https://doi.org/10.7840/kics.2025.50.7.1011

Performance Measurement of a Real-Time Optical Camera Communication System on an Edge Server

Tae Hyun Kim◆, Yeong Min Jang°

ABSTRACT

Optical camera communication (OCC), a branch of optical wireless communication, provides rapid, energy-efficient, and secure data transmission. This study introduces a real-time OCC performance measurement platform tailored for resource-limited edge devices. The system uses a two-dimensional on-off keying multiple input multiple output (2D OOK-MIMO) modulation technique to decode data from an 8 × 8 LED grid, processed on an edge server without GPU enhancement. Latency is evaluated using timestamp-based delay analysis to measure system performance. The performance measurement results indicate that the 10 Hz flicker rate yields low latency and bit error rates (BERs), thereby enhancing real-time performance. However, the 16 Hz flicker rate increases latency variability and BERs, reducing the dependability of the results without multi-processing programming. By contrast, multiprocessing, which utilizes the entire processor of the edge server, substantially improves the average latency from 94.1 ms to 30.09 ms. This performance improvement is achieved by parallelizing the frame acquisition, object detection, and data decoding stages, allowing the system to handle incoming frames concurrently. Notwithstanding computing resource limitations, the proposed framework sustains real-time performance, facilitating low-latency OCC deployment.

Key Words : Optical camera communication, Edge computing, Realtime system, Latency

Ⅰ. Introduction

Optical wireless communication, particularly optical camera communication (OCC), has garnered attention owing to advantages such as high-speed transmission, low energy consumption, and high security[1-3]. OCC uses a CMOS camera to receive and decode data transmitted via modulated light, offering natural immunity to electromagnetic interference and improved safety for human eyes[3]. These characteristics render OCC an appealing option for applications that require reliable, secure, and low-latency communication.

In real-time systems such as industrial IoT for manufacturing, delay time measurement is crucial for determining system performance. Delay, defined as the time difference between message transmission and database insertion, directly impacts decision-making processes and operational efficiency[4]. In OCC systems, precisely measuring delay is essential for ensuring low-latency communication and stable performance in edge-based environments.

However, OCC deployment on edge devices presents several challenges. Edge servers typically operate under limited processing power and lack dedicated GPUs, thereby complicating latency optimization. Additionally, OCC systems with high LED refresh rates often experience high latency variability and bit

※ This study was supported by the Korea Research Institute for Defense Technology Planning and Advancement (KRIT) grant funded by the Korean Government (Defense Acquisition Program Administration (DAPA)) (KRIT-CT-23-041, LiDAR/RADAR Supported Edge AI-based Highly Reliable IR/UV FSO/OCC Specialized Research Laboratory, 2024).
◆ First Author : School of Electronic Engineering, Kookmin University, ykj8806@kookmin.ac.kr, 학생회원
° Corresponding Author : School of Electrical Engineering, Kookmin University, yjang@kookmin.ac.kr, 종신회원
논문번호 : 202504-089-B-RU, Received April 15, 2025; Revised May 13, 2025; Accepted May 20, 2025

error rates (BER), further complicating real-time performance.

To address these issues, this study introduces an improved OCC performance measurement platform optimized for edge computing environments. The platform incorporates a two-dimensional on-off keying multiple input multiple output (2D OOK-MIMO) modulation technique for data decoding from an 8 × 8 LED grid and applies parallel programming to maximize the processing potential of the edge server's CPU. Parallel processing significantly reduces the average latency from approximately 94.1 ms to 30.09 ms only. This improvement enables the platform to maintain reliable performance even under resource-limited conditions.

This study additionally investigates the trade-offs between two LED refresh rates, i.e., 10 Hz and 16 Hz, analysing the impact on latency and BER. The obtained results reveal that the systems running at 10 Hz performed consistently better in terms of both latency and BER, whereas the 16 Hz configurations exhibited high latency variability and error rates. Despite these challenges, the enhanced platform demonstrated substantial performance gains, confirming the effectiveness of the proposed parallel programming approach.

The remainder of this paper is organized as follows. Section 2 outlines the proposed method, detailing the OCC platform architecture and parallel programming implementation. Section 3 describes the experimental setup, including the hardware specifications and testing conditions. Section 4 presents the results and discussion in which performance improvements and trade-offs are analysed. Finally, Section 5 concludes the study with the key findings and potential directions for future work.

