A Manipulation Scheme of Intelligent Moving Objects through an AI-Driven Management Framework

Jisuk Chae, Mose Gu, Yoseph Ahn, Jiwon Suh, Jaewon Hwang, Jooseung Nam, Yeonjoo Lee and Jaehoon (Paul) Jeong
Department of Computer Science & Engineering,
Sungkyunkwan University, Suwon, Republic of Korea
Email:{sue030124, rna0415, ahnjs124, sjw6136, carjh0621, skawntmd01, pphtz, pauljeong}@skku.edu

***Abstract*—Intent-Based Networking (IBN) enables goal-driven control of systems, yet its application to indoor autonomous driving remains limited. This paper presents a perception-centric extension of the framework for Interface to In-Network Computing Functions (I2ICF) using a ROS2-based robotic platform. To ensure stable real-time operations, we resolve conflicts between ROS2 and Flask with a multi-processing architecture, while a fine-tuned YOLOv8s model improves object detection in complex indoor environments. The system, composed of an Intelligent Moving Object (IMO), a desktop client, and a controller. It transforms raw sensor data into semantically rich records for monitoring and analysis. Results demonstrate the reliable translation of user intents into high-level policies for safe and efficient indoor navigation.**

***Index Terms*—Intent, I2ICF, Kubernetes, Moving Object, Object Detection**

I. Introduction

The advent of Intent-Based Networking (IBN) [1] has opened new opportunities for simplifying network and system management by allowing users to specify what they want rather than how their intents should be implemented. Traditionally, intent translation frameworks have been designed for large-scale telecommunication networks, focusing on policy orchestration across heterogeneous devices. However, as computing capabilities increasingly migrate toward network edges, enabling seamless control and manipulation of edge devices such as autonomous robots and Intelligent Moving Objects (IMOs) has become necessary. The Interface to In-Network Computing Functions (I2ICF) [2] [3] in the Internet Engineering Task Force (IETF) proposed a framework for a user to control and manage a Moving Object (MO) according to a user intent. While advances in autonomous driving technology have enabled the use of Mobility Objects (MOs)-such as robots and robot cars, enhancing the safety and efficiency of autonomous driving in indoor environments requires a tight integration of sensor data collection and transmission, high-level policy control, and real-time monitoring. Therefore, a reliable system for intent-based autonomous driving control that meets the I2ICF requirements is needed.

This paper aims to extend the applicability of the I2ICF framework by maintaining its structure and advantages while adding functions required for indoor autonomous driving, such as object detection and odometry information transmission to an edge server. To achieve this, we applied a ROS2-enabled robot car platform equipped with multiple onboard sensors (e.g., camera and odometry) as an IMO, since it provides an accessible yet sufficiently complex environment for validating intent-driven control. This platform supports real-time sensor data transmission to an edge server and integrates seamlessly with our object detection function using a YOLOv8s fine-tuned model [4] specialized for indoor environments, thereby enabling safer and more efficient indoor navigation.

While advances in autonomous driving technology have enabled the use of MOs, the unique constraints of indoor environments—such as narrow corridors, irregular obstacles, and inconsistent lighting—pose significant challenges for perception and navigation. Achieving safety and efficiency in such scenarios requires robust object detection, accurate odometry, and real-time data transmission, and monitoring. However, existing I2ICF-oriented designs remain limited in two key aspects:

1) **System-level instability**: Middleware conflicts (e.g., ROS2–Flask integration) often lead to blocking communication and degraded responsiveness.
2) **Perception robustness**: Pre-trained detection models lack adaptation to the specific visual and structural features of indoor environments, resulting in unreliable decision-making.

Therefore, a reliable intent-based robotic control system that resolves these limitations and aligns with I2ICF requirements is critically needed.

To extend the applicability of the I2ICF framework into indoor autonomous driving, we implemented a ROS2-enabled robotic platform (i.e., LIMO ROS2-based robot car) [5] equipped with a camera and odometry sensors. The platform streams real-time sensor data through a Flask-based transmission server integrated with a multi-processing architecture, which decouples a ROS2 subscriber from a web server to ensure stable and parallel execution.

