Low-Latency Streaming for Immersive Remote Interaction with a Spherical Display

Den TABATA*, Taiga KOBORI*, Keisuke SUGIURA†, Yoshiki YAMAGUCHI†‡,
Ryouhei TSUGAMI§, Toshihito FUJIWARA§, Tatsuya FUKUI§, Satoshi NARIKAWA§
* Graduate School of Science and Technology, University of Tsukuba, Tsukuba Ibaraki, 305-8573, Japan
† Institute of Systems and Information Engineering, University of Tsukuba, Tsukuba Ibaraki, 305-8573, Japan
‡ Research and Education Institute for Semiconductor and Informatics, Kumamoto University, Kumamoto, 860-8555, Japan
§ Access Network Service Systems Labs., NTT, Inc., Musashino, Tokyo, 180-8585, Japan

***Abstract*—We present an FPGA-based streaming architecture for immersive remote interaction using a spherical display. The framework integrates real-time distortion correction, perceptually guided compression, and adaptive buffer management to minimize end-to-end latency. Experiments on a Kintex-7 implementation demonstrate a $\sim$40% reduction in on-chip buffer usage and >90% bandwidth savings relative to raw Full-HD streams, achieving 84.8 ms end-to-end latency from gaze acquisition to projection. The results indicate feasibility for optical-network-based teleoperation while identifying HDMI transport as a remaining bottleneck.**

***Index Terms*—Immersive Remote Interaction, Hardware-based System, xR System, Memory Saving, Low Latency, Spherical Display**

I. Introduction

Remote operation technologies are increasingly required in domains such as construction machinery, infrastructure inspection, and autonomous driving, where safety and labor efficiency are of critical importance [1], [2]. By enabling operators to control multiple sites from a distance, these systems have the potential to reduce accidents and mitigate the shortage of skilled personnel. In environments where wireless communication is unreliable due to radio interference, including mountainous areas, underground tunnels, and industrial plants, optical fiber links offer stable and low-latency communication channels that ensure continuous operation [3].

For remote control to be both effective and safe, the communication system must support high-resolution video streaming with minimal latency. At the same time, the visualization platform should deliver immersive feedback that enables operators to perceive spatial relationships with precision [4], [5]. Conventional flat-panel displays are insufficient for this purpose, as they provide only limited peripheral awareness. Spherical displays, in contrast, offer panoramic visual feedback and a stronger sense of immersion, which enhances operator situational awareness and control accuracy. However, spherical displays inherently introduce significant geometric distortions that compromise depth perception and require real-time correction.

In addition to geometric correction, minimizing overall system latency remains a major challenge. Even modest buffering delays can impair control performance, particularly when millisecond-level responsiveness is required. Furthermore, raw high-definition video streams impose substantial bandwidth demands, making efficient compression strategies essential. Human visual system characteristics, such as the concentration of high acuity in the foveal region, provide an opportunity to reduce redundant data without degrading perceived quality [6].

This paper addresses these challenges by presenting a hardware-based architecture for immersive remote interaction using a spherical display. The proposed system integrates three essential elements: real-time distortion correction implemented on FPGA to eliminate external frame buffering delays, perceptually guided compression to reduce bandwidth by exploiting visual acuity distribution, and adaptive buffer management to minimize on-chip memory usage. Together, these techniques achieve both efficiency and immersion within strict latency constraints. The remainder of this paper introduces related work, details the methodology, presents experimental results obtained on a Kintex-7 FPGA platform, and discusses future directions for practical deployment.

II. Related Work

Immersive visualization for remote operation has long been studied in fields ranging from teleoperation to multimedia communication and human–machine interaction. Early research on full-surround and spherical displays highlighted their ability to enhance engagement by providing panoramic visual feedback. However, these systems often suffered from severe geometric distortions that degraded spatial awareness and depth perception [7], [8]. To mitigate these issues, correction algorithms based on interpolation and projection models were introduced, which improved image quality at the expense of computational complexity and increased latency [9]. These limitations motivated the exploration of hardware-accelerated solutions capable of real-time performance.