Ⅱ. Proposed Method

In this section, we present the architecture of the OCC measurement platform and the enhanced methodology that uses parallel programming to improve system performance.

2.1 Optical Camera Communication System

The system employs a 2D MIMO[5] modulation technique to encode data for transmission through the 8 × 8 LED matrix. Each LED within the matrix represents a binary state, i.e., an ON state for binary '1' and an OFF state for binary '0.' By combining these binary states across multiple LEDs, the system simultaneously transmits multiple data bits. This modulation technique is designed to improve data throughput while ensuring that the signal patterns are easily recognizable by the receiving camera.

The camera, operating at 60 frames per second (fps), continuously captures video frames that contain LED modulation patterns. Each frame is analyzed in real time by an edge device to decode the transmitted data. This decoding process involves detecting individual LED states, reconstructing the original binary data stream, and aligning the received data with the expected transmission sequence.

To precisely measure latency, each data packet carries three distinct timestamps that track the movement of the data through the system. The encoder timestamp is generated at the LED matrix during transmission to determine the precise moment the data leaves the transmitter. The decoder timestamp is recorded at the point at which the camera captures and successfully decodes the transmitted data. Finally, the database timestamp is added when the decoded data is written to the database.

2.2 Parallel Programming

The OCC monitoring platform faced significant performance issues during its initial development. The primary bottleneck was the sequential image processing pipeline where frame acquisition, object detection, and data decoding were implemented in series. This serial design generated delays, especially when handling multiple frames per second.

To resolve this issue, parallel programming was introduced to utilize the full capabilities of the edge device's Intel Core i5-8250U CPU, which features four cores and eight threads. The system achieved significant performance gains by distributing the OCC pipeline across independent concurrent threads, simultaneously processing different stages of data handling.

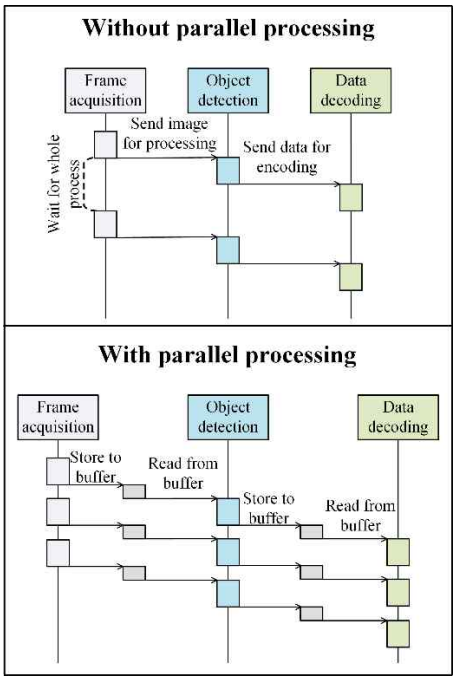

Fig. 1. Sequence diagram of OCC processes.

The improved design separates the data flow in the system into three parallelized tasks. The first task involves frame acquisition, in which a dedicated thread continuously captures frames from the camera. This ensures that the system maintains a consistent flow of incoming data without missing frames. Prioritizing this task proves crucial in minimizing data loss under high-traffic conditions.

The second task is object detection, where a separate thread identifies and localizes the LED source within each captured frame. This detection result is passed efficiently to the next stage without holding up the acquisition process. Parallel execution here ensures that detection keeps pace with incoming frames.

The third task involves data decoding, which interprets the spatial-temporal changes of the detected LED to reconstruct the original binary data. By assigning it to an independent thread, the system avoids decoding bottlenecks and ensures responsive, real-time operation.

Previous studies have explored techniques to improve decoding reliability in OCC systems, such as accumulating pixel rows within the region of interest to enhance signal quality and reduce bit errors[6]. While effective in improving BER, these approaches generally increase per-frame processing complexity and are less focused on system responsiveness. In contrast, the method proposed in this work emphasizes architectural improvements in the receiver by implementing parallel processing frame acquisition, object detection, and data decoding—across independent threads. This design choice in our paper prioritizes real-time performance and lower latency, making it more suitable for deployment on edge devices where responsiveness is critical.