On a client, a fine-tuned YOLOv8s model specialized for indoor environments performs object detection, producing semantically rich perception outputs. These outputs are synchronized with odometry data and encapsulated into JSON

payloads, including object metadata and annotated images. The controller server archives records including time-aligned JSON logs and JPEG images, enabling monitoring, assurance, and post-mission analysis.

The main idea of this paper is a modular system for autonomous data acquisition, designed to operate within an intent-based framework. The system enables a user to define an intent such as a high-level data collection mission (e.g., "survey the main corridor for safety equipment"), which the robotic platform then fulfills by autonomously navigating the designated area. The core of this fulfillment process is a powerful perception pipeline. We utilize a YOLOv8s model, specifically fine-tuned for the indoor environment, to transform the robot's raw camera feed into structured, semantic information in real-time.

This advanced object detection capability enables not only streaming live video and odometry but also identifying and logging specific objects of interest, providing richer data stream compared to simple telepresence. All of this multi-modal data—including the annotated images, object lists, and navigation logs—is transmitted from the desktop client as an edge server to a central server. This enables both real-time monitoring of the mission's progress and the creation of a comprehensive, archived dataset for later analysis and assurance. Therefore, our main idea is not just the creation of a mobile robot, but a complete, perception-centric ecosystem that translates a user intent into a high-level policy for the collection of verifiable, semantically-rich data records from the physical world.

The main contributions of this study are as follows:

- **Implementing Manipulation Scheme of IMO for I2ICF Framework:** We designed a perception-centric, intent-based data acquisition system as a distributed pipeline consisting of an IMO, an edge server, and a cloud server. This architecture transforms raw sensory inputs into semantically rich information, integrating object detection results with navigation data and then archiving them as synchronized JSON and image records for the I2ICF framework for monitoring and analysis purposes.
- **ROS2–Flask Conflict Resolution with Multi-Processing:** We resolved the long-standing conflict between ROS2 and Flask by introducing a multi-processing architecture that decouples the ROS2 subscriber from the Flask-based web server. This design ensures stable, parallel data transmission without blocking or bottlenecks, thereby enabling reliable real-time navigation and monitoring in indoor environments.
- **Indoor-Specific Object Detection with Fine-Tuned YOLOv8s:** We presented an indoor-specific perception pipeline by fine-tuning the YOLOv8s model with a custom dataset that captures the unique characteristics of indoor environments, including narrow corridors, complex obstacle arrangements, and diverse lighting conditions. This adaptation significantly improves detection accuracy and robustness, supporting safer and more reliable autonomous driving decisions.

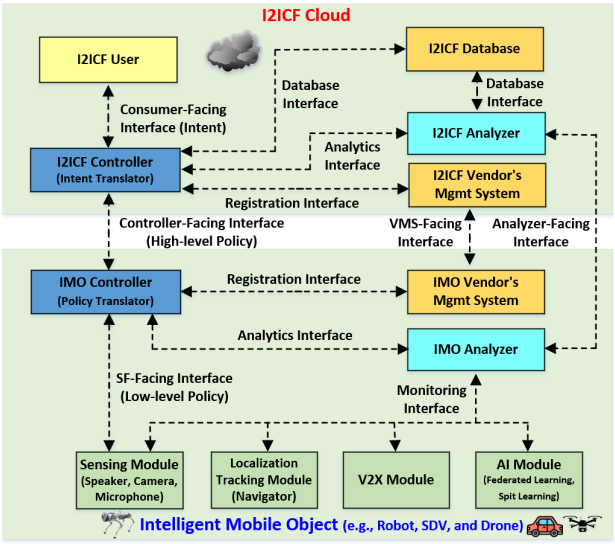


Fig. 1: Interfaces and Components of the I2ICF Framework

The remaining of this paper is structured as follows. Section II summarize and analyzes related work. Section III explains the design of our IMO manipulation scheme. Section IV present the implementation and evaluate the performance of our scheme. Section V concludes this paper with future work.

