]$. That is, R_X and R_Y control the pixel density in the peripheral region, e.g., with $R_X = 5$ the vertical pixel density in the top and bottom peripheral subregions (1–3 and 7–9 in Fig. 4.3) become one-fifth of that in the original viewport frame.
- *Foveal region center* is specified by the coordinates O_X and O_Y, with values determined by the gaze position $\in [-1, 1]$. When the foveal region is at the viewport center, we have $O_X = O_Y = 0$. Positive O_X, O_Y values shift the foveal region towards right and down, while negative values shift the foveal region towards left and top.

Given parameters, the ALXR server adopts Direct3D shaders to create the warped viewport frame, which is encoded by a video encoder. At the ALXR client, the decoded warped viewport frame is unwarped by Vulkan shaders for the original viewport frame. Note that the warped viewport frame has a reduced resolution compared to the original one, i.e.:

$$W_o = W_i \times [\Pi_X + (1 - \Pi_X)/R_X]; \tag{4.1}$$

$$H_o = H_i \times [\Pi_Y + (1 - \Pi_Y)/R_Y], \tag{4.2}$$

which devotes more bits to the foveal region at the same encoding bitrate. Fig. 4.4 summarizes the changes in frame resolution throughout the whole streaming process. Here, the server calculates W_o and H_o and passes these values to the encoder. Subsequently, the encoder encodes the viewport frame with W_o and H_o. On the client side, the client decodes and unwarps the viewport frame from $W_o \times H_o$ back to $W_i \times H_i$. Finally, the client displays the frame in $W_i \times H_i$ on the HMD.

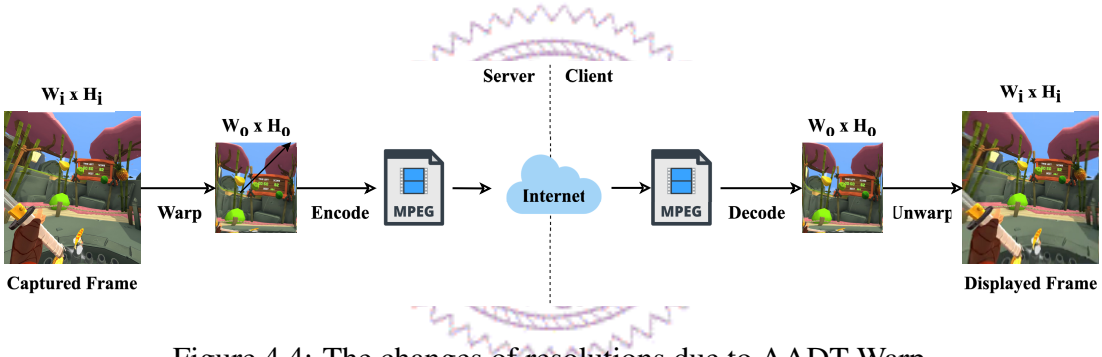


Figure 4.4: The changes of resolutions due to AADT Warp.

Chapter 5

Dynamic Foveation in a Cloud VR Gaming Platform

In this chapter, we present our initial realization of dynamic foveation using ALXR [2]. Moreover, we provide preliminary subjective evaluations to validate the efficacy of dynamic foveation and offer recommendations on foveation parameters. Last, we discuss the system limitations based on objective measurements.

5.1 Dynamic Foveation Mechanism

Introducing dynamic foveation to ALXR is challenging for several reasons: (i) the ALXR client lacks eye-tracker supports, (ii) the ALXR server only allows gamers to change foveation parameters before a session begins, and (iii) the ALXR server/client only implements less-complicated static foveation. In dynamic foveation, the foveation parameters used by the server and client need to be consistent at the frame level, which further complicates our tasks at hand. We have enhanced the ALXR project in the following to enable dynamic foveation.

Eye-tracker supports through OpenXR APIs. We invoke the head- and eye-tracking APIs provided by the official OpenXR SDK [38] at the client to get the gamer's head positions/orientations and gaze positions. We then send the computed foveal region positions back to the server for adapting the foveal region on the fly.

Reconfiguration supports from the renderers. If foveation is enabled, the server and client will undergo an additional warping/unwarping process in the foveation module. To enable dynamic foveation during a gaming session, we add reconfiguration modules before the foveation modules on the server and client. In addition, the ALXR server keeps monitoring any changes of foveation parameters from: (i) the client's gaze position and (ii) the server's dashboard. The former changes are caused by HMD gamer movements,

while the latter changes are due to manual configurations. Once the foveation parameters are changed, the server passes the new parameters to both the server and clients' reconfiguration modules for rendering the upcoming viewport frames.

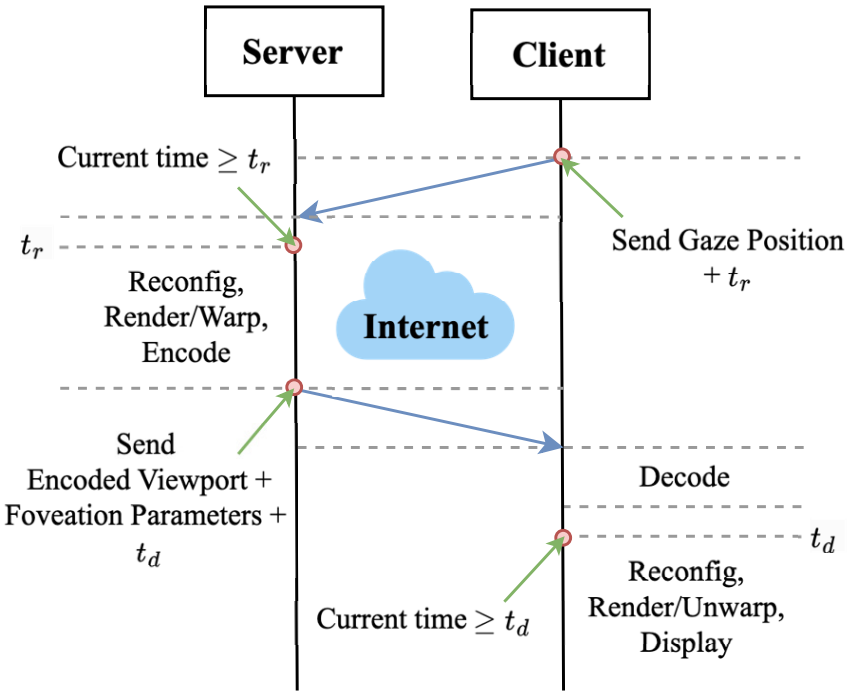


Figure 5.1: The timeline of adapting foveal region with dynamic gaze positions.

Mechanism to dynamically adapt foveal region. We develop a way to match the foveation parameters used by individual frames at the ALXR server and client. This is crucial to avoid any shape distortion due to inconsistent parameters. As illustrated in Fig. 5.1, we program: (i) the ALXR client to periodically upload the current gaze positions along with a predicted rendering timestamp called `renderingTime`, denoted as t_r, to the ALXR server, (ii) the ALXR server waits until the current time right passes t_r, reconfigures the foveation module, renders/warps the viewport frame, and sends the encoded frame with the foveation parameters and an anticipated display timestamp called `displayTime`, denoted as t_d, to the client, and (iii) the ALXR client waits until the current time right passes t_d and then decodes the viewport frame, reconfigures the foveation module, renders/unwarps, and displays the viewport frame. Here, t_r and t_d are computed by the network and system latency of the recent viewport frames, so that neither the ALXR server nor client are overloaded. The resulting cloud VR gaming platform was initially presented in our preliminary conference version [11], and serves as the baseline for our optimization efforts reported in the rest of this article.

5.2 Validation for the Efficacy of Dynamic Foveation

We conduct a user study using our cloud VR gaming platform for two reasons. First, AADT Warp comes with several foveation parameters, which need to be carefully chosen through user study. Second, we compare no foveation, static foveation, and dynamic foveation on the QoE of cloud VR gaming.

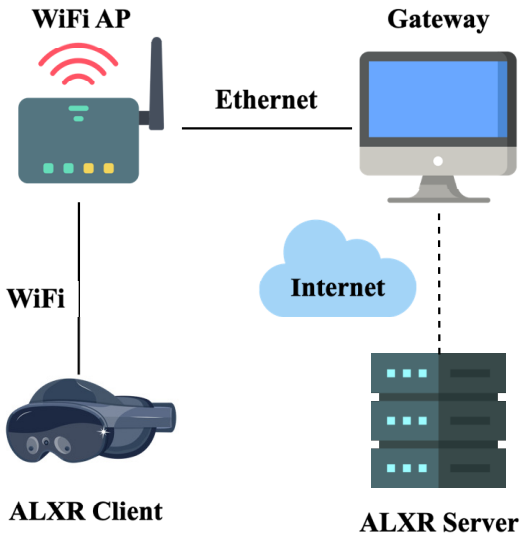


Figure 5.2: Our developed cloud VR gaming testbed topology.

5.2.1 Setup

Fig. 5.2 and 5.3 shows the topology and photo of our testbed. We installed our cloud server on a Windows 10 PC with an Intel Core i9 CPU (@ 3.50 GHz), 64 GB RAM, and an Nvidia RTX 3080 Ti GPU; and our HMD client on a Meta Quest Pro with a Qualcomm Snapdragon XR2+ CPU (@ up to 2.84 GHz), 12 GB RAM, and Adreno 650 GPU, along with ten sensors for eye/face tracking. The network for the HMD client is provided by the WiFi AP, which is connected to a gateway via Ethernet. Note that we have employed FreeBSD 13.1 as a gateway that connects to the Internet. ALXR uses an Nvidia H.264 video encoder with its Constant Bit Rate (CBR) rate controller. We set the target bitrate to 5 Mbps, which is available in most commodity broadband Internet access. Additionally, the target eye resolution is fixed at 1184×1056, with the server generating frames at a rate of 72 FPS. We reconfigure the foveal region at 10 Hz. We run *Fruit Ninja VR 2* in the experiments. Fruit Ninja is an action game requiring gamers to focus on fruits flying in the air from different directions, and then slash the fruits with a sword. We chose this game to be conservative because HMD gamers must move their eye gaze frequently to: (i) search for special items for bonus points and (ii) check the remaining time and score