Parallel to display correction, video compression methods guided by human visual system properties have been proposed to reduce bandwidth requirements. Okada et al. [10] demonstrated a coding scheme that exploits the fact that human vision requires high resolution only in the foveal region, while peripheral vision tolerates lower detail. This approach significantly reduces redundant information and makes high-resolution streaming feasible under limited bandwidth. Ushiwaka et al. [11] further advanced this concept by demon-

strating that perceptual compression can lower data rates by more than 90% without impairing perceived quality, thereby enabling real-time communication over constrained links.

In addition to compression and correction, FPGA-based architectures have been explored for real-time video processing [12]. Dedicated hardware pipelines can drastically reduce processing latency compared with software implementations. In particular, FPGA implementations of coordinate transformation and distortion correction have been shown to support real-time streaming with minimal overhead, making them attractive for immersive systems where responsiveness is critical [13].

Despite these advances, most prior works addressed these challenges in isolation. Existing studies either improved distortion correction [14], introduced perceptual compression [15], or implemented hardware acceleration [16], but few systems integrate all three aspects into a unified architecture [1]. Moreover, the majority of compression approaches have been evaluated in the context of flat displays, leaving their effectiveness for spherical immersive displays underexplored. A comprehensive solution that simultaneously achieves distortion correction, perceptual compression, and low-latency streaming for spherical displays remains an open problem. This study aims to address this gap by proposing and evaluating an FPGA-based system that combines these techniques within a single framework tailored for immersive remote interaction.

III. Methodology

The proposed system aims to provide immersive and low-latency visual feedback suitable for remote interaction through a spherical display. To achieve this, three key components were integrated: FPGA-based distortion correction without external frame buffering, perceptually guided video compression informed by real-time gaze tracking, and adaptive buffer management designed to minimize on-chip memory requirements.

A. System Architecture

Our entire system is summarized in Fig. 3. The architecture consists of two FPGAs connected via an HDMI cable. Full-HD video is first compressed based on gaze information, transmitted through the HDMI link between the devices, distortion-corrected on the FPGA, and streamed out as a projection-ready signal for the spherical display. This combination of real-time correction, perceptual compression, and adaptive buffering enables stable end-to-end streaming under strict latency and memory constraints while keeping communication overhead low.

Conventional distortion correction pipelines store full frames in external DRAM before coordinate transformation, which imposes both memory bandwidth overhead and substantial latency. Our system eliminates this step by implementing address translation directly within the FPGA.

Fig. 1 shows the result of projecting an image onto the spherical display without distortion correction, where the image is noticeably stretched in the vertical direction. We assume that distortion occurs in the vertical upward direction and correct it by applying a coordinate transformation along the vertical axis.

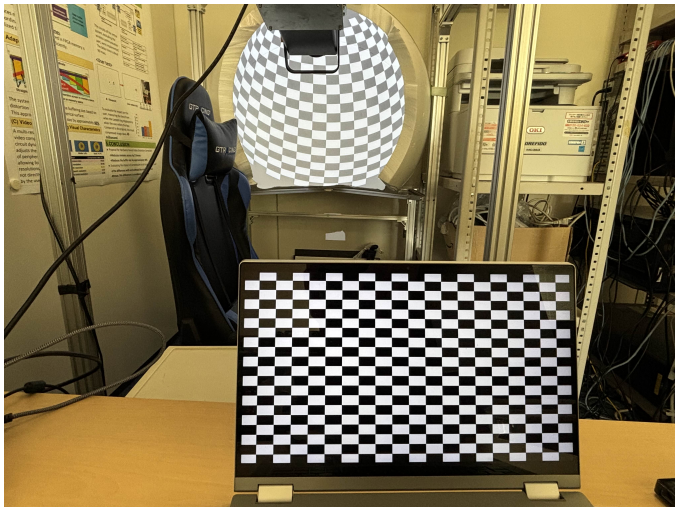

Fig. 1. Uncorrected projection onto the spherical display. The input video is projected without distortion correction, resulting in a visibly stretched image along the vertical direction.