II. Related Work

Intent-Based Networking (IBN) is a paradigm that abstracts network configuration by allowing operators to define a natural language-based intent rather than detailed device commands [1]. An IBN system automatically translates this intent into executable configurations (Intent Fulfillment) [6] and continuously verifies that the operational state matches the desired objectives (i.e., Intent Assurance) [1], [7]. Standardization efforts, such as those in the IETF and IRTF NMRG, have formalized IBN concepts and definitions and also explored their applications in various domains [8], [9].

The Interface to In-Network Computing Functions (I2ICF) which is illustrated in Fig. 1 extends the IBN concept beyond traditional networking to encompass various computing devices functions, including robotics, IoT, and autonomous vehicles [2]. In this framework, an Intent Translator can accept natural language input from a user, convert it into a high-level policy as structured task specifications, and deliver them to computing functions within the network [2], [3] . This approach enables heterogeneous systems—such as software-defined vehicles (SDVs)—to receive high-level human instructions and execute them through edge or servers resources with minimal manual configuration.

In SDV environments, fulfilling such high-level intents often requires advanced perception capabilities. One of the most critical capabilities of these is object detection, which allows the system to perceive and interpret its surroundings in real time, enabling safe navigation, hazard detection, and task-specific actions.

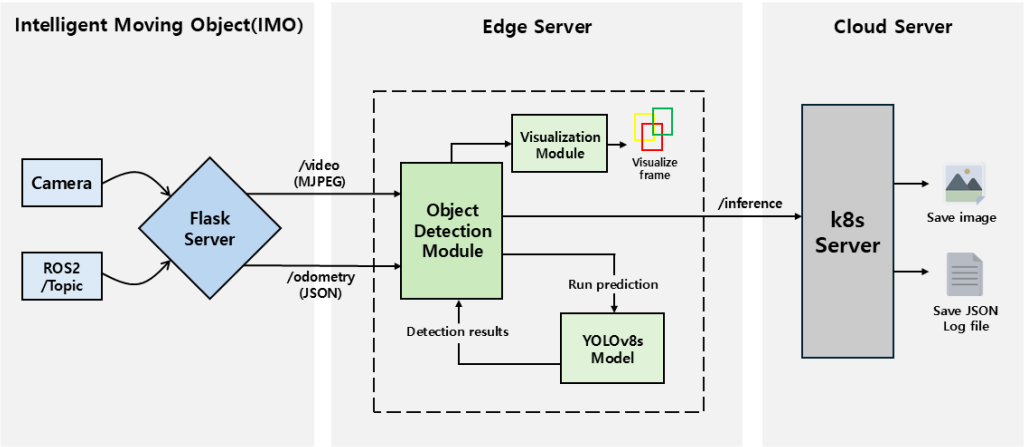


Fig. 2: Main Figure

The YOLO (You Only Look Once) series has been a leading approach for real-time detection since its first version in 2015 [10]. Among them, YOLOv8 introduces key improvements such as an anchor-free mechanism, a decoupled head, and the C2f module [4] [11], achieving enhanced speed and accuracy for robotics applications. YOLOv8 has been applied in autonomous driving and robotics with strong performance [11], while variants like YOLOv8-RTDAV [12] and small-object enhanced versions [13] demonstrate its adaptability to domain-specific tasks. In SDV systems, these detectors can operate on edge hardware for real-time perception and integrate with orchestration frameworks such as I2ICF for higher-level decision-making.