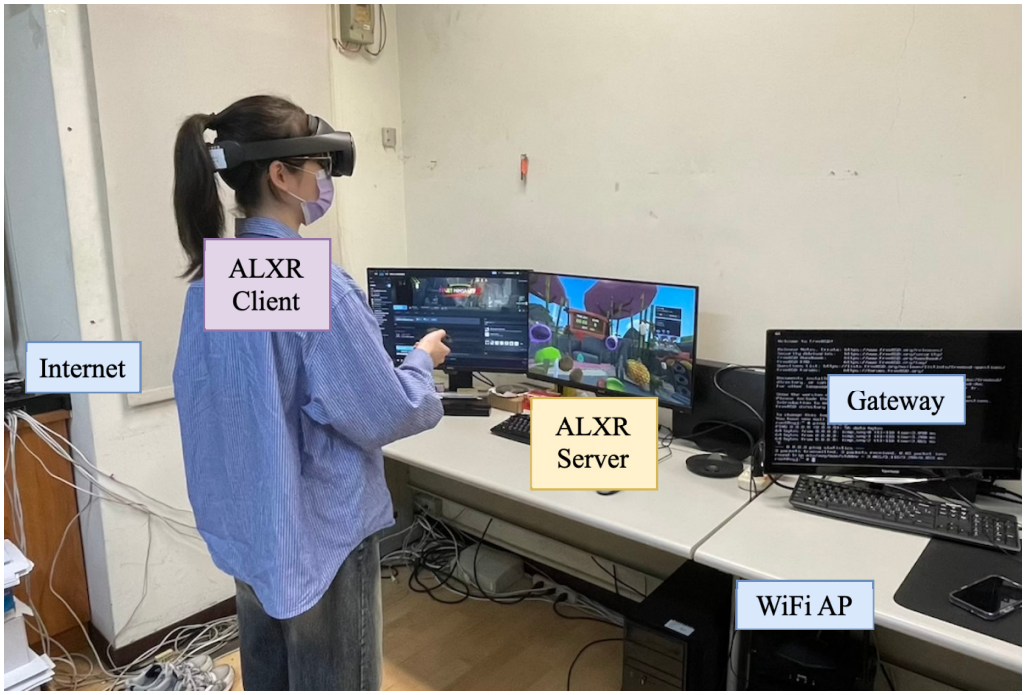


Figure 5.3: Our developed cloud VR gaming testbed photo.

at the top of the HMD viewport. When doing so, HMD gamers may spot artifacts caused by dynamic foveation. We emphasize that our research methodology can be generalized to other, less challenging cloud VR games, which is among our future work.

We ask each subject to play Fruit Ninja in the Arcade mode multiple times in *scenarios* with diverse foveation parameters. When designing scenarios, we opt for 1:1 aspect ratio for the foveal region size ($\Pi_X = \Pi_Y$) and compression ratios ($R_X = R_Y$) to shorten the duration of user study for each subject to avoid fatigue. Some pilot tests revealed that the gaming QoE remains rather similar unless the parameters are significantly varied. Hence, we divide the foveal region size and compression ratios into three intervals and choose the medium parameters in each of them. More specifically, for dynamic foveation, we consider 9 scenarios denoted as (D, Π, R), where D stands for dynamic foveation, $\Pi \in \{0.2, 0.5, 0.8\}$, and $R \in \{2, 5, 8\}$. For comparison, we also consider static foveation with 3 scenarios: (i) *aggressive* that is (S, 0.2, 8), (ii) *balanced* that is (S, 0.5, 5), and (iii) *safe* that is (S, 0.8, 2), where S stands for static foveation. Here, the aggressive scenario comes with a small foveal region with a high compression ratio. In contrast, the safe scenario tries to avoid negative impacts due to imperfect foveated warping or eye tracking. Last, we consider a scenario where foveation is disabled, denoted as (N, -, -), where N stands for no foveation. In total, we have 13 scenarios.

5.2.2 Procedure

We recruited 15 (9 male) subjects for our user study, and all of them were graduate students between 22–25 years old. First, we performed the Snellen test to check their vision: all of them passed with 20/20 corrected visual acuity. We also perform interpupillary distance adjustment and eye calibration to adapt the eye trackers to individual subjects. Before the first scenario of each subject, the concept of foveation, the main purpose of the user study, and the test procedure are orally explained. Next, every subject plays the considered game for about 5 minutes to get familiar with our cloud VR gaming platform and the game. That is followed by 13 scenarios in random order to avoid learning effects and biased ratings. Each session lasts for about 5 minutes. After each game, a subject has 2 minutes to provide his/her rating and take a break. It takes each subject 60–90 minutes to complete the whole experiment. We adopt a single-stimulus method: Absolute Category Rating (ACR) from an ITU-U recommendation [27]. The scores are between 1 and 5, representing the worst and best gaming QoE in *visual quality*, respectively.

Table 5.1: MOS with 95% Confidence Intervals in Parentheses from Different Foveation Parameters under No, Static, and Dynamic Foveation

Fove. / **Para.**	**No**	**Static**			**Dynamic**		
R / Π	-	**0.2**	**0.5**	**0.8**	**0.2**	**0.5**	**0.8**
2	3.47 (± 0.41)	-	-	2.87 (± 0.45)	3.60 (± 0.25)	3.80 (± 0.33)	3.00 (± 0.32)
5		-	**4.07** (± 0.34)	-	2.50 (± 0.47)	**4.67** (± 0.24)	3.80 (± 0.33)
8		1.20 (± 0.20)	-	-	1.87 (± 0.31)	3.07 (± 0.22)	3.13 (± 0.45)

5.2.3 Results

MOS under diverse foveation parameters. Table 5.1 reports the MOS of visual quality across all subjects along with 95% confidence intervals from individual scenarios. This table reveals the best foveation parameters for static and dynamic foveation, which are highlighted in bold font. With static foveation, the balanced scenario (S, 0.5, 5) results in the highest MOS of 4.07. In addition, with dynamic foveation, the same foveation parameters also lead to the highest MOS of 4.67. A deeper look into the recorded viewport videos reveals that:

- When the foveal region becomes too small, gamers could notice the sudden quality jump at the boundary of the foveal and peripheral regions.
- When the foveal region becomes too large, the visual quality of the foveal region is too low at the same encoding bitrate.
- When the compression ratios are too small, the foveal region is too blurred.
- When the compression ratios are too high, the peripheral region is too blurred.

In summary, we found a sweet spot with the foveation parameters (F, 0.5, 5), where $F \in \{S, D\}$, for the highest MOS in visual quality. We note that several subjects told us that: (i) they could see a clear boundary between the foveal and peripheral regions, which is annoying; and (ii) as the gaze positions move rapidly, the movements of the boundary are easy to spot, which makes them dizzy. Indeed, moving from (D, 0.5, 5) to (D, 0.2, R), where $R \in \{2, 5, 8\}$, we found that the MOS drops by 1.07–2.80, which is rather significant.

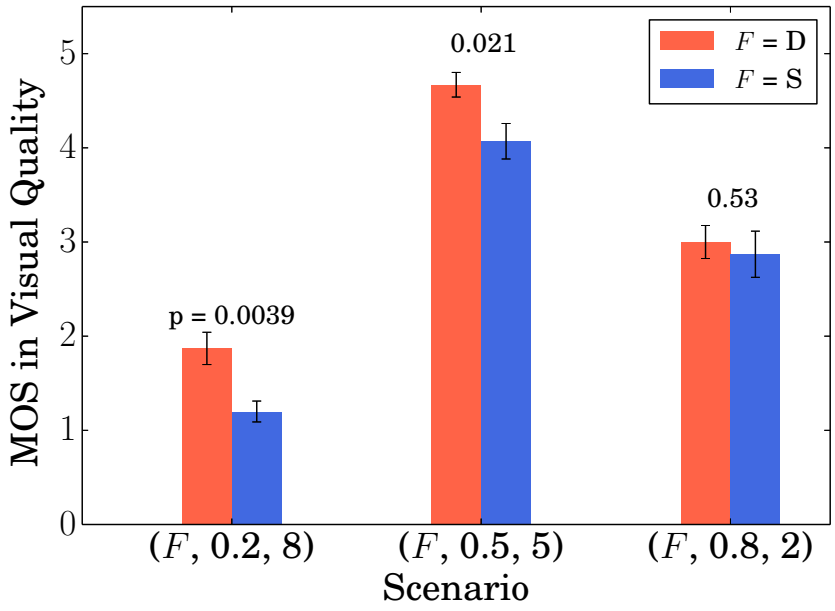


Figure 5.4: Significance test of MOS between dynamic and static foveation.

Next, we check the achieved MOS for statistical significance. Fig. 5.4 compares the MOS between dynamic and static foveation under three scenarios with Wilcoxon signed-rank test [66] results. Fig. 5.4 shows that both (F, 0.2, 8) and (F, 0.5, 5) achieve p values < 0.05, indicating that the MOS between S and D are significantly different. Indeed, with (F, 0.2, 8), where $F \in \{\text{S}, \text{D}\}$, a common feedback from some subjects is that without dynamic foveation, the foveal region prevents them from seeing: (i) fruits at the viewport edge and (ii) remaining time and score at the top of the viewport. This significantly affects the gaming QoE. Such issues are mitigated when dynamic foveation is enabled. For instance, with static foveation (S, 0.5, 5), subjects still report noticing the

foveal region boundary; but they no longer report the same with (D, 0.5, 5). On the other hand, subjects tend to rate (S, 0.8, 2) and (D, 0.8, 2) at the same visual quality, as the implications of larger foveal regions and lower compression ratios are more difficult to perceive. *This demonstrates that dynamic foveation consistently leads to better gaming QoE for most gamers.*

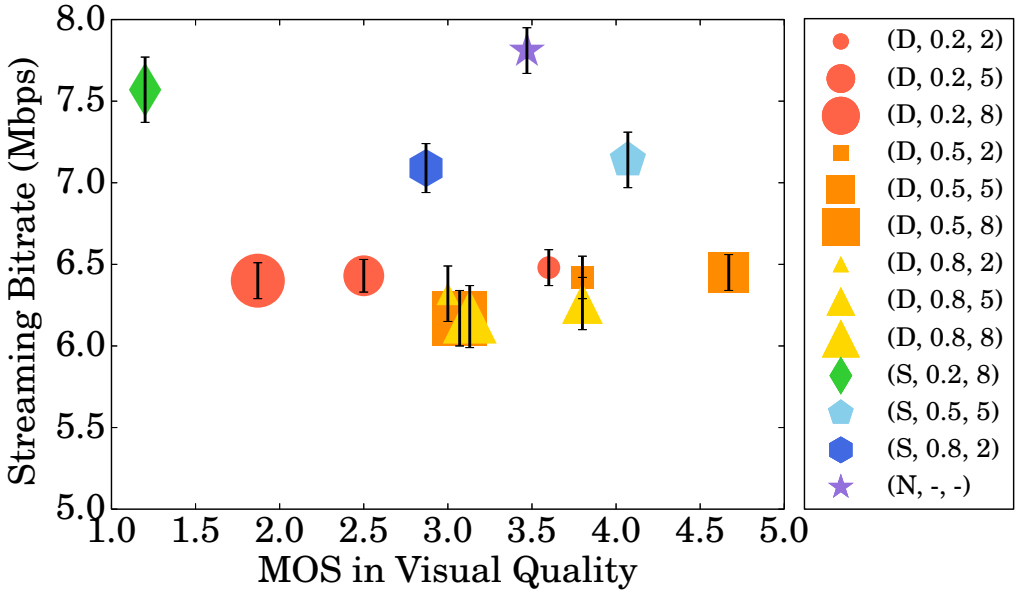


Figure 5.5: MOS versus streaming bitrate with 95% confidence intervals among 13 scenarios. The same marker shape represents the same foveal region size, while the marker size increases along with the compression ratio.

