Bus Indoor Situation Monitoring System Based on Congestion Model Using Lightweight Platform

Dong Hyun Kim, Yun Seob Kim, and Jong Deok Kim*
Department of Computer Science and Engineering
Pusan National University
Busan, South Korea
e-mail: {dhkim1106, giet278, kimjd}@pusan.ac.kr

***Abstract*— The utilization of data is becoming increasingly prevalent across various domains including public services, security, transportation, marketing, and more, leading to a growing interest in data utilization. Especially in the case of public buses, which are heavily used by many people, the importance of congestion is increasingly recognized due to its direct correlation with safety and the potential for numerous associated problems. The bus indoor situation monitoring system aims to predict bus interior congestion and ensure passenger safety through the use of Internet of Things (IoT) and artificial intelligence technology. This paper designs and implements a bus indoor situation monitoring system based on artificial intelligence to predict congestion inside the bus, demonstrating its practicality.**

Keywords-AI; Congestion model; Indoor situation monitoring; Lightweight platform.

I. INTRODUCTION

The utilization of data is expanding into various fields such as public services, security, transportation, and marketing, and there is a growing interest in data utilization. This paper collects and utilizes data related to public transportation among various application domains.

This paper focuses on collecting data and designing systems for buses, which have a broad range of applications in public transportation. The system identifies passenger movement direction, travel duration, and peak usage hours through cameras installed on the bus. Due to the ongoing occurrence of various forms of incidents inside buses, there has been an increasing demand for camera installations. By using cameras to understand the situation inside the bus, it is possible to design various application systems. Therefore, the data available from both inside and outside the bus includes factors such as the number of passengers inside the bus, congestion level, and occurrence of incidents. Based on this data, public agencies can utilize it in fields, such as tourism and security, while bus users can benefit from smooth bus travel by understanding the number of passengers inside the bus by time slots. Then, bus administrators can utilize incident and accident monitoring for accident prevention.

This paper aims to predict congestion using passenger count via cameras installed inside the bus, employing a lightweight platform. In Section 2, we will discuss the models used for object detection, the algorithms employed for tracking, and the protocols used for multimedia stream transmission in the 'Related Work' section. Section 3 will describe the actual development structure and implementation details. In Section 4, we will explain the performance evaluation, and in Section 5, we will present the conclusions and future work.

II. RELATED WORK

The bus indoor situation monitoring system based on a congestion model using a lightweight platform utilizes YOLO, DeepSORT, Real Time Streaming Protocol (RTSP), HTTP Live Streaming (HLS), and Message Queuing Telemetry Transport (MQTT) technologies as component technologies.

A. YOLO (You Only Look Once)

YOLO was introduced at the Computer Vision and Pattern Recognition (CVPR) conference in 2016 [1]. This technology enables real-time object detection and overcomes the drawbacks of traditional CNN-based methods, providing high-accuracy object detection even at high speeds. YOLO uses a 1-stage detection method that processes both localization and classification of objects in a single step.

The inference process of YOLO can be broadly divided into two parts. Firstly, through Bounding Box Regression, it infers the location information of objects. In Figure 1, this involves generating two bounding boxes for each grid cell and calculating the class scores for those boxes. In Figure 2, these class scores indicate how well a specific class fits within the bounding box and the probability of that object appearing. Secondly, in the Classification stage, it classifies the bounding boxes based on the class scores. Boxes with scores lower than a certain threshold are excluded, and after applying the Non-Maximum Suppression (NMS) algorithm to remove overlapping boxes, the boxes are classified into the class with the highest score [2].

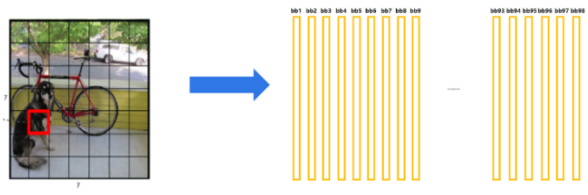

Figure 1. Structure diagram of class specific confidence scores.