Fig. 2. Projection onto the spherical display after applying the proposed distortion correction. The vertical stretching is successfully eliminated, producing a geometrically faithful image on the spherical surface.

We pre-store in the precomputed correction map the vertical displacement values $\alpha_{x,y}$ corresponding to the input video coordinates (x, y). As illustrated in Fig. 4, each input coordinate (x, y) is transformed to $(x, , y+\alpha_{x,y})$, and the incoming pixels are mapped in real time using this precomputed correction map and written into block RAM (BRAM). Fig. 2 shows the result of projecting the corrected image onto the spherical display, confirming that the vertical distortion is successfully eliminated. Corrected frames are then streamed out in raster order, maintaining output synchronization while avoiding the 16.7 ms delay associated with frame buffering in Full-HD video.

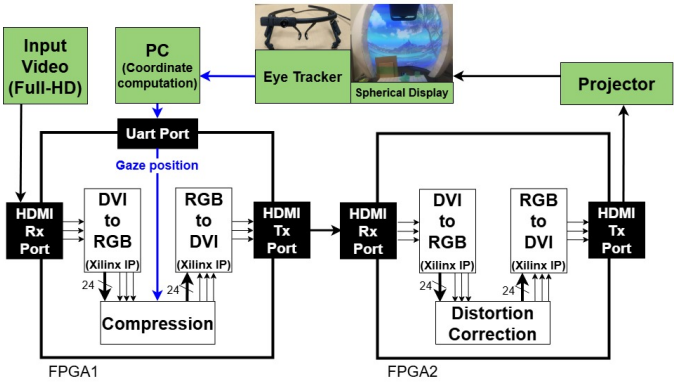

Fig. 3. Overview of the proposed immersive projection system. The pipeline integrates gaze tracking, perceptual compression, FPGA-based correction, and spherical projection for low-latency remote interaction.

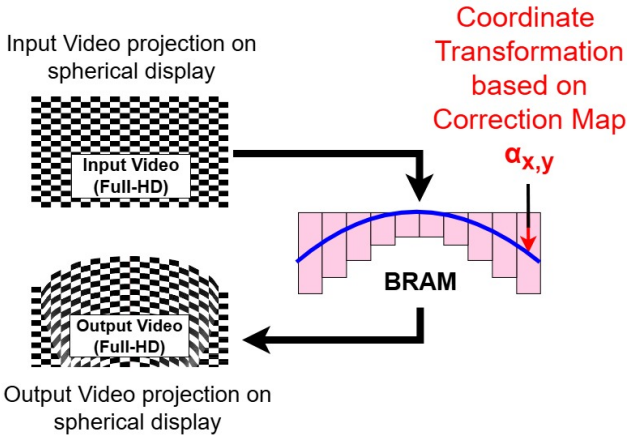

Fig. 4. Proposed distortion correction method implemented on FPGA. By removing external frame buffering, on-the-fly address translation ensures that distortion correction is applied without incurring the typical frame-level delay.

B. Adaptive Buffer Management

The correction process requires buffering to accommodate pixel displacement. A uniform buffer allocation of 300 lines is wasteful since distortion is minor in the center but severe at the edges. As shown in Fig. 5, we divide the frame into ten vertical regions, assigning shallower buffers to the center and deeper buffers to the periphery. This adaptive allocation reduces overall buffer usage by about 40% without degrading correction quality.

C. Precomputed Correction Map

Geometric distortion increases toward the periphery of the spherical display. Direct calculation of coordinate transformations in real time is infeasible for embedded platforms. Instead, we precompute displacement values and store them in a correction map. We displayed a Full-HD image on the spherical display and measured the vertical displacement values $\alpha_{x,y}$ for 18 horizontal lines. For each line (1920 pixels), measurements were taken every 160 pixels, yielding 160 × 18 sampled values of $\alpha_{x,y}$. Exponential regression was applied in the horizontal direction, and linear interpolation was applied in the vertical

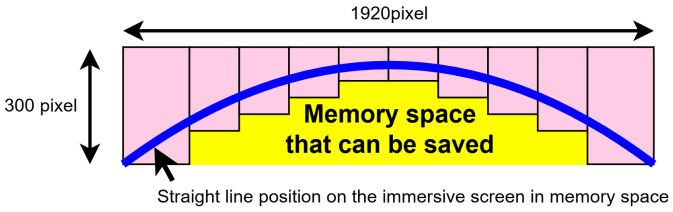

Fig. 5. Adaptive buffer allocation strategy. By tailoring buffer depth to distortion severity, memory usage is reduced by approximately 40% compared to uniform allocation.