Streaming bitrate under diverse foveation parameters. Next, we check whether (F, 0.5, 5), where $F \in \{S, D\}$, leads to excessively high streaming bitrates, overloading the platform. Fig. 5.5 gives the relation between the MOS and streaming bitrate of all scenarios. Note that the streaming bitrate is measured at the server, accounting for both video packets and control messages. We first observe that the streaming bitrate is slightly higher than the target bitrate of 5 Mbps. This can be attributed to: (i) the bitrate deviation of the rate controller of the video encoder and (ii) the additional traffic due to control messages and Forward Error Correction (FEC). The ALXR project employs FEC to rectify any transmission errors at the client. In this figure, markers closer to the bottom-right corner are better, as they lead to a higher MOS in visual quality at a lower streaming bitrate. Two key findings can be made in this figure:

- *Static foveation leads to higher QoE and lower streaming bitrate compared to no-foveation scenario.* No-foveation scenario consumes the most bitrate but achieves suboptimal MOS in visual quality. Our visual inspection confirms that, in no-foveation scenarios, the viewport frames are blurrier compared to those with static

and dynamic foveation.

- *Dynamic foveation achieves the highest QoE at a low streaming bitrate among all scenarios.* Dynamic foveation with the best parameters leads to an MOS of 4.67 without consuming excessive streaming bitrate.

To summarize, compared to the no-foveation scenario, the optimal parameters for static foveation result in a MOS increase in visual quality of 0.60 and a bitrate reduction of about 8.71%. Moreover, by adopting dynamic foveation, we achieve an additional increase of 0.60 in MOS, while reducing the bitrate by approximately 9.81%, compared to static foveation. *These results demonstrate the values of dynamic foveation in a cloud VR gaming system, answered our RQ1.*

5.3 Limitations of the Enhanced Cloud VR Gaming Platform

The ALXR platform enhanced with dynamic foveation still suffers from the following two limitations.

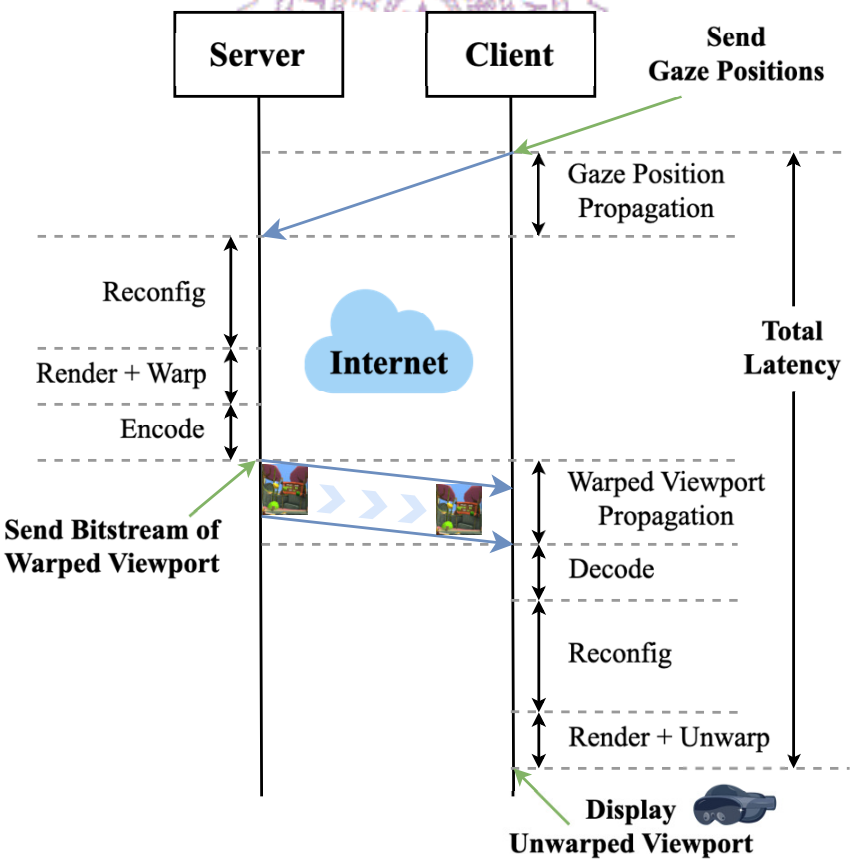


Figure 5.6: Latency components in cloud VR gaming platform.

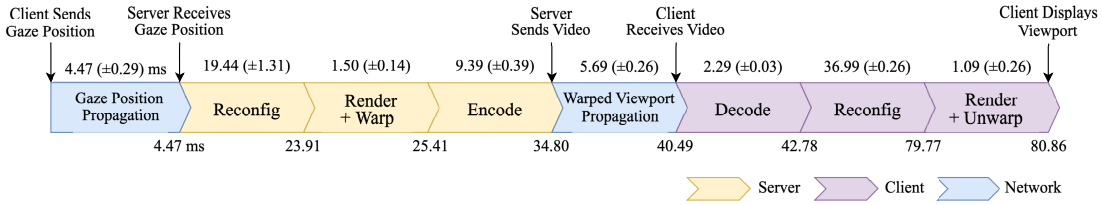
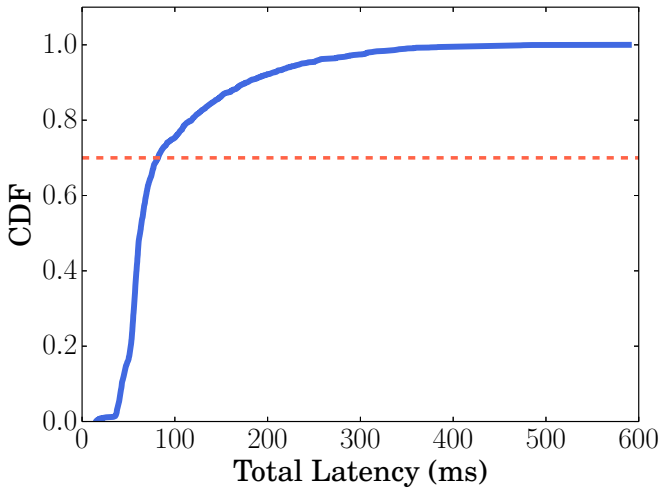


Figure 5.7: Per-component latency between sending gaze position and displaying the resulting viewports.

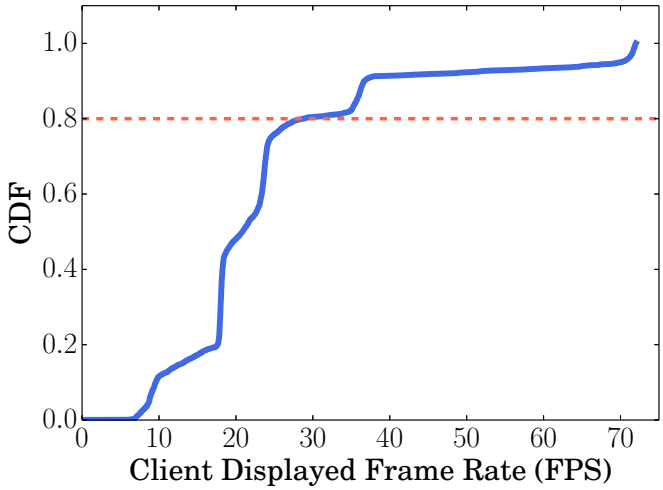
Reconfiguration overhead. Reconfiguring new gaze positions for the foveation modules incurs resource consumption. Therefore, updating the gaze position every single frame may lead to non-trivial high latency and low frame rate. To quantify the overhead, we instrumented our platform to record the per-component latency of individual frames and the client-displayed frame rate. Fig. 5.6 illustrates the timeline of latency components in cloud VR gaming platform. The total latency is divided into eight components: (i) gaze position propagation, (ii) server reconfiguration, (iii) server rendering and warping, (iv) server encoding, (v) warped viewport propagation, (vi) client decoding, (vii) client reconfiguration, and (viii) client rendering and unwarping. We asked an HMD gamer to play Fruit Ninja with medium foveation parameters (i.e., $\Pi_X = \Pi_Y = 0.5$ and $R_X = R_Y = 5$) and a viewport frame resolution of 1184×1056 for 90 seconds. We set the target frame rate at 72 FPS and updated the gaze position for every frame.

In total, we collected 2221 latency and 2787 frame-rate samples. Fig. 5.7 reports the average latency of each component with 95% confidence intervals. This figure reveals that our system achieves a total latency of 80.86 ms on average. Furthermore, Fig. 5.8(a) gives the CDF (Cumulative Distribution Function) of the total latency. This figure shows that about 30% of the total latency go beyond 80 ms, which is not sufficient for cloud VR systems [4]. Cross-referring to Fig. 5.7, we find that the dominating latency components are the server and client reconfigurations, which are 19.44 ($\pm$1.31) and 36.99 ($\pm$0.26) ms. This shows that the key opportunity to reduce the overhead lies in *reconfiguration modules*, which we study in Sec. 6.1. The overhead caused by reconfiguration modules also negatively affects the frame rate. The average frame rate (95% confidence interval) is 24.47 ($\pm$0.23) FPS. Fig. 5.8(b) presents the distribution of the frame rate, depicting that 80% of the samples fall below 30 FPS, which also exposes some room for improvement.

Single foveated warping approach. Our enhanced ALXR platform only supports AADT Warp thus far. It is not clear if AADT Warp is the best foveated warping approach for cloud VR gaming. Our user study indicates that the foveal/peripheral boundary due to AADT Warp is easy to spot. To our best knowledge, diverse foveated warping approaches have never been compared in cloud VR gaming [30]. Hence, we enhance our cloud VR



(a)



(b)

Figure 5.8: The distributions of: (a) latency and (b) frame rate in our experiments.

gaming platform to support *heterogeneous foveated warping approaches* in Sec. 6.3, so that their performance can be quantified and compared.

Chapter 6

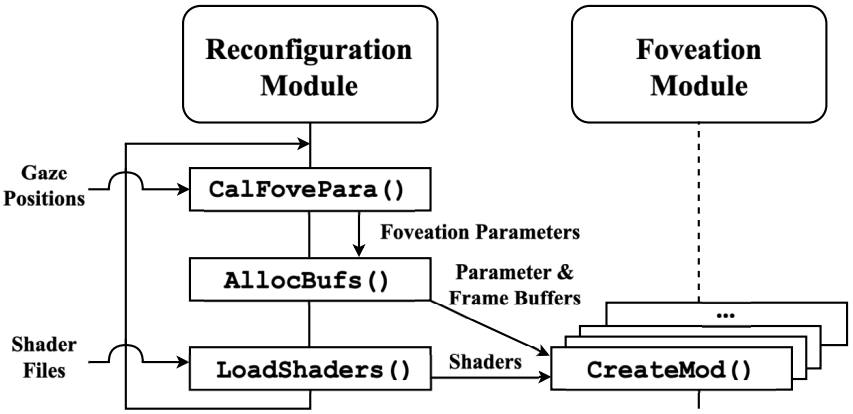
Optimized Dynamic-Foveation-Enabled Cloud VR Gaming Platform

In this section, we first present how we optimize the foveation and reconfiguration modules. This is followed by the details of a foveated warping approach without foveal/peripheral boundary. The optimized system performance is then objectively evaluated.