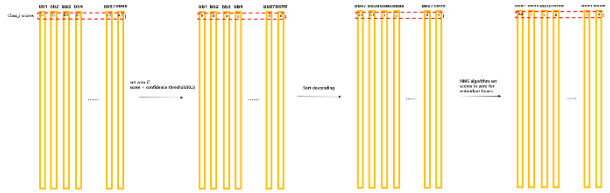

Figure 2. Concept diagram of classification stage.

B. *Deep Simple Online and Realtime Protocol (DeepSORT)*

DeepSORT goes beyond simple object detection by assigning IDs to each object, enabling them to be distinguished from one another and allowing for more precise operations than just basic object detection by utilizing movement path tracking data. Simple Online and Realtime Tracking (SORT) is one of the technologies used for object tracking in this process [3].

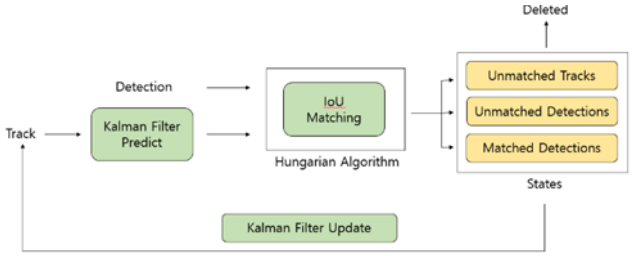

Figure 3. Concept diagram of SORT.

The object tracking algorithm SORT is structured in the order depicted in Figure 3. It involves predicting the position of objects being continuously tracked from the previous frame to the next frame. The predicted values are then compared with the objects detected by YOLO in the current frame to find the optimal match among the currently detected objects. In Figure 4, the 3-state configuration diagram of SORT is explained. Depending on the processing result, it is categorized into three states: matched, new detection, or deleted. If matched or newly detected, the information of the tracked object is updated or created with new values; otherwise, it is deleted.

However, SORT exhibits poor performance when encountering occlusion where objects overlap, re-entry when objects leave and re-enter the frame, and appearance changes due to noise. To address this issue, DeepSORT has been proposed.

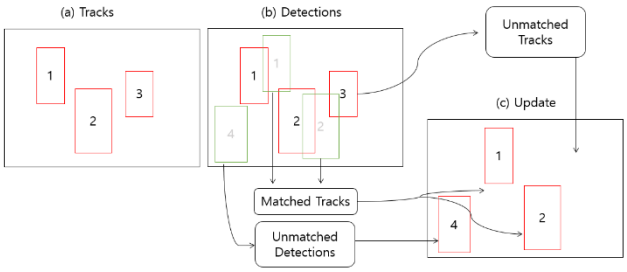

Figure 4. 3-State configuration diagram of SORT.

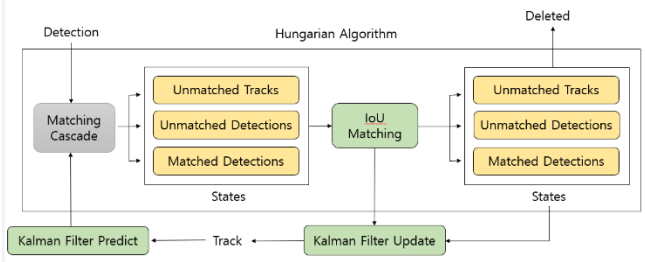

Figure 5. Concept diagram of DeepSORT.

To enhance the performance of SORT, DeepSORT improves the data association process. Figure 5 illustrates the data association process in DeepSORT, where Matching Cascade is introduced. Matching Cascade enhances accuracy by incorporating appearance and distance information of objects into the tracking process. Ultimately, DeepSORT is utilized to achieve higher accuracy compared to SORT and reduce the occurrence of multiple detections of the same object [4].

C. *Real Time Streaming Protocol (RTSP)*

RTSP is a control protocol designed to control streaming servers. It serves the purpose of control rather than transmitting media data.