To further enhance efficiency, several key optimizations were introduced in the parallel programming model. Thread prioritization was implemented to ensure that the frame acquisition thread maintained the highest priority, preventing data loss during intense activity. Additionally, batch processing was adopted for the object detection and decoding stages, during which multiple frames were grouped and processed together to reduce overhead. This minimized the delays owing to frequent thread switching. Finally, a shared memory buffer system was employed to facilitate rapid data exchange between threads, reducing the requirement for excessive data copying and improving overall throughput[7].

2.3 Latency Measurement

To evaluate the improved performance of the system, latency was assessed using the embedded timestamps described earlier. The latency metrics were analyzed using the following formulas to measure both responsiveness and stability.

Latency was measured by comparing the timestamp of data transmission at the OCC transmitter with that of data reception at the OCC receiver. In IoT manufacturing systems, the reception timestamp is recorded and embedded in the payload, and the timestamps of reception and database insertion are logged[4].

$$L_k = D_t + D_d \tag{1}$$

$$\overline{L} = \frac{1}{N}\sum_{k=1}^{N} L_k \tag{2}$$

$$\sigma_L = \sqrt{\left(\frac{1}{N-1}\right)\sum_{k=1}^{N}(L_k - \bar{L})^2} \quad (3)$$

$$\bar{J} = \frac{1}{N-1}\sum_{k=1}^{N}|L_{k+1} - L_k| \quad (4)$$

Equation (1) calculates the latency of the system; D_t is the delay from the transmitter to the receiver, i.e., the time difference between the encoder and decoder timestamps, and D_d is the delay from the receiver to the database, i.e., the time difference between the decoder and database timestamps. The total latency for each transmission is denoted as L_k, representing the end-to-end delay observed in the k-th transmission.

Equation (2) is used to measure the average latency $\bar{L}$, which represents the mean value of latency over N transmissions. It is computed as the arithmetic average of the individual latency values L_k, where k denotes the index of each transmission. This average serves as a baseline metric for evaluating the typical delay experienced in the system during normal operation.

Equation (3) uses the standard deviation σ_L of the latency to accurately measure the variability of the latency. It is derived by taking the square root of the variance of the latency values, offering insight into how much the latency deviates from the mean latency $\bar{L}$. A smaller σ_L indicates more consistent system performance, whereas a larger value implies higher fluctuation and less predictability in communication delays.

Equation (4) measures the average jitter $\bar{J}$ to determine the latency variation between consecutive transmissions. It is calculated as the mean of the absolute differences between successive latency values. By quantifying the temporal variation in delay between adjacent data packets, jitter provides a critical metric for assessing real-time system stability, particularly in time-sensitive applications where consistent delivery intervals are essential.

Ⅲ. Experimental Setup

In this section, we present the architecture of the OCC performance measurement platform and the enhanced methodology that uses parallel programming to improve system performance.

3.1 Hardware Specifications

The system was deployed on a resource-constrained edge server to reflect realistic deployment conditions. The edge device employed was a JECS-8250B-i5 mini-PC, equipped with an Intel Core i5-8250U CPU running at 1.6 GHz, featuring four cores and eight threads. The device was configured using 16 GB of DDR4 RAM and operated on Windows 10 (64-bit). The edge device lacked GPU support; therefore, all processing tasks relied entirely on CPU resources, rendering this device an ideal platform for testing the effectiveness of parallel programming in overcoming hardware limitations.

The OCC transmitter employed an 8 × 8 LED matrix, configured to operate at two distinct flicker rates, i.e., 10 Hz (100 ms cycle) and 16 Hz (60 ms cycle). These refresh rates were selected to analyze the impact of flicker speed on latency and BER performance. Each flicker rate test was conducted separately to assess system behaviors across different data transmission speeds. In addition, a 60 Hz flicker rate was introduced to evaluate the system under eye-safe, flicker-free conditions that align with real-world deployment scenarios. All experiments are conducted in a fixed 3 meter distance.