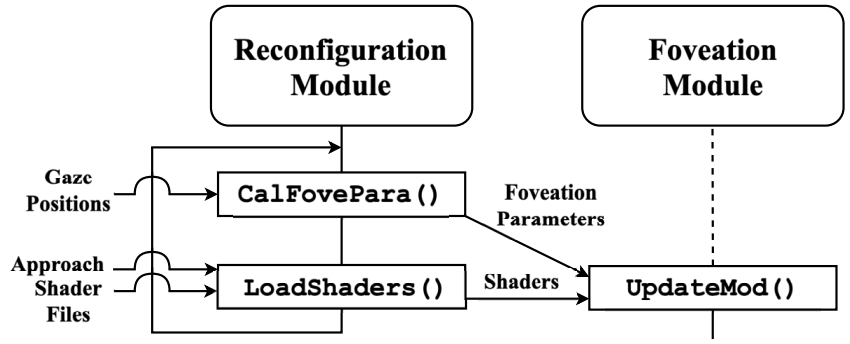
6.1 Foveation Reconfiguration Optimization

In our unoptimized cloud VR gaming platform, the reconfiguration module creates a *new* foveation module whenever the gaze positions are changed. Fig. 6.1(a) outlines the operations at the unoptimized cloud server, while the same operations are done at the HMD client. We note that two shaders are needed to warp the viewport frame, which are the vertex and the pixel shaders. The vertex shader manipulates the geometric properties of 3D vertices, such as their positions, color, and texture coordinates; while the pixel shader controls the color and appearance properties of 2D pixels, such as lighting, shading, and texture mapping. More precisely, the operations done by the reconfiguration module are: (i) `CalFovePara()` that calculates the foveation parameters, such as the resolution of viewport frames, (ii) `AlloBufs()` that allocates buffers for foveation parameters and viewport frames, (iii) `LoadShaders()` that loads the vertex and pixel shaders, and (iv) `CreateMod()` that creates the foveation module using the buffers and shaders. After all steps are completed, a new foveation module is established, replacing the old one.

The foveation module in Fig. 6.1(a) is repeatedly recreated from scratch, because the foveation parameter buffer is read-only. This is the root cause of the inferior performance of dynamic foveation in Sec. 5.3. To address this issue, we modify both the server and client in the following. At the server side, the foveation module is programmed in Direct3D, which is a graphics library for Windows. The class of the foveation parameter



(a)



(b)

Figure 6.1: The sequence diagrams for: (a) (unoptimized) creation and (b) (optimized) update operations of the foveation module. In (b), we skip the creation of the reusable foveation module for brevity.

buffer is `buffer`, which is instantiated with a flag that can be: `Default`, `Immutable`, `Dynamic`, or `Staging`. Among them, `Default` is suitable for more static data with a good balance between performance and access modes, `Immutable` is for strictly static data for optimal performance, `Dynamic` is ideal for frequently updated data for frequent writes on the CPU, and `Staging` is for rapid data transfer between the CPU and GPU. The unoptimized ALXR server employs `Immutable` flag. We first switch it to `Dynamic`. By doing that, we can reuse the buffer and avoid the expensive recreation of the foveation module. At the client side, the foveation module is programmed in Vulkan, which is an open-source multi-OS library for 3D graphics and computing. The class of the foveation parameter buffer is `constants`, which can be instantiated with either `Specialization` or `Push` constants. `Specialization` constants enable developers to specify shaders at the compilation time for better performance. In contrast, `Push` constants allow the CPU to send constants into existing shaders. The unoptimized ALXR client used `Specialization` constants, which needs to be changed into `Push` constants. Fig. 6.1(b) presents the resulting optimized operations, where `UpdateMod()`

only updates the foveation parameter buffer whenever the gaze position is updated.

An unoptimized foveation module can only initialize a pair of vertex and pixel shaders. To enable heterogeneous shaders, we add an argument for the foveated warping approach in the optimized foveation module illustrated in Fig. 6.1(b). This argument can be adjusted through the cloud server GUI, allowing gamers to select the foveated warping approach. The chosen shaders are then loaded/updated in the foveation module. At the HMD client side, the client receives the argument from the server, loading/updating the corresponding shaders accordingly.

6.2 An Alternative Foveated Warping Approach

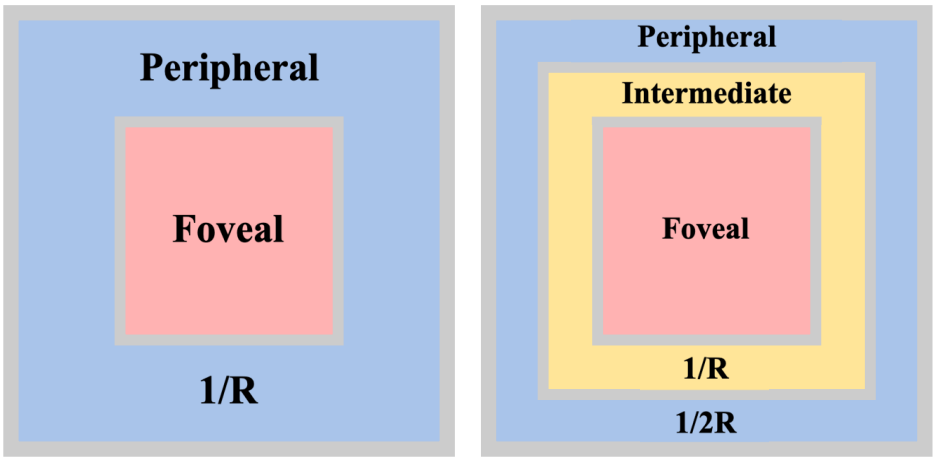
Table 6.1: Different Foveated Warping Approaches

	1 Region	**2 Regions**	**3 Regions**
Non-uniform	Foveated Radial Warp	AADT Warp	Same as AADT Warp
Uniform	No foveation	AADT2 Warp	AADT3 Warp

Table 6.1 presents various foveated warping approaches characterized by different regions and sampling settings referring to the previous work [30]. We define three regions in a viewport frame: (i) foveal region, (ii) intermediate region, and (iii) peripheral region, as illustrated in Fig. 2.2. When the viewport frame has only one region, *No foveation* implies uniformly sampling the entire frame, while *Foveated Radial Warp* applies a warping function to gradually distort the frame from the gaze position. In cases where the viewport frame is divided into two regions, the peripheral region can be sampled in two ways: one with uniform sampling, denoted as *AADT2 Warp*, and another with non-uniform sampling, similar to the AADT Warp used in the previous user study. Similarly, with three regions, two sampling methods are employed. The uniform-sampling approach is denoted as *AADT3 Warp*. However, if the peripheral region is non-uniformly sampled, it is merged with the intermediate region, resulting in the same outcome as in the two-region cases, equivalent to AADT Warp. Detailed implementations of these approaches are provided below.

6.2.1 AADT2 & AADT3

Fig. 6.2 demonstrates the examples of AADT2 Warp and AADT3 Warp. In AADT2 Warp, the viewport frame is divided into two regions, where the foveal region is sampled at the original full rate, while the peripheral region is uniformly sampled at a rate of $\frac{1}{R}$.



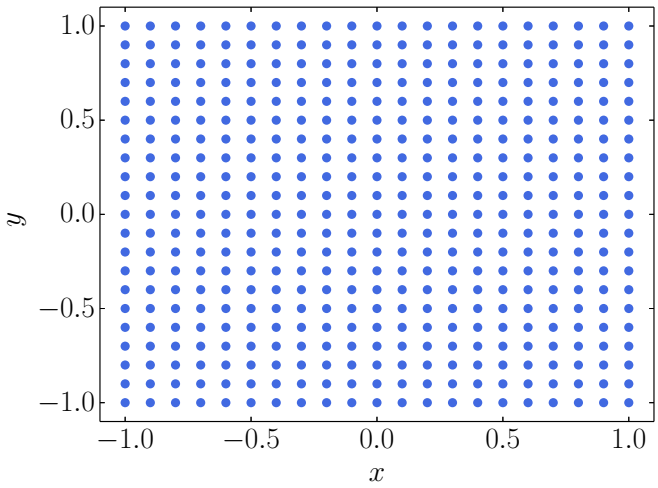
(a) (b)

Figure 6.2: Examples of: (a) AADT2 and (b) AADT3.

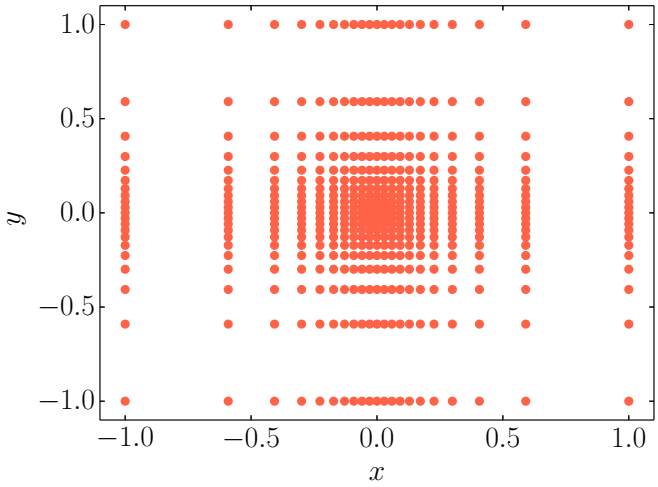
In other words, the resolution of the peripheral region is reduced by R. In AADT3 Warp, the viewport frame is divided into three regions, with an additional intermediate region in the middle. The foveal region is sampled once per pixel, consistent with AADT2 Warp. While the intermediate region is uniformly sampled at a rate of $\frac{1}{R}$, the peripheral region is sampled at $\frac{1}{2R}$. Note that the size of the foveal region is followed by Π, while the intermediate and peripheral region are calculated by the sampling rate R and $2R$, respectively.