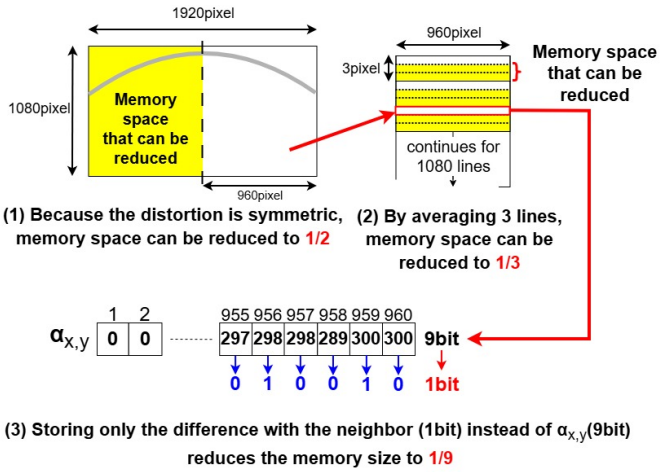

Fig. 6. Correction map reduction strategy. By exploiting distortion characteristics, memory usage is reduced by a factor of 54 compared to the original.

direction to compute $\alpha_{x,y}$ values for the entire 1920×1080 frame. As a result, the maximum vertical displacement was 300.

A naive implementation for $\alpha_{x,y}$ values requires approximately 18.7 Mbit for Full-HD resolution with 9-bit precision. This size is prohibitively large, making it difficult to implement on resource-constrained FPGA. To reduce memory requirements, we exploit the distortion characteristics and drastically minimize memory consumption, as illustrated in Fig. 6. Because the distortion occurs symmetrically, with $\alpha_{x,y} = \alpha_{1919-x,y}$, we exploit this property to reduce the memory size by half (Fig. 6, (1)). In addition, averaging over three horizontal lines reduces it by another factor of three (Fig. 6, (2)). Here, the frame buffer was vertically divided, and measurements confirmed that distortion remains nearly constant within about three lines. Thus, every three lines were grouped and averaged to shrink the parameter table. Furthermore, by storing only the difference from the neighboring coordinate instead of the 9-bit vertical offset, the required bits are reduced to 1/9 (Fig. 6, (3)). As a result, the correction map α, originally a 1920×1080 array with 9-bit entries, is compressed to a 960×360 array with 1-bit entries. This strategy compresses the correction map to 1/54 of its original size (345.6 kbit) while preserving sub-pixel accuracy, which remains imperceptible to users.

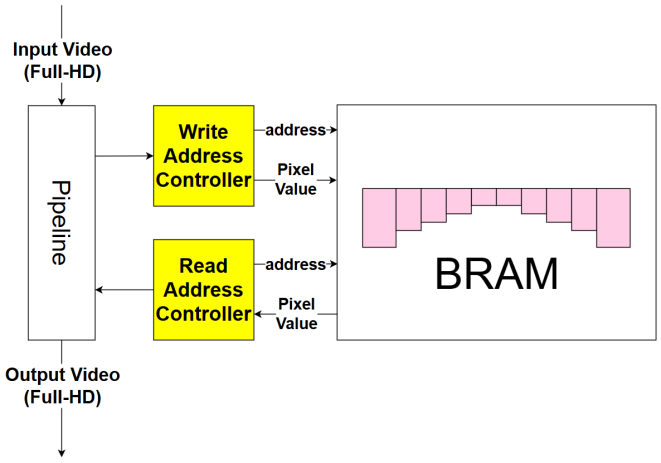

Fig. 7. Overall architecture of the distortion correction circuit, consisting of a Write Address Controller for BRAM input and a Read Address Controller for BRAM output.