This hardware configuration was selected to mimic

Table 1. System device specifications

Items	Sort
Edge server	JECS-8250B-i5
CPU	Intel Core i5-8250U 1.6 GHz
RAM	16 GB
Operating system	Windows 10
Camera	IDS U3-3040CP
LED matrix	8x8
Modulation	2D OOK-MIMO
Programming language	Python

practical edge computing environments in which hardware resources are limited and performance optimization is crucial.

3.2 Testing Scenarios

A series of controlled performance tests were conducted to evaluate the effectiveness of the enhanced OCC platform and the impact of parallel programming. These tests were designed to measure the latency, stability, and data accuracy of the system across different operating conditions. The hardware used and the setup are illustrated in Figure 2. By conducting repeatable tests under various configurations, the improvements achieved owing to parallel processing were isolated.

The testing procedure incorporated two key LED refresh rates, i.e., 10 Hz (100 ms cycle) and 16 Hz (60 ms cycle). These refresh rates were chosen to reflect typical OCC transmission settings and compare the performances under slow and fast flicker cycles. The tests were conducted twice, i.e., using the original sequential processing model and when implementing the improved parallel processing. This direct comparison clearly identified the influence of parallel programming.

To ensure consistency and reduce variability, each test configuration was implemented across four consecutive runs. Each run lasted for 5 min, yielding thousands of data samples for each test case. This extended test duration was crucial for capturing meaningful performance trends while minimizing the impact of outliers.

In the parallel programming configuration, multiple threads were deployed to simultaneously execute distinct tasks in the OCC processing pipeline. A dedicated thread was responsible for frame acquisition, ensuring continuous and uninterrupted capture of video frames from the camera. Concurrently, a separate thread performed object detection to localize the LED source within each frame as they arrived. In parallel, another decoding thread processed the detected object data to reconstruct the transmitted binary information. This multi-threaded approach allowed each critical task to operate independently, significantly reducing bottlenecks and improving system responsiveness and throughput.

The same hardware and environmental conditions were maintained during the sequential and parallel processing tests to ensure effective comparison. The tests were conducted in a controlled lighting environment to minimize noise and reflections that could affect LED detection accuracy. Additionally, the camera was positioned at a fixed distance from the LED matrix to maintain consistent image clarity and field-of-view throughout the test runs.

By implementing parallel programming in the improved system, the tests demonstrated measurable improvements in latency reduction, stability, and data integrity. The results of these tests provided clear evidence of the effectiveness of parallel programming in addressing the performance limitations encountered in the original sequential design.

Figure 3 displays the flowchart for measuring system performance, with a total of three timestamps included, i.e., the encoder, decoder, and database timestamps. To ensure accurate time representation,

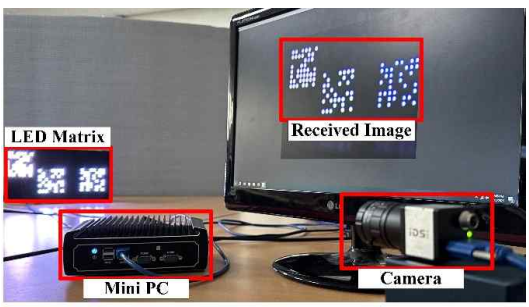

Fig. 2. Experimental setup of the OCC system.

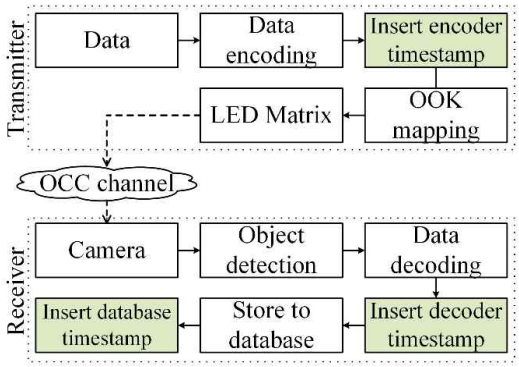

Fig. 3. Flowchart for measuring system performance.

the ISO 8601 standard was used. For example, the date "2024-12-25" specifies the year, month, and day, whereas the time "12:00:00.500" includes hours, minutes, seconds, and milliseconds (500 ms is the fractional part).

Algorithm 1 summarizes the step-by-step processing flow of the proposed multithreaded receiver.