6.2.2 Foveated Radial Warp

We next introduce *Foveated Radial Warp* [30] that features a single region described by a *warping function* for determining the sampling coordinates. Its warping function mimicks the HVS: the acuity of human eyes decreases when moving away from the gaze position. Fig. 6.3 gives sampling coordinates in a viewport frame, showing the density variation in a warped viewport frame. We let $p = (p_x, p_y)$ where $p_x, p_y \in [-1, 1]$ are the x- and y-axis coordinates normalized to the resolution of unwarped viewport frames. Similarly, we let $q = (q_x, q_y)$, where $q_x, q_y \in [-1, 1]$ are in warped viewport frames. We next develop the warping function for Foveated Radial Warp [30], denoted as $q = F(p)$ for any p in an unwarped viewport frame. $F(p)$ was inspired by the *Equiangular Cubemap (EAC)* projection, denoted as $E(q)$, proposed by Google [15]. EAC is a well-known 360° video projection [34] that maps spherical images into rectangular ones that can be efficiently compressed by existing 2D video codecs. When projecting 360° video for VR representation, it can be simply categorized into three types: (i) *equirectangular projection*, where the poles receive a significant number of pixels, while the equator gets relatively



(a)



(b)

Figure 6.3: Sampling coordinates in (a) unwarped and (b) warped viewport frames. Foveated Radial Warped with $M = 4.7$ is employed.

fewer pixels. This poses a challenge because important contents in 360° videos are often distributed around the equatorial regions; (ii) *traditional cubemap*, which involves embedding the sphere in a cube and projecting the image on the sphere outward onto the surface of the cube. While it offers an improvement over equirectangular projections, it still results in substantial variation in pixel density; (iii) EAC, the one used by Google on Youtube [15], which addresses the variation in pixel density by altering the locations where the video's pixel samples are taken. Unlike traditional cubemap, where samples have varying lengths depending on their location on the cube face, EAC is specifically designed to maintain equal lengths, ensuring uniformly allocated pixels. This correction contributes to a more consistent distribution of pixels in the representation of 360° video

for VR.

Notice that the 360° video projection works in the inverse direction of the foveated warping function. In particular, a generalized unwarping function of EAC, which maps a point on a cube face to a video pixel sample, can be written as:

$$p = E(q) = (\frac{2}{\pi})\tan^{-1}(4q), \tag{6.1}$$

where $\frac{2}{\pi}$ is a normalizing term mapping the $\tan^{-1}$ value from $[-\frac{\pi}{2}, \frac{\pi}{2}]$ to $[-1, 1]$, while $4q$ inside the $\tan^{-1}$ function gives the slope of the function. According to Eq. (6.1), we refer to the term "4" before q as the magnitude, which increases the rate of approaching the asymptotes. We then define the magnitude of downsampling the pixels approaching the side of the viewport frames as $M \in \mathbb{Q}^+$, which replaces the term 4 in Eq. (6.1) as:

$$p = E(q) = (\frac{2}{\pi})\tan^{-1}(Mq), \tag{6.2}$$

However, since the output is no longer between -1 and 1 with M, we need to find another normalizing term other than $\frac{2}{\pi}$. Let A be the new normalizing term. To map the results from $q \in [-1, 1]$ to $p \in [-1, 1]$, by solving Eq. (6.3) and (6.4), we get $A = \frac{1}{\tan^{-1}M}$.

$$E(-1) = A\tan^{-1}(M(-1)) = -1, \tag{6.3}$$

$$E(1) = A\tan^{-1}(M(1)) = 1, \tag{6.4}$$

$$A = \frac{1}{\tan^{-1}M}, \tag{6.5}$$

Thus, we have the unwarping function:

$$p = E(q) = \frac{1}{\tan^{-1}M}\tan^{-1}(Mq), \tag{6.6}$$

We note that the term $\frac{1}{\tan^{-1}M}$ normalizes the coordinates to be in $[-1, 1]$. In addition, when $M = 1$, Foveated Radial Warp degrades to no warping, i.e., uniform sampling across viewport frames. Last, by inversing Eq. (6.6), we write the warping function as:

$$q = F(p) = \frac{\tan(\tan^{-1}(M)p)}{M}, \tag{6.7}$$

which defines the Foveated Radial Warp approach.

6.3 Objective Evaluations

We conduct objective evaluations to measure the optimized total latency and client frame rate.

6.3.1 Setup

We use the testbed presented in Sec. 5 for experiments. We asked 15 (9 male) subjects aged between 22–26 years old to play Fruit Ninja on the unoptimized and optimized platforms, each for 90 seconds. The bitrate is set to 5 Mbps using the Nvidia H.264 encoder with a CBR rate controller, and the unwarped viewport resolution is set to 1184×1056. We set the frame rate at the server as 72 FPS. The unoptimized platform adopted AADT Warp with optimal $\Pi = 0.5$ and $R = 5$, where the gaze positions were updated at 10 Hz. The optimized platform adopted Foveated Radial Warp with $M = 4.7$, as recommended in Kämäräinen et al. [30].

Table 6.2: System Measurements

Platform	Latency (ms)	Frame Rate (FPS)
Unoptimized	69.15 (± 0.73)	25.64 (± 0.15)
Optimized	13.36 (± 0.52)	68.78 (± 0.05)
Improvement	5.18X	2.68X

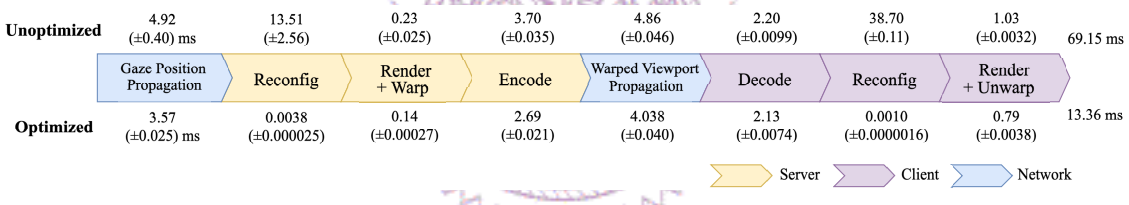


Figure 6.4: Per-component latency measured on the unoptimized and optimized platforms.

6.3.2 Results

Table 6.2 reports the achieved performance in average latency and FPS with 95% confidence intervals across the subjects. This table shows that the unoptimized platform suffers from 69.15 ms total latency on average, while the optimized platform reduces that by 5.18 times. Fig. 6.4 gives the per-component latencies, which reveals that the optimized platform significantly reduces the server reconfiguration latency from 13.51 to 0.0038 ms, and the client reconfiguration latency from 38.70 to 0.0010 ms. This demonstrates that our optimizations on reconfiguration operations in Fig 6.1(b) efficiently reduce the total latency. Table 6.2 also depicts that the unoptimized platform suffers from a low average frame rate of 25.64 FPS, while the optimized platform improves it by 2.68 times, reaching 68.78 FPS. *To sum up, our optimization efforts in the foveation module have resulted in a significant improvement in latency and frame rate, answered our RQ2.*

Chapter 7

Subjective Evaluations

In this chapter, we carry out a user study to subjectively evaluate the optimized cloud VR gaming platform by quantifying the gaming QoE.

7.1 Setup

Table 7.1: Testing Scenarios

Scenario	Platform	Foveated Warping Approach	Π	R	M
Unopt	Unoptimized	AADT Warp	0.5	5	-
Opt$_{4.7}$	Optimized	Foveated Radial Warp	-	-	4.7
Opt$_{6.3}$	Optimized	Foveated Radial Warp	-	-	6.3
Opt$_{7.9}$	Optimized	Foveated Radial Warp	-	-	7.9

The experimental testbed and settings are aligned with those in Sec. 6.3 if not otherwise specified. We asked 15 subjects to play Fruit Ninja under four testing scenarios summarized in Table 7.1. *Unopt* scenario represents the unoptimized cloud VR gaming platform, serving as the baseline. Following the optimal parameters in Sec. 5.2, we let $\Pi = 0.5$ and $R = 5$ with gaze positions updated at 10 Hz. For the optimized cloud VR gaming platform, we only consider three M values to avoid fatigue. In particular, we consider $M \in \{4.7, 6.3, 7.9\}$, where $M = 4.7$ and 7.9 were reported to perform well in objective metrics [30]. In addition, we consider their middle point of $M = 6.3$. We denoted them with Opt_M.

We follow the same preparatory procedure in Sec. 5.2. Particularly, we perform the Snellen test, interpupillary distance adjustment, and eye calibration on each subject. At the beginning, we orally explain the goal of the user study and share the QoE questionnaires. Each subject then undergoes a testing game to get familiar with the system and

Table 7.2: QoE Questionnaire

Question	Description	Score
Overall Quality	How would you rate the overall quality of this gaming scenario?	1 (Bad) – 5 (Excellent)
Visual Quality	How would you rate the visual quality of this gaming session?	1 (Bad) – 5 (Excellent)
Immersive Level	How is your assessment about the sense of immersion during this gaming scenario?	1 (Low) – 5 (High)
Interaction Quality	How responsive was the environment to actions that you performed?	1 (Not responsive) - 5 (Completely responsive)
Cybersickness	Are you feeling any sickness or discomfort now?	1 (Unbearable) – 5 (No problem)

game, followed by four testing scenarios in a random order. After each game, a subject has two minutes to rate that testing scenario, followed by a break. For the QoE questionnaires, we ask five questions [45, 58, 60] given in Table 7.2: *Overall Quality*, *Visual Quality*, *Immersive Level*, *Interaction Quality*, and *Cybersickness*. Except for *Cybersickness*, these scores follow ACR [27] on a 1-5 scale. We adopt the Vertigo scale [45] for subjects to report their discomfort level, which is also on a 1-5 scale. To avoid confusion, higher scores represent better experience on all considered QoE questions, including cybersickness.

7.2 Results

Table 7.3: MOS of Different QoE Questionnaires

Scenario	Overall	Visual	Immersive	Interaction	Cybersickness
Unopt	2.60 ($\pm$ 0.08)	3.53 ($\pm$ 0.09)	2.47 ($\pm$ 0.12)	2.47 ($\pm$ 0.09)	3.60 ($\pm$ 0.12)
Opt$_{4.7}$	**4.40** ($\pm$ 0.06)	**4.27** ($\pm$ 0.07)	**4.58** ($\pm$ 0.58)	**4.67** ($\pm$ 0.06)	**4.67** ($\pm$ 0.08)
Opt$_{6.3}$	4.00 ($\pm$ 0.08)	3.80 ($\pm$ 0.07)	4.00 ($\pm$ 0.10)	4.47 ($\pm$ 0.08)	**4.67** ($\pm$ 0.08)
Opt$_{7.9}$	3.53 ($\pm$ 0.58)	3.00 ($\pm$ 0.11)	3.60 ($\pm$ 0.11)	4.20 ($\pm$ 0.10)	4.33 ($\pm$ 0.10)

Fig. 7.1 and Table 7.3 gives the average QoE scores with 95% confidence intervals under different testing scenarios. We observe that the optimized platform generally achieves higher QoE scores than the unoptimized one in all aspects. We also perform Friedman test [52] to assess whether a significant difference exists between the unoptimized and optimized platforms. As shown in Fig. 7.1, all QoE questions have p values < 0.05, indicating a significant difference in their MOS. *This confirms a preference among the subjects for the optimized platform over the unoptimized one.*