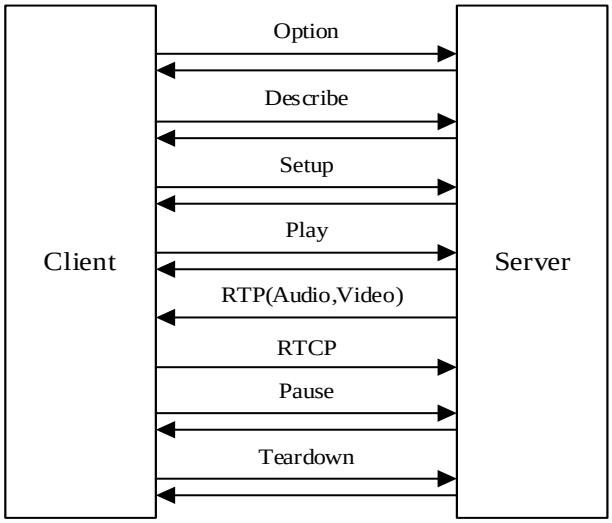

Figure 6. Streaming control process of RTSP.

Figure 6 depicts the overall process of the streaming control protocol, where OPTIONS retrieves available commands. DESCRIBE requests media information from the server, while SETUP retrieves information on how it should be transmitted. Subsequently, PLAY initiates media playback. This allows RTP and RTCP to transmit actual media data and quality control data. Additionally, PAUSE can temporarily halt streaming, and TEARDOWN terminates all sessions [5].

D. HTTP Live Streaming (HLS)

HLS is an Apple protocol for video streaming introduced in 2009, used for delivering media content over the internet. In Figure 7, the operation process of HLS is depicted. This involves encapsulating streaming data into MPEG-2 Transport Stream, segmenting it, and then transmitting it. The Stream Segmenter divides the MPEG-2 Transport Stream into segments based on a set time interval and generates an m3u8 file containing metadata about the segments. Subsequently, the client uses HTTP to request the ts files and metadata from the server for streaming [5].

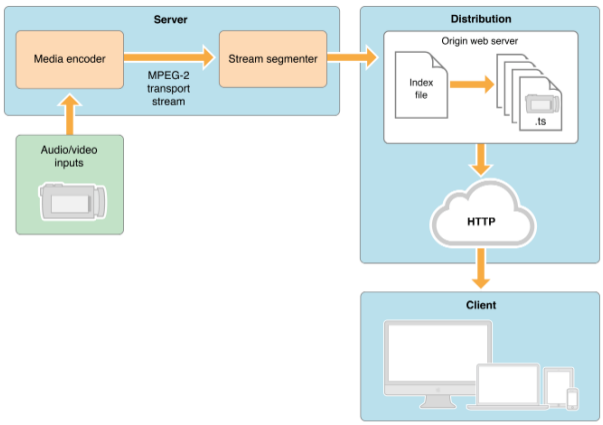

Figure 7. Operation structure of HLS.

E. Message Queue Telemetry Transport (MQTT)

MQTT is a lightweight messaging protocol based on the Publish-Subscribe model, developed in 1999 for exchanging messages between IoT devices and systems. It is widely used due to its suitability for devices with limited resources, such as those found in IoT. MQTT was chosen for use inside the bus because it can be advantageously utilized in limited bandwidth and low-power environments

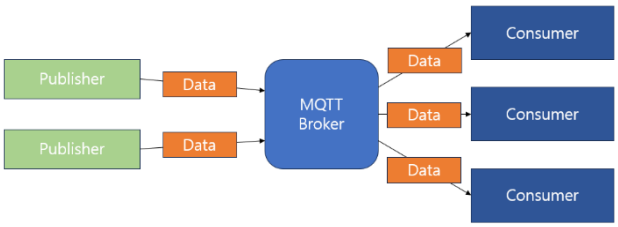

Figure 8. Operation structure of MQTT.