As shown in Figure 3, the system is structured into three parallel threads: frame acquisition, object detection, and data decoding, with data passed through thread-safe queues to ensure synchronization and stability. By separating these tasks into parallel threads and managing them through queue-based communication, the system achieves low latency and stable performance, even on devices with limited computing resources.

Algorithm 1 Parallel OCC Receiver with Frame Acquisition, Detection, and Decoding

```
Input: Live camera frames at 60 fps
Output: Decoded data packets with timestamps
Initialize buffers: Q1 (acquisition to detection), Q2 (detection to decoding)
Launch three parallel threads:
Frame_Acquisition
Object_Detection
Data_Decoding
procedure FRAME_ACQUISITION
    while system is running do
        Capture frame from camera
        Timestamp frame and store in Q1
    end while
end procedure
procedure OBJECT_DETECTION
    while system is running do
        Read frame from Q1
        Detect LED region and extract binary pattern
        Store result in Q2
    end while
end procedure
procedure DATA_DECODING
    while system is running do
        Read binary data from Q2
        Detect preamble and decode payload
        Apply Reed-Solomon and CRC check
        Save decoded result with timestamp
    end while
end procedure
```

3.3 Performance Metrics

To comprehensively evaluate system performance, several key metrics were measured throughout the testing process. Latency measurements were prioritized, with three specific metrics used to assess responsiveness and consistency.

Mean latency represented the average delay experienced across all data packets during each test run. This metric clearly measured the overall system responsiveness.

To assess variability in latency performance, the standard deviation metric was calculated, clarifying the consistency in the system's performance over time. Additionally, jitters were measured to evaluate short-term fluctuations in latency between consecutive data packets.

In addition to latency-related metrics, the BER was analyzed to assess data accuracy. The BER was calculated by comparing the decoded data against the original transmitted data, providing a clear indicator of transmission integrity.

Ⅳ. Experiment and Results

4.1 Latency Analysis

Based on the previous scenario, we collected data. We consecutively ran the data collection scenario 7 times using three refresh rates, i.e., 10 Hz, 16 Hz, and 60 Hz, of the LED matrix. The limited computing power of the edge device limited the image processing capabilities; therefore, we determined whether the reduced refresh rate of the LED matrix affected the latency of the device. Additionally, we obtained results from the device with parallel processing.

The dataset labels in Table 2 refer to different test conditions. 10Hz-1 and 10Hz-2 are two separate runs of the system using a 10 Hz LED refresh rate, while 16Hz-1 and 16Hz-2 were recorded at 16 Hz. The "-1" and "-2" simply indicate repeated trials to check for consistency in performance. A third test was conducted at 60 Hz, reflecting a more typical flicker rate for user-facing applications. The datasets labeled P-10Hz and P-60Hz correspond to test using parallel processing for frame acquisition, object detection, and

Table 2. Measurement results demonstrating system performance

Dataset	Mean [ms]	Standard deviation [ms]	Jitter [ms]	BER
10Hz-1	94.1	45.6	59.3	0.08
10Hz-2	100.1	47	62.2	0.1
16Hz-1	140.4	50.8	68.4	0.23
16Hz-2	148.3	50.9	67.8	0.34
60Hz	161.7	54.2	68.6	0.45
P-10Hz	30.09	11.48	12.67	0.002
P-60Hz	33.2	13.0	14.9	0.004

data decoding. These combinations allow for a clear evaluation of both flicker rate and processing architecture on system performance.

The latency distribution is further illustrated in Figure 4, which depicts the frequency of the observed latency values across all datasets. As shown in the figure, the parallel processing system consistently maintains lower latency values compared to the non-parallelized counterparts. In addition, Figure 5 presents a boxplot comparing latency variability. These data highlight that the 10 Hz datasets demonstrated more stable latency performance than the 16 Hz and 60 Hz counterparts. Parallel processing further improved latency consistency, with the parallelized system exhibiting the narrowest distribution range and lowest median latency.