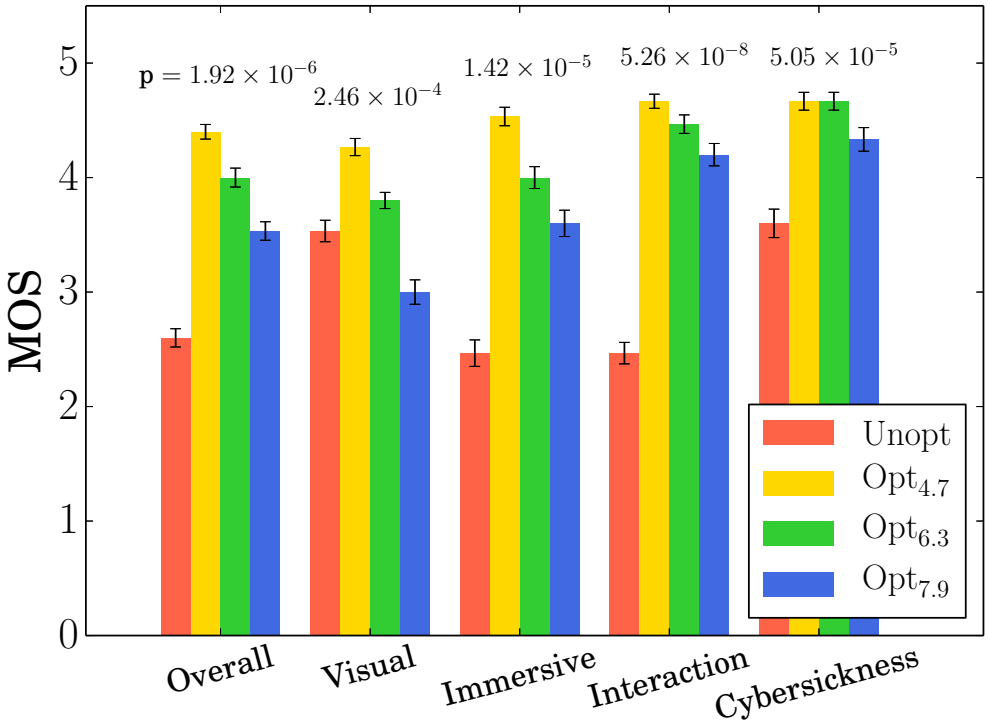


Figure 7.1: MOS of different QoE questionnaires.

To understand why the subjects rated the optimized platform with higher QoE scores, we next analyze the open-ended feedback from the subjects and surmise the possible reasons behind the QoE scores in visual quality, immersive level, interaction quality, and cybersickness. Concerning visual quality, although the unoptimized platform may achieve relatively good quality with $\Pi = 5$ and $R = 0.5$ in the previous user study, the system's additional latency due to reconfiguration overhead hampers the smoothness of the gaming experience. Some subjects mentioned that the visual quality in Unopt is not too bad, but the inferior smoothness negatively affects their visual quality scores. In terms of immersive level and interaction quality, the scores exhibit a similar trend, where the MOS of Unopt both drop to 2.47. Subjects reported that higher latency could heighten the sense of unrealism in the VR world and adversely affect interaction quality due to the Motion-to-Photon (MTP) delay. About cybersickness, subjects noted that the game itself does not impose excessive uncomfortable feelings. However, if the platform incurs additional latency, such as that perceived in the unoptimized platform, they feel a sense of nausea when moving. This effect is also reflected in their scores, where Unopt suffered from the lowest MOS of 3.60 in cybersickness.

Fig. 7.2 shows the MOS of visual quality versus streaming bitrate with 95% confidence intervals among different scenarios. Regardless of extra bitrate exceeding the target bitrate that may be caused by the rate controller of the video encoder and FEC

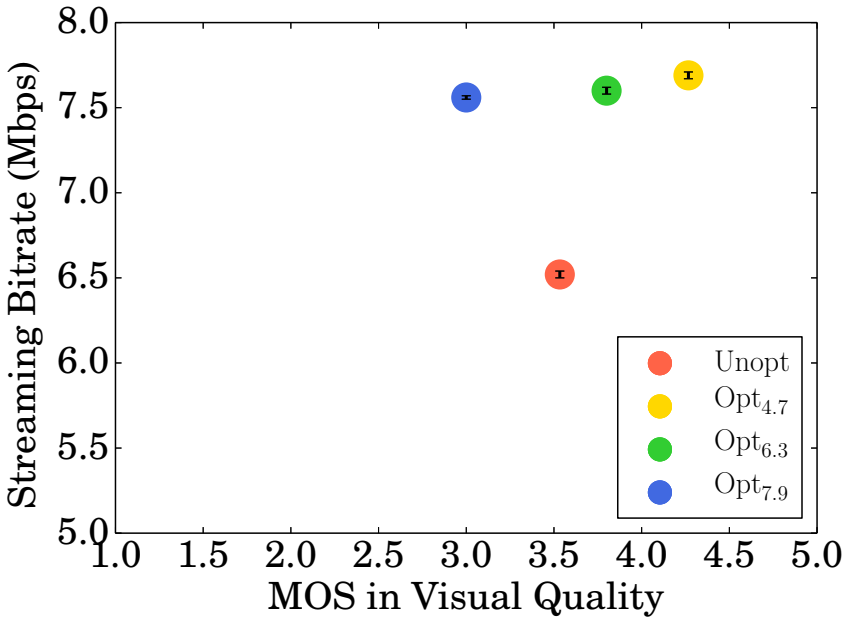


Figure 7.2: MOS of visual quality versus streaming bitrate with 95% confidence intervals among four scenarios.

mechanism, Foveated Radial Warp causes a slightly higher bitrate than AADT Warp in the unoptimized platform, which are 6.52 ($\pm$ 0.02) Mbps and around 7.50 Mbps respectively. This indicates that Foveated Radial Warp may demand more bitrate during frame encoding.

Next, we discuss the implications of different M values in the Foveated Radial Warp approach. According to Fig. 7.1, we observe that $\text{Opt}_{4.7}$ achieves the highest scores across all QoE aspects, while $\text{Opt}_{6.3}$ and $\text{Opt}_{7.9}$ are the second and last place, respectively. In $\text{Opt}_{4.7}$, subjects reported that the artifacts in the peripheral region were hardly noticeable unless they moved their eyes rapidly forward to the viewport boundary. In $\text{Opt}_{6.3}$, some subjects were unable to discern a noticeable difference from $\text{Opt}_{4.7}$, while others could still see slight blurriness. As the visual quality affects other QoE aspects, $\text{Opt}_{6.3}$ leads to slightly lower QoE scores than $\text{Opt}_{4.7}$. In $\text{Opt}_{7.9}$, as shown in Fig. 7.3, subjects reported that the blurring effect induced by the warping function becomes pronounced. This results in the lowest scores in visual quality, as well as discomfort leading to reductions in immersive level, interaction quality, and cybersickness scores, compared to $\text{Opt}_{4.7}$ and $\text{Opt}_{6.3}$. In fact, $\text{Opt}_{7.9}$ suffers from a lower MOS in visual quality than Unopt. Hence, it is crucial to carefully choose the M value for the best gaming experience. Considering the streaming bitrate among the scenarios in Table 6.2, different M values in Foveated Radial Warp do not result in notable differences in bitrate values.

In summary, according to the subjective evaluation results, compared to Unopt, $\text{Opt}_{4.7}$ results in a MOS increase of 1.80 in overall quality, 0.74 in visual quality, 2.11 in im-

(a) (b)

Figure 7.3: The screenshot of warped viewport frames from Foveated Radial Warp with: (a) $M = 4.7$ and (b) 7.9.

mersive level, 2.2 in interaction quality, and 1.07 in cybersickness. *This suggests that we achieve a significant improvement in gaming QoE, answered our RQ3.*

Chapter 8

Conclusion

In this chapter, we begin by summarizing the contributions and key take-away messages of this thesis with pertinent experimental results. This is followed by a discussion of future work and potential research directions that can be explored in subsequent studies.

8.1 Key Take-away Messages

In this thesis, we first enhanced an open-source remote VR gaming system, called ALXR [2], to support real-time cloud VR gaming with dynamic foveation [3]. We set out to answer three RQs by first conducting a user study with an action game called Fruit Ninja to empirically study the potential of dynamic foveation and identify the best foveation parameters for improving the gaming QoE. Our user study revealed that transitioning from static into dynamic foveation yielded an additional 0.60 increase on MOS (on a scale of 1–5) while saving 9.81% bitrate, which addressed our RQ1. Second, to ensure optimal system performance and gaming QoE, we refine the foveation module of our cloud VR gaming platform to reduce the overhead due to the dynamic foveation parameters and incorporate an alternative foveated warping approach. Objective measurements on the optimized system demonstrated a significant reduction of total system latency by 55.79 ms (80.68%), resulting in a more seamless gaming experience at 68.78 FPS (95.53% of the configured frame rate), which addressed our RQ2. Last, we conducted comprehensive subjective evaluations comparing the optimized and unoptimized platforms. The user study demonstrated that our optimized platform could boost the MOS in overall quality by 1.80 while effectively cutting the MOS in cybersickness by 1.07, which addressed our RQ3. Several novel insights were also observed in the two user studies, which are useful on their own sights to future developers of cloud VR gaming:

- Gamers are intolerant to the sudden quality jumps between the foveal and peripheral regions, which are more noticeable when the foveal regions are smaller.

- Gamers are sensitive to latency and frame rate, which could seriously affect the gaming QoE in various aspects, including visual quality, immersive level, interaction quality, and cybersickness.

8.2 Future Work

Like other QoE-related studies, our user study is limited by the number of test scenarios that could lead to fatigued subjects. This may not be a serious issue, as we release the source code of our cloud VR gaming platform [3], which enables the research community to extend our work in multiple directions, including but not limited to the following.