In Figure 8, MQTT operates on a Publish/Subscribe model, where devices publish to specific topics and other devices communicate by subscribing to desired topics. Devices can subscribe to one or more topics, which are organized hierarchically as sub-topics. Additionally, devices can choose the Quality of Service (QoS) level to determine the reliability of message delivery. QoS 0 indicates 'At most once' delivery, meaning messages are simply transmitted through the topic. QoS 1 indicates 'At least once' delivery, where the subscribing client may receive the message at least once, and if it's uncertain whether the message was received, it will be resent a predetermined number of times. Lastly, QoS 2 indicates 'Exactly once' delivery, providing maximum reliability by applying strict handshaking to ensure the message is delivered exactly once [6][7].

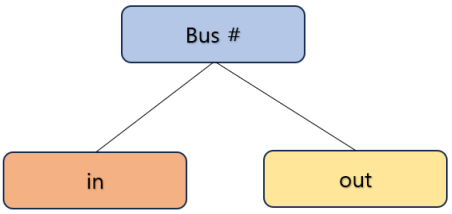

Figure 9. Hierarchical structure of topic.

In Figure 9, the MQTT protocol was utilized to efficiently communicate on the lightweight platform, Jetson [8]. Topics were structured as /{bus#}/in and /{bus#}/out to publish the boarding and alighting passenger counts for each bus. Additionally, to enhance reliability while considering duplicates, the QoS level was set to 1, as data is published only when the passenger count changes, rather than periodically.

III. Design and Implementation

The system proposed in this paper was initially designed assuming a single bus. However, as the research progressed, multiple bus scenarios were assumed to increase the applicability by implementing them as closely as possible to real-world situations. In Figure 10, the overall system configuration is depicted. As a result, the use of Jetson and communication with Jetson became inevitable. Video processing within each bus was conducted through Jetson, while data storage and transmission were handled using Amazon Web Services (AWS) IoT and S3.

Previously, in traditional web development, the front-end and back-end frameworks were separated. However, as the data to be transmitted was not extensive and complex UI/UX was not necessary for directly providing data to users, the system's architecture was changed to implement the front-end within Django.

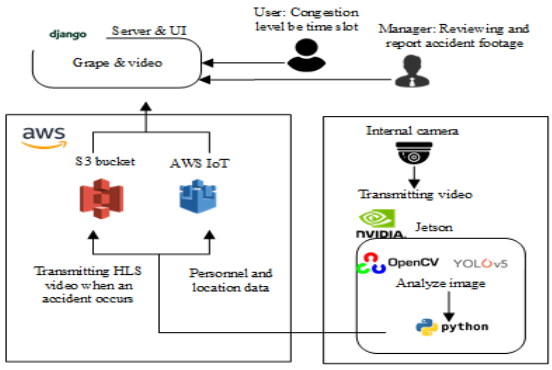

Figure 10. The overall system configuration.

A. Congestion measurement model

The bus congestion measurement model is given by (1), where the bus congestion is denoted by R_C, the number of seats by N_S, the number of handrails by N_H, and the number of passengers by P_N.

$$R_C = \frac{P_N}{N_S + N_H} \times 100 \quad (1)$$

The bus interior footage predominantly captures boarding and alighting activities, as well as the central area of the bus, making it difficult to ascertain the number of seats and handrails. Therefore, the denominator of the model was set to 110% of the seating capacity. For example, for Hyundai Elec City buses in Korea, this would correspond to 54 passengers.

The congestion level of the bus is defined into four categories: 'Spacious,' 'Normal,' 'Crowded,' and 'Very Crowded,' based on the real-time number of passengers on board. The index for these four categories is shown in Table I. While individuals may perceive congestion differently, the levels were established based on general situations.

TABLE I. BUS CONGESTION STATUE INDEX

Spacious	Normal	Crowded	Very Crowded
50% or less	50 ~ 70%	70 ~ 100%	Over 100%

The percentage criteria were calculated based on the bus with the highest number of seats. For 'Spacious,' it refers to before all seats are occupied; 'Normal' indicates roughly half of the standing capacity being occupied, and beyond that, it is divided into 'Crowded' when the seating capacity is filled, and 'Very Crowded' when it exceeds the seating capacity.