D. Distortion Correction Circuit

As shown in Fig. 7, the FPGA implementation of distortion correction is organized into write and read controllers.

The write controller, shown in Fig. 8, stores incoming pixels into BRAM based on the precomputed correction map.

When pixels are written into BRAM by the write controller after coordinate transformation, blank regions may appear between lines. These blanks occur because the displacement difference between successive lines, i.e., $\alpha_{x,y} - \alpha_{x,y-1}$, can be greater than 0. In such cases, the projected coordinates $(y - 1 + \alpha_{x,y-1})$ and $(y + \alpha_{x,y})$ differ by more than 1, leaving pixel positions in between unassigned and resulting in blank regions. The read controller, shown in Fig. 9, retrieves pixel values from BRAM and simultaneously fills these blanks during output. To enable this process, the system uses a blank map that indicates where blank regions occur in the image. The blank map is precomputed in advance and stored in BRAM within the FPGA. Since blanks appear symmetrically, the required blank map size is 1.0 Mbit (1920/2 $\times$ 1080). While reading from BRAM, the controller also checks the blank map to determine whether the fetched pixel is blank. If the pixel is not blank, it is stored in a one-line buffer; otherwise, the corresponding pixel value from the previous line in the buffer is output instead. The one-line buffer requires 46.1 kbit of storage (1920 pixel $\times$ 24 bit). This procedure effectively fills blank regions by propagating the pixel value directly above, extending it downward into the empty locations.

E. Gaze-Guided Perceptual Compression

Streaming raw Full-HD video at 60 fps requires nearly 3 Gbps bandwidth, which exceeds the capacity of many practical links. To address this, we apply compression guided by human visual system properties. An eye tracker identifies the operator's gaze point, and resolution is preserved at the foveal region while progressively reduced toward the periphery. This method reduces the required bandwidth by more than 90%

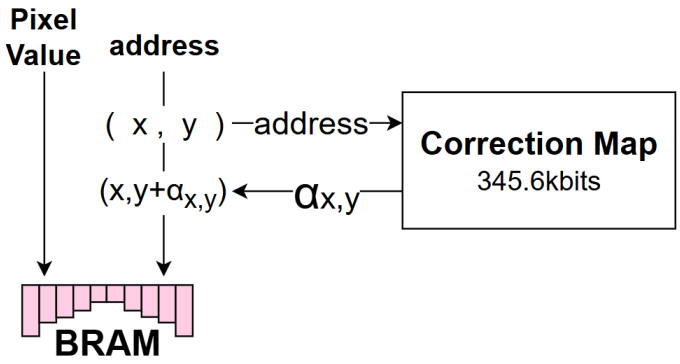

Fig. 8. Write address controller driven by the precomputed correction map.

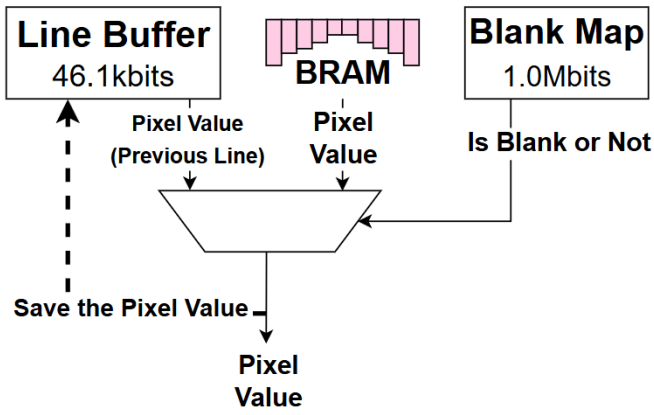

Fig. 9. Read address controller with blank-area compensation. Empty regions caused by coordinate mapping are filled using values from adjacent pixels.

compared to the uncompressed stream, without compromising perceived image quality. Fig. 10 shows the block diagram of the compression circuit. Five levels of compression (1/1, 1/4, 1/9, 1/16, and 1/36) are computed in parallel. For example, in the 1/36 case, a 12×12 pixel block is down-sampled into a 2×2 block. In parallel with this process, the distance between each input pixel (x, y) and the gaze position (g_x, g_y) is calculated. The gaze position is obtained from an eye tracker, and the distance calculation is performed for one frame using the gaze position at the beginning of the frame. The distance for each 12×12 block is evaluated using an initialization and recurrence scheme as follows.