8.2.1 Study the Implications of Different Network Conditions, Foveation Approaches, and Game Genres in Gaming QoE

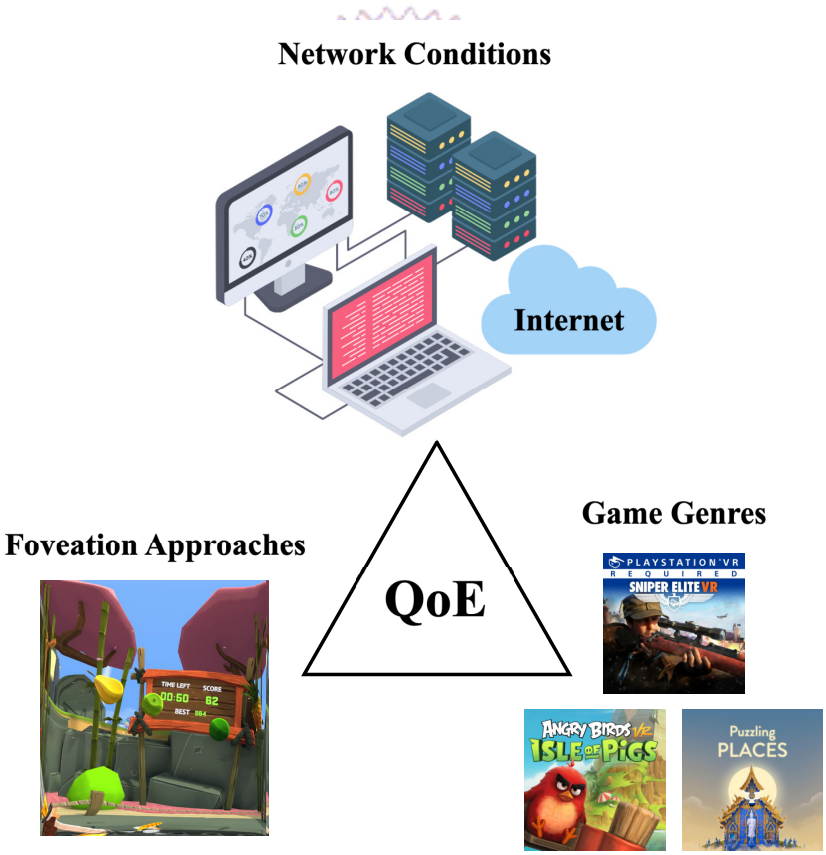


Figure 8.1: Factors affecting QoE in cloud VR gaming.

Fig. 8.1 shows the factors that may affect gaming QoE in cloud VR gaming, which are discussed below. Human perception is rather challenging to predict, especially under diverse and dynamic network conditions. For example, as previously mentioned, gamers are more sensitive to latency. High system latency could lead gamers to abandon playing the

games. Additionally, different bitrate settings or unstable bandwidth during playtime may also result in noticeable quality fluctuations, leading to an unsatisfactory gaming experience. Therefore, it's worthwhile to assess gaming QoE in the optimized cloud VR gaming platform under these network conditions. In this thesis, we compare the performance of AADT Warp and Foveated Radial Warp on the unoptimized and optimized platforms, respectively. However, it might be rather unfair to AADT Warp, as its quality could be influenced by the suboptimal system performance of the unoptimized platform. Thus, in the future, we propose not only to reevaluate these two approaches on the optimized platform, but also to explore and assess more foveated warping approaches from the literature, aiming to achieve optimal visual quality in cloud VR gaming. As the optimized system is able to incorporate various foveation approaches [25, 30], future developers can evaluate their approaches on our optimized platform. Moreover, gamers may demand different aspects of gaming experience, e.g., first-person-shooter gamers may require short latency, puzzle gamers may prefer high visual quality, and role-playing gamers may ask for a comfortable HMD experience. *These additional user studies could help game developers and cloud VR gaming service providers to better design their game and cloud VR gaming platforms for higher gaming QoE.*

8.2.2 Develop Methods to Adapt the System Parameters

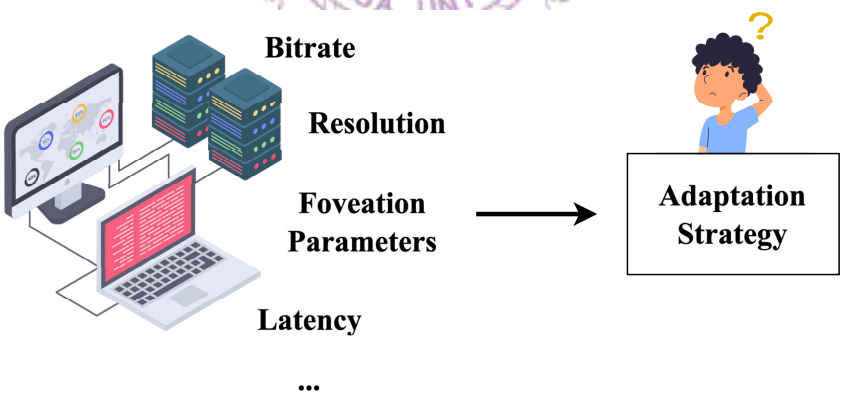


Figure 8.2: Develop adaptation strategies to adjust system parameters dynamically.

As illustrated in Fig. 8.2, cloud VR gaming platforms are complex and inherently have quite a few system parameters, such as bitrate, resolution, foveation parameters, and foveated warping approaches. These parameters are crucial for developing adaptation strategies to achieve better performance. However, selecting a one-size-fits-all set of parameters is infeasible, e.g., under limited network bandwidth, the cloud VR gaming platform may opt for reducing the resolution of the rendered viewport frames to avoid sluggish responses. In contrast, under excessive network latency between the cloud server

and HMD client, a service provider may choose to migrate the cloud server to a closer data center. In terms of the foveation parameters, the optimal foveal region size $\Pi = 0.5$ and $R = 5$ might differ when evaluated on the optimized platform. For instance, if the total latency is significantly reduced, gamers may find smaller foveal region sizes acceptable. This aspect is worth exploring, and the adaptability of foveal region sizes in response to current network conditions is also worthy of investigation. In addition, the ongoing research involves combining QoE modeling techniques with adaptation algorithms to dynamically adjust system parameters based on the measured QoS in real-time [33]. *These methods are crucial for dynamically adapting various system parameters of the cloud VR gaming platform to retain HMD gamers.*

8.2.3 Apply Our Developed Techniques to a Wider Range of Applications

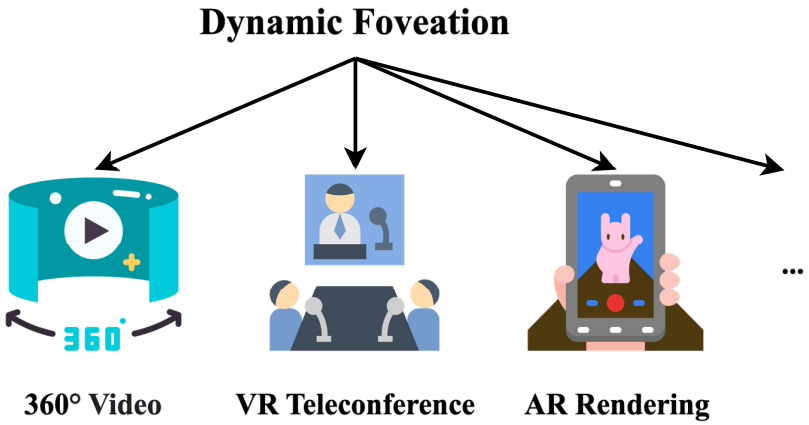


Figure 8.3: Various kinds of applications that can apply dynamic foveation for better performance.

Dynamic foveation may be beneficial for various applications extending beyond cloud VR gaming, as illustrated in Fig. 8.3. This is because cloud VR gaming imposes the highest sensitivity in quality, latency, and bandwidth, compared to other applications listed in Table 4.1 , i.e., 360° video streaming, VR teleconferencing, AR rendering, and cloud gaming, which possess distinct multimedia properties respectively. For example, when viewers watch 360° videos, their attention may be focused on specific areas. By applying dynamic foveation to these areas, the viewing experience can be enhanced, thereby conserving bandwidth usage. Similarly, during VR remote meetings, participants may concentrate on specific areas of the conference table or interactions with others. Through the use of dynamic foveation, the visual details of these areas can be enhanced, while increasing participant engagement and immersion. In cloud gaming, gamers may focus

their attention on specific characters or parts of the game environment. Utilizing dynamic foveation in these focal points can optimize visual quality, heightening the immersion and realism of the gaming experience. Moreover, with the increasing popularity of AR applications, offloading AR applications to the cloud [57] along with dynamic foveation may lead to a more immersive AR experience, which was not possible without the techniques developed in this thesis. *The same can be done to other distributed XR applications along the reality-virtuality continuum, advancing this emerging field.*

Bibliography

[1] The official explanation of aadt, 2019. https://reurl.cc/eX59yL.

[2] The github of alxr, 2021. https://github.com/korejan/ALVR/releases.

[3] Cloud vr gaming platform supporting dynamic foveation, 2024. https://github.com/Jia-WeiFang/ALXR_foveation.git.

[4] R. Albert, A. Patney, D. Luebke, and J. Kim. Latency requirements for foveated rendering in virtual reality. *ACM Transactions on Applied Perception*, 14(4):1–13, 2017.

[5] C. Anthes, R. García-Hernández, M. Wiedemann, and D. Kranzlmüller. State of the art of virtual reality technology. In *Proc. of IEEE Aerospace Conference (AeroConf'16)*, pages 1–19, Big Sky, MT, 2016.

[6] W. Cai, M. Chen, and V. Leung. Toward gaming as a service. *IEEE Internet Computing*, 18(3):12–18, 2014.

[7] W. Cai, R. Shea, C.-Y. Huang, K.-T. Chen, J. Liu, V. Leung, and C.-H. Hsu. A survey on cloud gaming: Future of computer games. *IEEE Access*, 4:7605–7620, 2016.

[8] W. Cai, C. Zhou, V. Leung, and M. Chen. A cognitive platform for mobile cloud gaming. In *Proc. of IEEE International Conference on Cloud Computing Technology and Science (CloudCom'13)*, pages 72–79, Bristol, UK, 2013.

[9] K.-T. Chen, C.-Y. Huang, and C.-H. Hsu. Cloud gaming onward: Research opportunities and outlook. In *Proc. of IEEE International Conference on Multimedia & Expo Workshops (ICMEW'14)*, pages 1–4, Chengdu, China, 2014.

[10] S. Chen, B. Duinkharjav, X. Sun, L.-Y. Wei, S. Petrangeli, J. Echevarria, C. Silva, and Q. Sun. Instant reality: Gaze-contingent perceptual optimization for 3d virtual reality streaming. *IEEE Transactions on Visualization and Computer Graphics*, 28(5):2157–2167, 2022.

[11] J.-W. Fang, K.-Y. Lee, T. Kämäräinen, M. Siekkinen, and C.-H. Hsu. Will dynamic foveation boost cloud vr gaming experience? In *Proc. of Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'23)*, pages 29–35, Vancouver, Canada, 2023.

[12] S. Firdose, P. Lunqaro, and K. Tollmar. Demonstration of gaze-aware video streaming solutions for mobile vr. In *Proc. of IEEE Conference on Virtual Reality and 3D User Interfaces (VRW'18)*, pages 749–750, Tuebingen/Reutlingen, Germany, 2018.

[13] F. Frieß, M. Braun, V. Bruder, S. Frey, G. Reina, and T. Ertl. Foveated encoding for large high-resolution displays. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1850–1859, 2020.

[14] R. Goebel, L. Muckli, and D.-S. Kim. Visual system. *The Human Nervous System Elsevier*, pages 1280–1305, 2004.

[15] Google. Eac used by google in youtube, 2017. https://reurl.cc/qrG6Wq.

[16] B. Guenter, M. Finch, S. Drucker, D. Tan, and J. Snyder. Foveated 3d graphics. *ACM Transactions on Graphics*, 31(6):1–10, 2012.