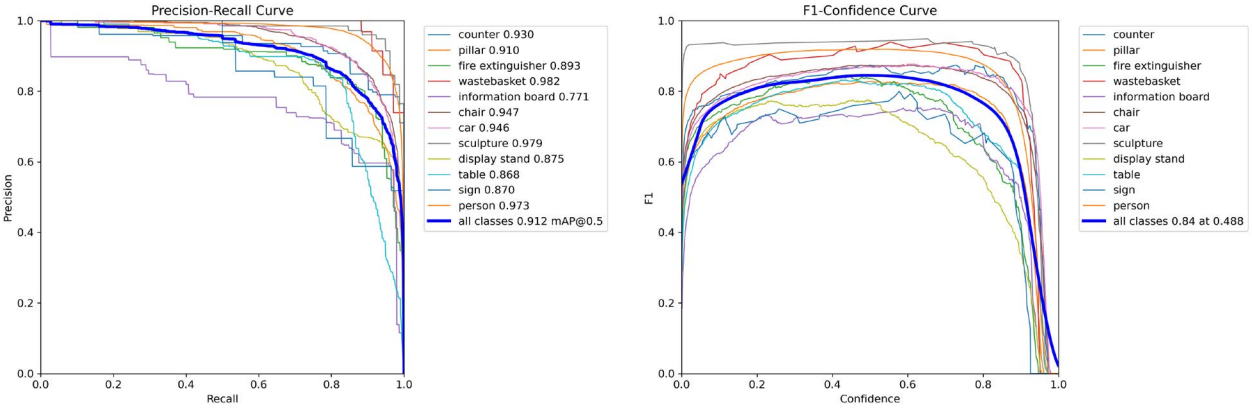
III. Design of IMO Manipulation Scheme

This section presents the core design of our scheme. In Fig. 2, it illustrates the overall workflow of our proposed system module by module. There are three main modules in our system.

- **Intelligent Moving Object (IMO):** On the edge server, data collection and transmission are carried out. The intelligent vehicle captures raw image frames from the onboard camera and subscribes to odometry messages from ROS2 topics. Both data streams are forwarded to an edge server in Desktop in Fig. 2 via a Flask-based multiprocessing server, which publishes the video as MJPEG stream, and odometry data as JSON messages.
- **Edge Server:** On the edge server, the Object Detection Module receives both the image and odometry streams. The incoming frames are processed by a fine-tuned YOLOv8s model, which performs real-time object detection. The detection results are then used by the Object Detection Module. Using this information, the module generates a visualized frame with annotated bounding boxes through the Visualization Module and simultaneously prepares inference results that combine detection outputs with odometry metadata.
- **Cloud Server:** The inference results are transmitted to the cloud server (i.e., Kubernetes server) through the /inference endpoint. The controller server stores detection outputs in two formats: annotated image files for visualization and JSON log files for structured analysis. This dual storage mechanism enables both real-time monitoring of the environment and long-term logging for offline evaluation.

The proposed system operates as a pipeline, processing raw sensor data into archived records. The workflow begins at the IMO, hosted on the robotic platform, where data generation occurs in two parallel, non-blocking processes to ensure real-time responsiveness. One process continuously captures raw image frames from the onboard camera using OpenCV, while another ROS2 subscriber process listens to the odometry sensor and updates the latest odometry message into a shared queue. The Flask application then serves these data: the camera feed is encoded and published as a continuous MJPEG stream, while the odometry data are provided on demand as JSON messages retrieved from the queue without interfering with video streaming.

The process then moves to the edge server, which consumes both streams to perform its core perception tasks. For each incoming frame from the MJPEG stream, the edge server first requests the most recent odometry data to synchronize the visual frame with the robot's physical state. The frame is then processed by the Object Detection Module, where the fine-tuned YOLOv8s model performs inference to identify objects and generate class labels, confidence scores, and bounding box coordinates. Once detection is complete, the results are branched into two paths: the Visualization Module overlays bounding boxes and labels onto the frame to produce annotated outputs for real-time monitoring, while the raw detection results are combined with odometry metadata to form a comprehensive inference log. This log, along with the annotated frame encoded in Base64, is packaged into a single JSON payload.