At the beginning of each frame, the difference values are initialized as

$$dx = 0 - g_x, \tag{1}$$

$$dy = 0 - g_y, \tag{2}$$

$$S(x, y) = dx^2 + dy^2. \tag{3}$$

For horizontal updates (pixel-wise in x direction), the recurrence relations are

$$S(x + 1, y) = S(x, y) + 2dx + 1, \tag{4}$$

$$dx \leftarrow dx + 1. \tag{5}$$

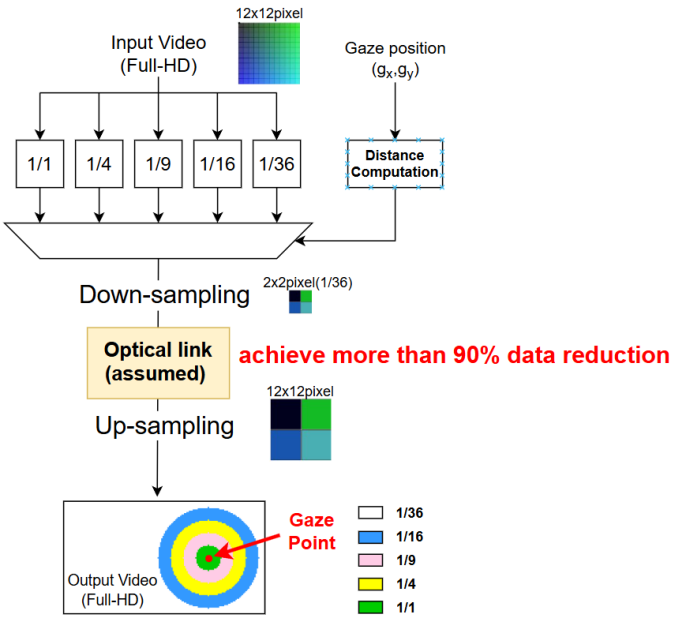

Fig. 10. Compression circuit leveraging gaze information. Resolution is preserved at the foveal region and gradually reduced in the periphery, decreasing bandwidth demand while maintaining visual quality.

For vertical updates (line-wise in y direction), the recurrences are

$$dy^2 \leftarrow dy^2 + 2dy + 1, \tag{6}$$

$$dy \leftarrow dy + 1, \tag{7}$$

$$S(0, y+1) = dx^2 + dy^2 \quad \text{with } dx = 0 - g_x. \tag{8}$$

The calculated distance is then compared with threshold values to determine the compression stage. The selected compressed output is transmitted to the subsequent circuit, where the image is up-sampled back to Full-HD resolution.

IV. Experimental Setup and Results

A. Experimental Platform

The proposed system was implemented in Verilog HDL on a Digilent Genesys 2 board equipped with a Xilinx Kintex-7 XC7K325T FPGA, operating at 148.5 MHz. Logic synthesis and place-and-route were performed using Vivado 2022.2. A commercial projector (LG HU80KS) with a wide-angle conversion lens (Raynox DCR-CF187PRO) was used to project corrected images onto a custom-built spherical screen.

Eye-tracking data were acquired in real time using a Pupil Core device (Pupil Labs). Using Pupil Labs' software Pupil Capture [17], pupil-center positions were obtained and streamed to the PC. On the PC , pupil-center extraction, calibration, and gaze-coordinate computation were performed. A Python 3 script then transmitted the resulting gaze coordinates to the FPGA via UART.

Resource utilization after place-and-route is reported in Table I, with values representing the combined usage across the two FPGAs. BRAM consumption reached 94%, reflecting the memory-intensive nature of buffering and correction maps, while logic and DSP utilization remained modest.