[17] M. Hegazy, K. Diab, M. Saeedi, B. Ivanovic, I. Amer, Y. Liu, G. Sines, and M. Hefeeda. Content-aware video encoding for cloud gaming. In *Proc. of ACM Multimedia Systems Conference (MMSys'19)*, pages 60–73, New York, NY, 2019.

[18] L. Hsiao, B. Krajancich, P. Levis, G. Wetzstein, and K. Winstein. Towards retina-quality vr video streaming: 15ms could save you 80% of your bandwidth. *ACM SIGCOMM Computer Communication Review*, 52(1):10–19, 2022.

[19] C.-F. Hsu, A. Chen, C.-H. Hsu, C.-Y. Huang, C.-L. Lei, and K.-T. Chen. Is foveated rendering perceivable in virtual reality? exploring the efficiency and consistency of quality assessment methods. In *Proc. of ACM International Conference on Multimedia (MM'17)*, pages 55–63, Mountain View, CA, 2017.

[20] C.-Y. Huang, C.-H. Hsu, Y.-C. Chang, and K.-T. Chen. Gaminganywhere: An open cloud gaming system. In *Proc. of ACM Multimedia Systems Conference (MMSys'13)*, pages 36–47, Oslo Norway, 2013.

[21] Huawei. Cloud vr network solution white paper, 2018. https://reurl.cc/ml3OqV.

[22] W. Ijsselsteijn, Y. de Kort, K. Poels, A. Jurgelionis, and F. Bellotti. Characterising and measuring user experiences in digital games. In *Proc. of International Con-*

ference on Advances in Computer Entertainment Technology (ACE'07), pages 1–4, Salzburg, Austria, 2007.

[23] G. Illahi, T. Gemert, M. Siekkinen, E. Masala, A. Oulasvirta, and A. Ylä-Jääski. Foveated video streaming for cloud gaming. In *Proc. of IEEE International Workshop on Multimedia Signal Processing (MMSP'17)*, pages 1–6, Luton, UK, 2017.

[24] G. Illahi, T. Gemert, M. Siekkinen, E. Masala, A. Oulasvirta, and A. Ylä-Jääski. Cloud gaming with foveated video encoding. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 16(1):1–24, 2020.

[25] G. Illahi, M. Siekkinen, T. Kämäräinen, and A. Ylä-Jääski. Foveated streaming of real-time graphics. In *Proc. of ACM Multimedia Systems Conference (MMSys'21)*, pages 214–226, Istanbul, Turkey, 2021.

[26] N. Inc. Vmaf - video multi-method assessment fusion, 2019. https://github.com/Netflix/vmaf.

[27] T. Installations and L. Line. Subjective video quality assessment methods for multimedia applications. *Networks*, 910(37):5, 1999.

[28] L. Itti. Automatic foveation for video compression using a neurobiological model of visual attention. *IEEE Transactions on Image Processing*, 13(10):1304–1318, 2004.

[29] Y. Jin, M. Chen, T. Goodall, A. Patney, and A. Bovik. Subjective and objective quality assessment of 2d and 3d foveated video compression in virtual reality. *IEEE Transactions on Image Processing*, 30:5905–5919, 2021.

[30] T. Kämäräinen and M. Siekkinen. Foveated spatial compression for remote rendered virtual reality. In *Proc. of Workshop on Metaverse Systems and Applications (MetaSys'23)*, pages 7–13, New York, NY, 2023.

[31] T. Kämäräinen, M. Siekkinen, A. Ylä-Jääski, W. Zhang, and P. Hui. A measurement study on achieving imperceptible latency in mobile cloud gaming. In *Proc. of ACM Multimedia Systems Conference (MMSys'17)*, pages 88–99, Taipei, Taiwan, 2017.

[32] P. Kortum and W. Geisler. Implementation of a foveated image coding system for image bandwidth reduction. In *Proc. of Human Vision and Electronic Imaging (HEVI'96)*, pages 350–360, San Jose, CA, 1996.

[33] K.-Y. Lee, J.-W. Fang, Y.-C. Sun, and C.-H. Hsu. Modeling gamer quality-of-experience using a real cloud vr gaming testbed. In *Proc. of International Workshop*

on Immersive Mixed and Virtual Environment Systems (MMVE'23), pages 12–17, Vancouver, Canada, 2023.

[34] J.-L. Lin, Y.-H. Lee, C.-H. Shih, S.-Y. Lin, H.-C. Lin, S.-K. Chang, P. Wang, L. Liu, and C.-C. Ju. Efficient projection and coding tools for 360° video. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(1):84–97, 2019.

[35] P. Lungaro, R. Sjöberg, A. Valero, A. Mittal, and K. Tollmar. Gaze-aware streaming solutions for the next generation of mobile vr experiences. *IEEE Transactions on Visualization and Computer Graphics*, 24(4):1535–1544, 2018.

[36] P. Lungaro and K. Tollmar. Qoe design tradeoffs for foveated content provision. In *Proc. of International Conference on Quality of Multimedia Experience (QoMEX'17)*, pages 1–3, Erfurt, Germany, 2017.

[37] Meta. Oculus link, 2019. https://reurl.cc/YVnknO.

[38] Meta. Openxr sdk document, 2023. https://reurl.cc/qkk393.

[39] D. Mishra, M. Zarki, A. Erbad, C.-H. Hsu, and N. Venkatasubramanian. Clouds+ games: A multifaceted approach. *IEEE Internet Computing*, 18(3):20–27, 2014.

[40] I. Mohammadi, M.-R. Hashemi, and M. Ghanbari. An object-based framework for cloud gaming using player's visual attention. In *Proc. of IEEE International Conference on Multimedia & Expo Workshops (ICMEW'15)*, pages 1–6, Turin, Italy, 2015.

[41] S. Möller and A. Raake. *Quality of experience: advanced concepts, applications and methods*. 2014.

[42] Nvidia. The official website of nvidia cloudxr, 2020. https://reurl.cc/10djxm.

[43] S. Patil, Y. Chen, and T. Rosing. Gazetube: Gaze-based adaptive video playback for bandwidth and power optimizations. In *Proc. of IEEE Global Communications Conference (GLOBECOM'15)*, pages 1–6, San Diego, CA, 2015.

[44] A. Patney, M. Salvi, J. Kim, A. Kaplanyan, C. Wyman, N. Benty, D. Luebke, and A. Lefohn. Towards foveated rendering for gaze-tracked virtual reality. *ACM Transactions on Graphics*, 35(6):1–12, 2016.

[45] P. Pérez, N. Oyaga, J. Ruiz, and A. Villegas. Towards systematic analysis of cybersickness in high motion omnidirectional video. In *Proc. of International Conference on Quality of Multimedia Experience (QoMEX'18)*, pages 1–3, Cagliari, Italy, 2018.

[46] G. V. Research. Cloud gaming market size, 2020. https://reurl.cc/b7RLRM.

[47] G. V. Research. Virtual reality market size, share & trends analysis report by technology, 2020. https://reurl.cc/MRdadk.

[48] M. Romero-Rondón, L. Sassatelli, F. Precioso, and R. Aparicio-Pardo. Foveated streaming of virtual reality videos. In *Proc. of ACM Multimedia Systems Conference (MMSys'18)*, pages 494–497, Amsterdam, Netherlands, 2018.

[49] J. Ryoo, K. Yun, D. Samaras, S. Das, and G. Zelinsky. Design and evaluation of a foveated video streaming service for commodity client devices. In *Proc. of ACM Multimedia Systems Conference (MMSys'16)*, pages 1–11, Klagenfurt, Austria, 2016.

[50] Sctanf. The linear and non-linear function of foveated warping method in alvr, 2021. https://reurl.cc/3O5N10.

[51] R. Shea, J. Liu, E. Ngai, and Y. Cui. Cloud gaming: architecture and performance. *IEEE network*, 27(4):16–21, 2013.

[52] M. Sheldon, M. Fillyaw, and W. Thompson. The use and interpretation of the friedman test in the analysis of ordinal-scale data in repeated measures designs. *Physiotherapy Research International*, 1(4):221–228, 1996.

[53] J. Shen, L. Yu, L. Li, and H. Li. Foveation-based wireless soft image delivery. *IEEE Transactions on Multimedia*, 20(10):2788–2800, 2018.

[54] S. Shi and C.-H. Hsu. A survey of interactive remote rendering systems. *ACM Computing Surveys*, 47(4):1–29, 2015.

[55] S. Shi, W. Jeon, K. Nahrstedt, and R. Campbell. Real-time remote rendering of 3d video for mobile devices. In *Proc. of ACM International Conference on Multimedia (MM'09)*, pages 391–400, Beijing, China, 2009.

[56] S. Shi, K. Nahrstedt, and R. Campbell. A real-time remote rendering system for interactive mobile graphics. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 8(3s):1–20, 2012.

[57] M. Siekkinen, O. Mertanen, and T. Kämäräinen. Streamed reality, 2023. https://www.streamedreality.com/.

[58] A. Singla. Assessment of visual quality and simulator sickness for omnidirectional videos. *Springer Assessment*, 2024.

[59] R. Solso. *Cognition and the visual arts*, volume 1. 1994.

[60] H. Tran, N. Ngoc, C. Pham, Y. Jung, and T. Thang. A subjective study on qoe of 360 video for vr communication. In *Proc. of IEEE International Workshop on Multimedia Signal Processing (MMSP'17)*, pages 1–6, Luton, UK, 2017.

[61] Valve. Steamvr, 2023. https://www.steamvr.com/en/.

[62] B. Wandell. *Foundations of vision*, volume 8. 1995.

[63] Z. Wang, A. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[64] Z. Wang, L. Lu, and A. Bovik. Foveation scalable video coding with automatic fixation selection. *IEEE Transactions on Image Processing*, 12(2):243–254, 2003.

[65] S. Westen, R. Lagendijk, and J. Biemond. Perceptual image quality based on a multiple channel hvs model. In *Proc. of International Conference on Acoustics, Speech, and Signal Processing (ICASSP'95)*, volume 4, pages 2351–2354, Detroit, MI, 1995. IEEE.

[66] R. Woolson. Wilcoxon signed-rank test. *Wiley Encyclopedia of Clinical Trials*, pages 1–3, 2007.

[67] Z. Xia, Y. Zhang, F. Ma, C. Cheng, and F. Hu. Effect of spatial distortions in head-mounted displays on visually induced motion sickness. *Optics Express*, 31(2):1737–1754, 2023.

[68] A. Zare, A. Aminlou, M. Hannuksela, and M. Gabbouj. Hevc-compliant tile-based streaming of panoramic video for virtual reality applications. In *Proc. of ACM International Conference on Multimedia (MM'16)*, pages 601–605, Amsterdam, Netherlands, 2016.

[69] W. Zou, S. Feng, X. Mao, F. Yang, and Z. Ma. Enhancing quality of experience for cloud virtual reality gaming: an object-aware video encoding. In *Proc. of IEEE International Conference on Multimedia & Expo Workshops (ICMEW'21)*, pages 1–6, Shenzhen, China, 2021.