The 10 Hz systems exhibit a lower BER than the 16 Hz and 60 Hz systems, as shown in Figure 6. A clear correlation existed between increasing latency and worsening error performance, possibly owing to the timing mismatch between the LED and the camera

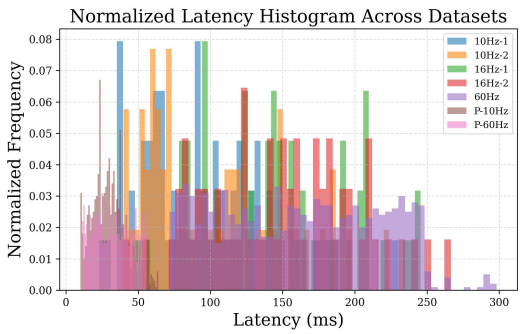

Fig. 4. Histogram of latency across datasets.

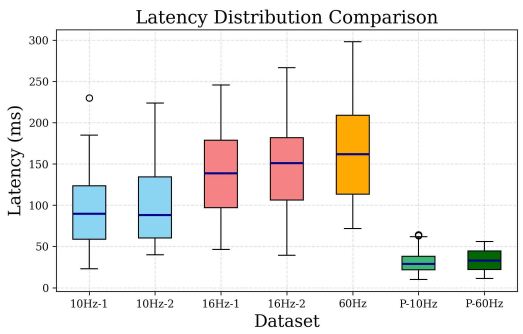

Fig. 5. Boxplot of latency for the entire datasets.

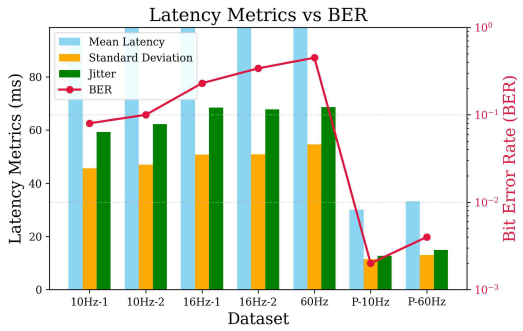

Fig. 6. Comparing the BER and latency metrics.

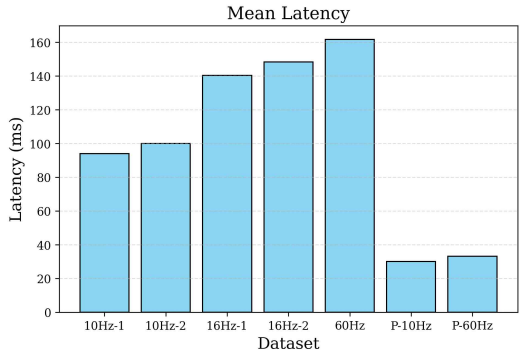

Fig. 7. Comparing mean latency

and the instability of the edge device with limited processing power.

In conclusion, the results received were satisfactory. The best and worst performances corresponded to average latencies of 30.09 ms and 161.7 ms, respectively. Implementing OCC in the edge device is still feasible to obtain real-time system performance.

4.2 Prior OCC Systems Comparison

To highlight the practical advantages of the proposed system, a direct comparison is made with two representative OCC receivers from the literature: Sitanggang et al.[5] and Zhang et al.[6].

Table 3 presents a comparison between the proposed system and two representative OCC receivers from previous studies. While all three systems use 2D OOK modulation, they differ in decoding strategy, hardware requirements, and overall performance. The proposed system delivers a data rate of 11.52 kbps with a low mean latency of 30.09 ms and a BER of

Table 3. Comparison of results with prior OCC systems

Metrics	Proposed system (this work)	Sitanggang et al. [5] (2023)	Zhang et al. [6] (2021)
Modulation scheme	2D OOK-MIMO	2D OOK-MIMO	OOK
Decoding method	Parallel programming pipeline	CNN-based decoding	Sequential sampling
Hardware requirement	Mini pc with CPU only	GPU-equipped pc	Standard desktop pc
Distance	3 m	5 m	1 m
Data rate	11.52 kbps	3.84 kbps	5 kbps
Mean latency	30.09 ms	Not specified	Not specified
BER	0.0020	~0.020	0.0038

0.0020, all achieved using a CPU-only mini PC. In contrast, Sitanggang et al.[5] use a CNN-based decoder that requires GPU acceleration, and Zhang et al.[6] rely on sequential sampling with a standard desktop PC, both resulting in lower throughput or higher system demands. These results demonstrate that the proposed approach is well-suited for real-time OCC applications on resource-constrained edge devices.