(a) Precision-Recall curve of Fine-tuned Model

(b) F1 curve of Fine-tuned Model

Fig. 3: Evaluation of Fine-tuning

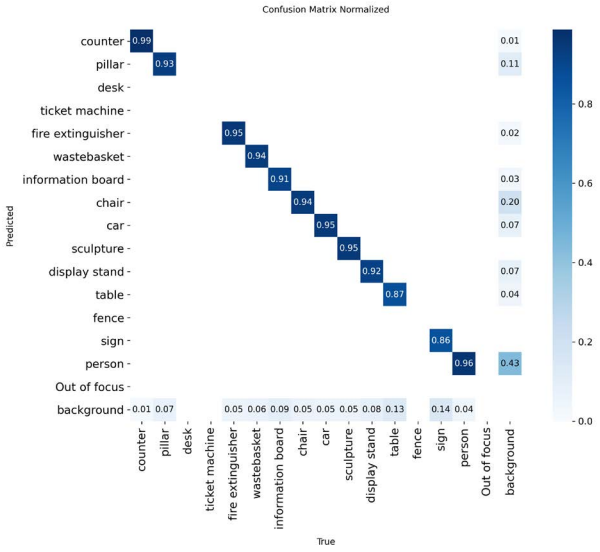


Fig. 4: Confusion Matrix for Fine-tuning Inference

Finally, the JSON payload is transmitted to the inference endpoint of the cloud server. Upon reception, the cloud server performs dual-storage actions: the structured detection and odometry results are archived as JSON log files, and the Base64-encoded annotated images are decoded and saved as JPEG files. This dual-storage mechanism ensures that perception outputs are durably preserved in synchronized formats, enabling both immediate visual verification and long-term analytical evaluation.

IV. Implementation

This section describes the implementation of the proposed I2ICF-based framework for IMOs to demonstrate the feasibility of our architecture.

For the implementation, we used ROS2 Humble [14] for odometry data handling, Flask for HTTP-based streaming, and OpenCV for camera processing at the IMO platform. On the desktop client, we applied the YOLOv8s model [4] fine-tuned with indoor datasets to perform real-time object detection. The server-side framework was deployed on a Kubernetes-based cloud environment, which provided scalable orchestration of containerized components.

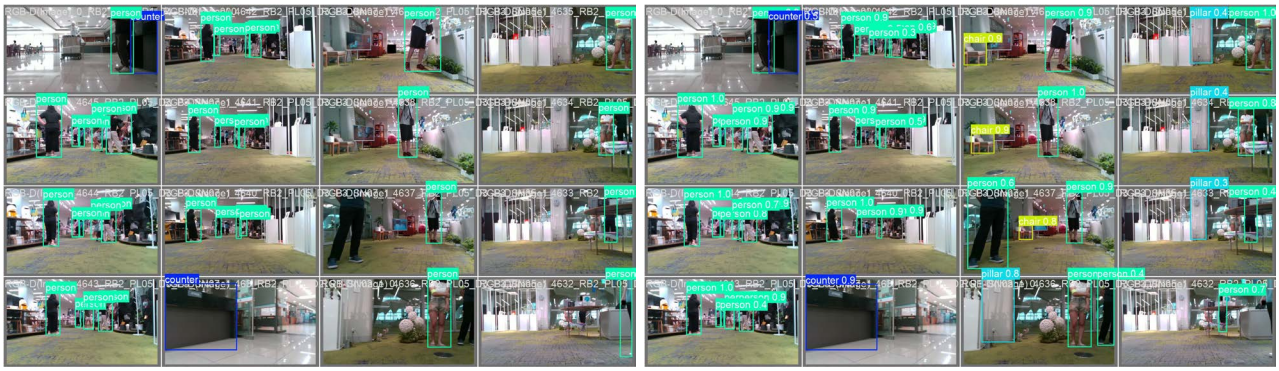
The **IMO** was implemented using the LIMO ROS2 robotic platform equipped with a camera and odometry sensors. A Flask server and ROS2 node were executed in a multiprocessing architecture to avoid process conflicts. The Flask server delivered real-time MJPEG video streams through the video endpoint and JSON-formatted odometry data through the odometry endpoint. This ensured continuous and stable data streaming to the client.

The **Edge Server** acted as the perception engine. To enable robust perception in indoor navigation scenarios, we fine-tuned a YOLOv8s model with a dataset optimized for indoor navigation. We considered several datasets such as ETH [15], HEV-I [16], and UCY [17], but these were not suitable for indoor navigation due to their outdoor or aerial collection conditions. Therefore, we selected the AI Hub dataset [18], which provides robot-perspective indoor driving data, as the most appropriate choice for fine-tuning.