TABLE I
FPGA Resource Utilization

Resource	Used	Available	Util. (%)
LUT	30,802	203,800	15.1
LUTRAM	2,577	64,000	4.0
FF	29,763	407,600	7.3
BRAM	837	890	94.0
DSP	30	840	3.6

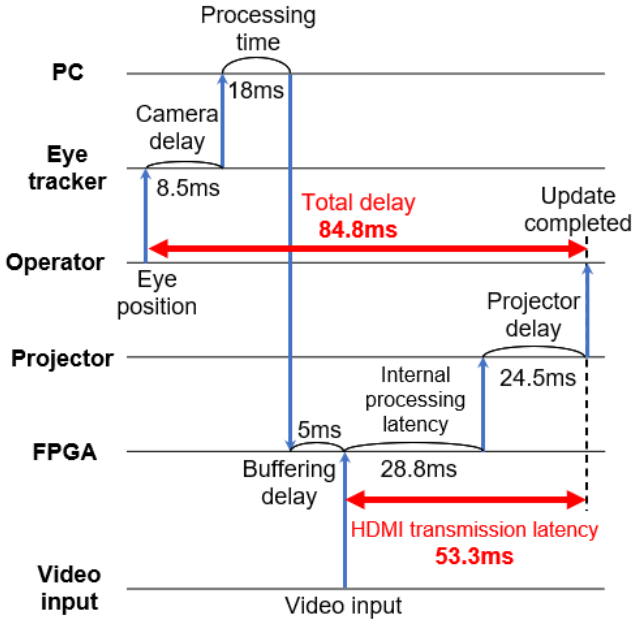

Fig. 11. Measured latency breakdown. The cumulative system latency from gaze capture to display projection is 84.8 ms, of which 53.3 ms is attributed to HDMI transport.

B. Latency Analysis

Display performance is monitored using high-speed cameras that capture reference timestamps on both input and projection sides. Timestamps allow measurement of the actual end-to-end delay. Fig. 11 presents the measured latency contributions of each system component. The eye tracker introduced an average delay of 8.5 ms due to camera capture and preprocessing. The PC-based gaze coordinate transformation added approximately 18 ms. On the FPGA, distortion correction and compression incurred a processing delay of 28.8 ms, together with an additional 5 ms buffering delay. The projector contributed 24.5 ms of internal processing latency. Of the total end-to-end delay of 84.8 ms from gaze acquisition to spherical display projection, 53.3 ms was attributed to HDMI transport, accounting for approximately 63% of the overall latency. These results confirm real-time feasibility while clearly identifying HDMI transfer as the dominant latency component.

C. Bandwidth and Memory Efficiency

The effectiveness of the perceptual compression scheme was confirmed through throughput analysis. By maintaining high resolution only in the foveal region and reducing resolution in peripheral areas, the required bandwidth was reduced to less than 10% of the raw Full-HD stream, while maintaining perceived image quality.

Adaptive buffer management further reduced on-chip memory requirements compared with the uniform allocation of 300-line buffers. On this FPGA, the proposed method lowers BRAM consumption by approximately 40%, demonstrating that the design remains feasible even under strict memory constraints.

D. Discussion of Results

The results confirm that the integration of distortion correction, perceptual compression, and adaptive buffering enables immersive spherical visualization within strict latency constraints. The system achieves near real-time responsiveness and efficient memory usage on a single mid-range FPGA platform. The analysis also reveals the limitations of conventional HDMI interfaces, motivating the adoption of low-latency transport methods such as SDI or dedicated FPGA-to-projector links in future implementations.

V. Conclusions

We introduced a unified FPGA pipeline for immersive spherical visualization in remote interaction. The system achieves 84.8 ms processing-to-display latency, $>$90% bandwidth reduction via gaze-guided compression, and $\sim$40% on-chip memory savings through adaptive buffering and map compaction. Future work will focus on end-to-end transport minimization, UHD scaling, and controlled user studies quantifying the impact on task performance.