4.3 Standard Deviation

Standard deviation analysis was conducted to assess the variability of latency results, thereby revealing system stability. As shown in Figure 8, the 10 Hz datasets exhibit lower standard deviation values than the 16 Hz and 60 Hz datasets. This indicates that lower flicker rates increased the predictability and consistency of latency performance.

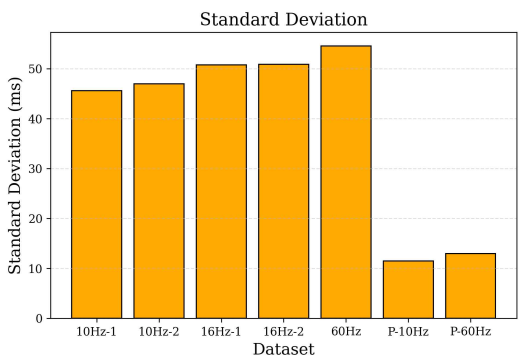

Fig. 8. Comparing standard deviations.

The most notable improvement was observed in the parallel processing configuration in which the standard deviation decreased to 11.48 ms only, a substantial reduction with respect to the values obtained from the original system. This reduction highlights the improved stability of the system, as parallel processing effectively eliminated the fluctuations caused by processing delays.

4.4 Jitter

Jitter measurements were obtained to evaluate short-term latency fluctuations, which are particularly important for real-time systems. As presented in Figure 9, the jitter values follow a similar pattern to the standard deviation results. The 10 Hz datasets exhibited less jitter than the 16 Hz and 60 Hz datasets, confirming that low flicker rates increased performance stability.

As mentioned previously, the introduction of parallel processing significantly improved system stability. The jitter value of the parallelized system was reduced to 12.67 ms, a considerable improvement over the highest jitter value of 68.6 ms recorded using the 60 Hz dataset. This result emphasizes the impact of parallel programming in maintaining consistent response times during continuous data flow.

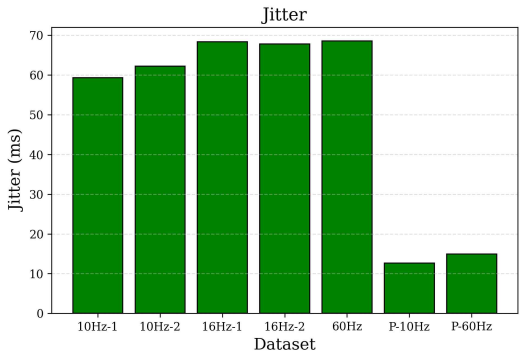

Fig. 9. Comparison of the jitter values.

4.5 Bit Error Rate

The BER performances of the systems were determined to assess the reliability of data transmission across all configurations. As depicted in Figure 10, the 10 Hz datasets maintain lower BER values than the 16 Hz and 60 Hz counterparts. This aligns with

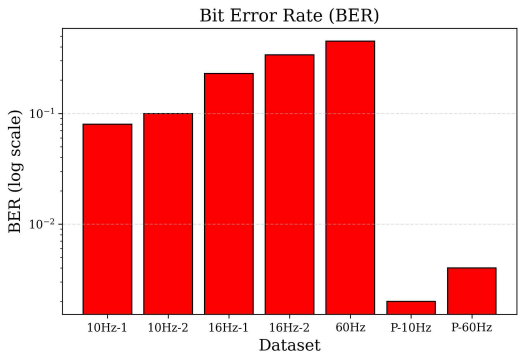

Fig. 10. Comparison of the BER values.

expectations, as low refresh rates improved synchronization between the LED matrix and the frame capture process of the camera.

However, the greatest improvement in BER performance was observed in the parallelized system, with a BER of 0.002, drastically lower than any of the non-parallelized configurations. This result suggests that the reduced latency and improved timing stability directly improved synchronization and minimized bit errors.

V. Conclusion

This study introduced an enhanced OCC performance measurement platform to overcome the performance limitations encountered in resource-constrained edge computing environments. Parallel programming techniques enable the system to effectively address the latency and stability issues observed in the initial design.