The AI Hub dataset provides indoor, robot-perspective RGB imagery with multi-modal pairs (e.g., RGB JPG and depth PNG) and annotations in JSON format covering bounding boxes, cuboids, and action segments. In total, 150,229 labeled frames were available, from which we sub-sampled approximately 60,000 training images and 6,000 validation images while preserving scene and class diversity. The training was performed for 50 epochs.

We adopted the official 15-class taxonomy (e.g., counter, pillar and desk). The captures span 12 types of indoor venues (e.g., restaurants, exhibitions, terminals, parking lots and churches) and two robot platforms (e.g., quadruped RB1 and wheeled RB2), supporting diverse layouts and viewpoints.

The precision–recall curves in Fig. 3a show that the fine-tuned model achieved an overall mAP@0.5 (i.e., mean Av-



(a) Ground Truth Labels (b) Predicted Bounding Boxes

Fig. 5: Inference of fine-tuned model

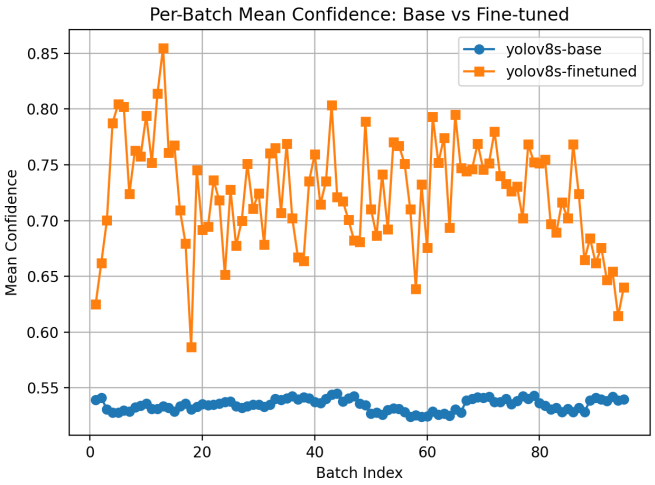


Fig. 6: Comparison of Confidence between baseline and Fine-tuned model

erage Precision when the IoU threshold is fixed at 50%) of 0.912, with consistently high precision and recall across most categories. As shown in Fig. 3b, the F1–confidence analysis indicates that the optimal confidence threshold is around 0.49, yielding a balanced F1 score of 0.84. Furthermore, the normalized confusion matrix in Fig. 4 confirms balanced detection accuracy across major categories while highlighting minor misclassifications in visually similar classes.

At runtime, the edge server consumed the MJPEG video stream and odometry JSON from the robot, performed inference using the fine-tuned YOLOv8s model, and fused the detection results with odometry data. The outputs—bounding boxes, class labels, and confidence scores—were encapsulated into JSON payloads along with Base64-encoded images. These payloads were transmitted per second to the server via REST API.

The Kubernetes-based **Cloud Server** acted as an Analyzer, receiving and storing the JSON payloads and images from the edge server. In the implementation, each component of the I2ICF framework was containerized and deployed as a Pod, while communication between components was handled through Kubernetes Services. All the results were archived as synchronized JSON logs and JPEG images, providing a verifiable dataset for monitoring and intent assurance in the I2ICF framework. The overall implementation flow is illustrated in Fig. 2, where the IMO streams sensing data, the desktop performs in-network perception tasks, and the server maintains the closed-loop assurance [8], [19] mechanism.

V. Performance Evaluation

This section describes the experiment conducted to verify whether the proposed fine-tuned model operates correctly in the actual IMO environment. Fig. 5a shows the ground truth labels of the dataset, while Fig. 5b shows the inference results of the fine-tuned model. Overall, the predicted bounding boxes align well with the ground truth, accurately detecting both the object classes and their locations. Although in some images additional objects were detected or relatively low confidence scores were observed, the overall detection performance remained stable, with the average confidence score of bounding boxes in Fig. 5b being approximately 0.76.