Acknowledgements

This work was partially supported by JSPS KAKENHI under Grant Number JP19H00806 and by the AMD University Program. The authors would like to express their sincere gratitude for this support. The study was also conducted with the approval of the Ethics Review Committee of the Faculty of Systems and Information Engineering, University of Tsukuba (Approval Number 2022R713).

References

[1] J. Dóka, B. G. Nagy, D. Jocha *et al.*, "Enablers of low-latency immersive interaction in future remote-rendered mixed reality applications," in *Proceedings of the ACM Multimedia Systems Conference (MMSys '25)*, 2025, pp. 170–180.

[2] J. P. Rolland and H. Fuchs, "Optical versus video see-through head-mounted displays in medical visualization," *Presence*, vol. 9, no. 3, pp. 287–309, 2000.

[3] P. J. Winzer, "Scaling Optical Fiber Networks: Challenges and Solutions," *Optics & Photonics News*, vol. 26, no. 3, pp. 28–35, 2015.

[4] W. Hashimoto, Y. Mizutani, and S. Nishiguchi, "Projection simulator to support design development of spherical immersive display," in *Proceedings of HCI International 2017 – Posters' Extended Abstracts*, C. Stephanidis, Ed., 2017, pp. 17–24.

[5] D. Englmeier, "Spherical Tangible User Interfaces in Mixed Reality," Ph.D. dissertation, Ludwig-Maximilians-Universität München, 2021.

[6] I. W. E. D. Rahmanu and G. Molnár, "Using spherical video-based immersive virtual reality technology to investigate vocabulary mastery among university sophomores in EFL settings," *Cogent Education*, vol. 11, no. 1, p. 2425226, 2024.

[7] H. Iwata, "Full-surround image display technology," *Transactions of the Information Processing Society of Japan Computer Vision and Image Media (CVIM)*, vol. 42, no. SIG13(CVIM3), pp. 41–48, 2001.

[8] W. Hashimoto, T. Mizutani, and T. Nishiguchi, "Projection simulator supporting the design and development of immersive spherical displays," in *Proceedings of the Information Processing Society of Japan Conference*, 2016, pp. 1–4, no. 33C-05.

[9] T. Mori, M. Tonoma, Y. Ohsumi *et al.*, "High quality image correction algorithm with cubic interpolation and its implementation of a dedicated hardware engine for fish-eye lens," *Journal of the Institute of Image Electronics Engineers of Japan*, vol. 36, no. 5, pp. 680–687, 2007.

[10] M. Okada, T. Sato, K. Inada *et al.*, "High efficiency video coding transmission system based on human vision properties," *Transactions of the Information Processing Society of Japan*, vol. 59, no. 7, pp. 1425–1434, 2018.

[11] K. Ushiwaka, Y. Yamaguchi, and H. Yano, "Low-latency stream compression for high-immersive remote control based on human visual acuity distribution," in *Proceedings of the IEICE Technical Report (RECONF)*, vol. 120, no. 36, 2020, pp. 1–6.

[12] P. Greisen, S. Heinzle, M. Gross *et al.*, "An FPGA-based processing pipeline for high-definition stereo video," *EURASIP Journal on Image and Video Processing*, vol. 2011, p. 18, 2011.

[13] H. T. Ngo and K. V. Asari, "A pipelined architecture for real-time correction of barrel distortion in wide-angle camera images," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 3, pp. 436–444, 2005.

[14] H. Blasinski, W. Hai, and F. Lohier, "FPGA architecture for real-time barrel distortion correction of colour images," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, 2011, pp. 1–6.

[15] L. Lin, Y. Wang, J. Liu *et al.*, "SJND: A Spherical Just Noticeable Difference Modelling for 360° Video Coding," *Signal Processing: Image Communication*, vol. 138, p. 117354, 2025.

[16] N. Ujjainkar, E. Shahan, K. Chen *et al.*, "Exploiting Human Color Discrimination for Memory- and Energy-Efficient Image Encoding in Virtual Reality," in *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '24)*, 2024.

[17] Pupil Labs, "Pupil Capture," https://docs.pupil-labs.com/core/software/pupil-capture/, accessed: 2025-10-03.