B. People counting

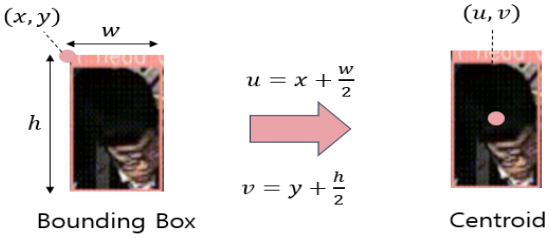

Figure 11. Concept of counting people using bounding box.

To determine the number of passengers inside the bus, we utilized the information from the Bounding Boxes (BBox) of the objects. As shown in Figure 11, objects inherently possess the coordinates of the top-left corner. We adjusted the position of these coordinates to the center of the BBox, and based on whether the baseline passed through the center of the head, we calculated the number of passengers.

Based on the information being tracked, we determined changes in the number of passengers in the current frame compared to the previous frame, depending on whether the objects crossed the baseline. As illustrated in Figure 12, the change in the number of passengers before and after crossing the baseline can be observed.

To definitively identify individuals boarding the bus after climbing the stairs and exclude those outside, we positioned the baseline at the end of the staircase where it meets the bus structure. Consequently, passengers who have fully boarded the bus are counted, while individuals outside remain undetected as they are obstructed by the structure and cannot pass through the baseline. Furthermore, to avoid duplicate counting, a unique ID is assigned during tracking, ensuring that objects previously processed are not counted again if they cross the baseline.

Figure 12. Before and after passengers cross the baseline.

C. Communication between Jetson and the cloud.

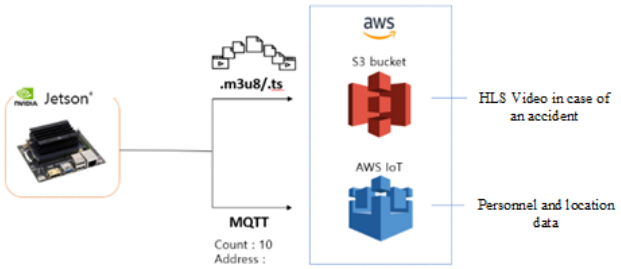

Figure 13. Operation process of Jetson.

In Figure 13, operation process of Jetson is depicted. Developed by NVIDIA, Jetson is an AI computing platform designed for Graphic Processing Unit (GPU)-accelerated parallel processing. Installing Jetson on each bus enables real-time analysis of incoming footage, extracting necessary information, and communicating with AWS IoT and S3. In Figure 14, the operation process of AWS IoT is depicted. This allows for alleviating the burden of transmitting large video data itself and accessing AWS IoT and S3 from clients to utilize congestion graphs and accident footage.

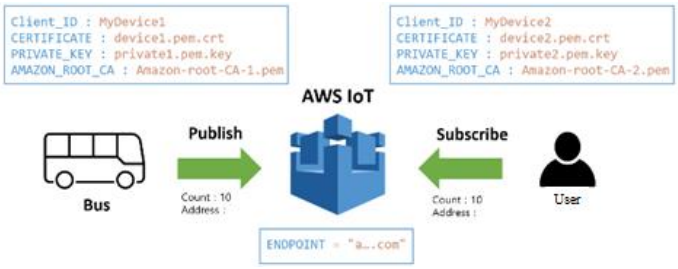

Figure 14. The operation process of AWS IoT.

Devices used within public transportation and by users communicate with respective brokers. The central broker in Figure 15 acts as a system that coordinates messages between buses and user clients, handling tasks such as message reception and filtering, identifying clients subscribing to each message, and message transmission. Additionally, it can be observed that each device accesses the broker with different certificates.

IV. PERFORMANCE ANALYZE

Figure 15. Jetson connection screen.

In Figure 15, the screen displayed when Jetson is running is shown, allowing real-time monitoring of the number of passengers boarding and alighting inside the bus.