The most significant outcome of this study is the substantial reduction in system latency. In the original design, the lowest and highest latencies recorded were 94.1 and 161.7 ms using the 10Hz-1 and 60 Hz datasets, respectively. After implementing parallel processing, the improved system achieved an impressive latency reduction, with an average latency of 30.09 ms. This marks a 68% reduction compared to the lowest latency of the non-parallelized system and demonstrates a remarkable improvement of around 81% in relation to the highest latency recorded. The substantial reduction in latency confirms the effectiveness of parallel programming in optimizing CPU utilization and eliminating processing bottlenecks without relying on GPU acceleration.

Overall, this study demonstrates that integrating parallel programming into OCC systems offers a powerful solution for improving latency and stability and maintaining reliable data accuracy. Owing to the reduced latency from 94.1 ms (best-case) and 161.7 ms (worst-case) to 30.09 ms only with a BER of 0.002, the improved system effectively satisfies the demands of real-time OCC performance in resource-constrained environments.

Future work should explore adaptive parallelization techniques that dynamically allocate CPU resources based on data load conditions to improve efficiency under variable traffic intensity. Additionally, exploring alternative image processing libraries or hardware acceleration can further reduce latency. Furthermore, adaptive flicker rate algorithms may provide a means to dynamically adjust LED refresh rates to balance latency reduction with data accuracy.

References

[1] A. Celik, I. Romdhane, G. Kaddoum, and A. M. Eltawil, “A top-down survey on optical wireless communications for the internet of things,” *IEEE Commun. Surv. & Tuts.*, vol. 25, no. 1, pp. 1-45, Firstquarter 2023. (https://doi.org/10.1109/COMST.2022.3220504)

[2] S. Aboagye, A. R. Ndjiongue, T. M. N. Ngatched, O. A. Dobre, and H. V. Poor, “RIS-assisted visible light communication systems: A tutorial,” *IEEE Commun. Surv. & Tuts.*, vol. 25, pp. 251-288, 2023.

[3] W. Liu, J. Ding, J. Zheng, X. Chen, and C.-L. I, “Relay-assisted technology in optical wireless communications: A survey,” *IEEE Access*, vol. 8, pp. 194384-194409, 2020.

[4] D. Gamero, et al., “Scalability testing approach for Internet of Things for manufacturing SQL and NoSQL database latency and throughput,” *J. Computing and Inf. Sci. Eng.*, vol. 22, no. 6, p. 060901, Dec. 2022.

(https://doi.org/10.1115/1.4055733)

[5] O. S. Sitanggang, V. L. Nguyen, H. Nguyen, R. F. Pamungkas, M. M. Faridh, and Y. M. Jang, "Design and implementation of a 2D MIMO OCC system based on deep learning," *Sensors*, vol. 23, no. 17, p. 7637, 2023. (https://doi.org/10.3390/s23177637)

[6] P. Zhang, Q. Wang, Y. Yang, Y. Wang, Y. Sun, W. Xu, J. Luo, and L. Chen, "Enhancing the performance of optical camera communication via accumulative sampling," *Optics Express*, vol. 29, no. 12, pp. 19015-19023, 2021. (https://doi.org/10.1364/OE.430503)

[7] Y. Babuji, A. Woodard, Z. Li, D. S. Katz, B. Clifford, R. Kumar, L. Lacinski, R. Chard, J. M. Wozniak, I. Foster, M. Wilde, and K. Chard, "Parsl: Pervasive parallel programming in Python," in *Proc. 28th Int. Symp. High-Performance Parall. and Distrib. Computing*, pp. 25-36, Phoenix, AZ, USA, Jun. 2019. (https://doi.org/10.1145/3307681.3325400)

Tae Hyun Kim

2024~Present : Researcher, Defense Technology Specialized Research Lab.

2021~Present : B.S. candidate, School of Electronic Engineering, Kookmin University

<Research Interests> AI., Communications, Automotive Control

[ORCID:0009-0000-3169-2706]

Yeong Min Jang

1985 : B.S. degree, Kyungpook National University

1987 : M.S. degree, Kyungpook National University

1999 : Ph.D. degree, University of Massachusetts

2002~Present : Professor, School of Electrical Engineering, Kookmin University

<Research Interests> AI, OWC, FSO, OCC, Internet of energy, Sensor Fusion

[ORCID:0000-0002-9963-303X]