Figure 16. Human recognition result.

Figure 16 shows the video being processed for object detection inside the Jetson. Object detection is performed based on the received video, and the counts for boarding and alighting individuals are tallied separately. Figure 16 depicts the result of detecting human objects.

TABLE II. THE ACCURACY OF PEOPLE DETECTION

	Bus number 1	Bus number 2	Bus number 3	Bus number 4	accuracy (%)
Get on	9/14	8/11	5/8	11/14	70.21
Get off	0/0	4/5	13/15	16/19	84.62

After testing with several videos, the accuracy of passenger detection is as shown in Table II. While there is a certain percentage difference between manual counting and counting using YOLOv5, it maintains an accuracy of over 70%. With improvements in video quality and additional data collection and training, it is expected to achieve even more accurate results.

TABLE III. THE ACCURACY OF PEOPLE DETECTION

	Bus number 1	Bus number 2	Bus number 3	Bus number 4	accuracy (%)
Get on	9/14	8/11	5/8	11/14	70.21
Get off	0/0	4/5	13/15	16/19	84.62

V. CONCLUSION AND FUTURE WORK

Various application services developed using data are actively underway. Especially in the transportation sector such as buses, diverse application services are being developed, making our lives more convenient.

The bus indoor environment monitoring system based on a lightweight platform and congestion model utilizes vision-based artificial intelligence technology to predict the congestion level inside the bus. In this paper, video data was collected and stored by installing cameras inside the bus. To calculate congestion levels, vision technology was used to estimate passenger counts during boarding and alighting. Passenger counting utilized vision technologies such as YOLOv5 and OpenCV, and data was transmitted using technologies like RTSP, HLS, and MQTT, assuming various bus scenarios. Leveraging these technological components, we designed a congestion model-based bus indoor environment monitoring system using a lightweight platform and validated its practicality using Jetson, considering its application in real buses. In future work, we will focus on enhancing system accuracy and efficiency through several technical improvements. We plan to integrate advanced AI models such as YOLOv7 and EfficientDet for more precise passenger counting and congestion prediction. Additionally, we will adopt more powerful edge computing devices like the NVIDIA Jetson Xavier to reduce latency and improve real-time processing. Incorporating IoT sensors for monitoring environmental factors such as temperature, humidity, and CO2 levels will provide a more comprehensive solution.

ACKNOWLEDGMENT

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No. 2020R1I1A3065947) and by a Korea Institute for Advancement of Technology (KIAT) grant funded by the Korea Government (MOTIE) (P0017006, HRD Program for Industrial Innovation).

REFERENCES

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779-788, 2016.

[2] J. Terven, and D. Cordova-Esparza, D. (2023). A comprehensive review of YOLO: From YOLOv1 to YOLOv8 and beyond. arXiv preprint arXiv:2304.00501.

[3] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking" IEEE International Conference on Image Processing (ICIP), Phoenix, AZ, USA, pp. 3464-3468, Sep. 2016. https://doi.org/10.1109/ICIP.2016.7533003.

[4] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric", IEEE International Conference on Image Processing (ICIP), Beijing, China, pp. 3645-3649, Sep. 2017. https://doi.org/10.1109/ICIP.2017.8296962.

[5] R. Pantos and W. May, "HTTP Live Streaming," IETF RFC 8216, Aug. 2017. [Online] Available: https://www.ietf.org/rfc/rfc8216.tx.

[6] P. Julio, “MQTT Performance Analysis with OMNeT++,” M.S. thesis, IBM Zurich Research Laboratory, Institut Eurecom, Sep. 2005.

[7] The MQTT protocol, http://www.mqtt.org/, last access June 2024.

[8] M. I. Uddin, M. S. Alamgir, M. M. Rahman, M. S. Bhuiyan, and M. A. Moral, “AI traffic control system based on deepstream and IoT using NVIDIA Jetson nano”, In 2021 2nd International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST), pp. 115-119, IEEE, 2021.