To provide additional validation of the proposed method's effectiveness, a comparative experiment between the baseline and fine-tuned models was conducted using approximately 6,000 test images, with a batch size of 64. For each image, the number of detected bounding boxes and their corresponding confidence scores were obtained. The sum of the confidence scores was divided by the number of bounding boxes to compute the average confidence per image. These values were then aggregated across 64 images to calculate the mean confidence for each batch. This process was repeated for a total of 95 batches, and the resulting per-batch mean confidence values were visualized in Fig. 6 to show the comparison. The results show that the fine-tuned object detection performance is up to 30% higher than the baseline performance. The results displayed in Fig. 7 are a snapshot of the cloud server's logs sent by the edge server. The logs show object detection and

Fig. 7: Object Detection and Odometry Log Transmission

odometry logs, confirming that object detection and odometry data were successfully transmitted without collision.

VI. Conclusion

This paper proposed a perception-centric extension of the I2ICF framework to enable intent-driven autonomous driving in indoor environments. By leveraging a ROS2-based robotic platform, we designed a distributed pipeline that integrates sensing, perception, and data management across an IMO, an edge server, and a cloud server. To address instability caused by ROS2–Flask conflicts, we introduced a multiprocessing architecture that ensures reliable real-time data transmission. Furthermore, by fine-tuning a YOLOv8s model, we achieved improved detection accuracy even in indoor navigation environments. Experimental evaluations demonstrated that our system effectively translated a user intent into a high-level policy for safe and efficient indoor navigation while generating semantically rich logs for monitoring and analysis. The results support the feasibility of applying I2ICF concepts to an indoor IMO pipeline, suggesting potential extensions toward robotics and autonomous systems.

As future work, we will expand our framework by enhancing intent translation mechanisms, supporting multi-robot coordination, and deploying the system over 5G-enabled edge environments to validate scalability in more complex scenarios.

Acknowledgments

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2024-00398199), and (No. RS-2022-II221015, Development of Candidate Element Technology for Intelligent 6G Mobile Core Network).

References

[1] A. Clemm, L. Ciavaglia, L. Z. Granville, and J. Tantsura, "Intent-Based Networking - Concepts and Definitions," RFC 9315, Oct. 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc9315

[2] J. P. Jeong, Y. C. Shen, Y. Ahn, Y. Kim, E. P. D. Jr., and K. Yao, "Interface to In-Network Computing Functions (I2ICF): Problem Statement," Internet Engineering Task Force, Internet-Draft draft-jeong-opsawg-i2icf-problem-statement-00, Mar. 2025, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-jeong-opsawg-i2icf-problem-statement/00/

[3] J. P. Jeong, Y. C. Shen, Y. Ahn, Y. Kim, E. P. D. Jr., and K. Yao, "A Framework for the Interface to In-Network Computing Functions (I2ICF)," Internet Engineering Task Force, Internet-Draft draft-jeong-opsawg-i2icf-framework-00, Mar. 2025, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-jeong-opsawg-i2icf-framework/00/

[4] Ultralytics, "Yolov8 by ultralytics," https://github.com/ultralytics/ultralytics, 2023.

[5] AgileX Robotics, "Limo ros2: Ros2 mobile robot platform," 2025, accessed: 2025-08-18. [Online]. Available: https://global.agilex.ai/products/limo-ros2

[6] M. Gu, J. P. Jeong, and Y. Ahn, "An Intent Translation Framework for IoT Networks," Internet Engineering Task Force, Internet-Draft draft-gu-nmrg-intent-translator-01, Jul. 2025, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-gu-nmrg-intent-translator/01/

[7] A. Leivadeas and M. Falkner, "A survey on intent-based networking," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 625–655, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/9925251

[8] J. P. Jeong, Y. Ahn, M. Gu, Y. Kim, and P. Jung-Soo, "Intent-Based Network Management Automation in 5G Networks," Internet Engineering Task Force, Internet-Draft draft-jeong-nmrg-ibn-network-management-automation-06, Jun. 2025, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-jeong-nmrg-ibn-network-management-automation/06/

[9] K. Yao, D. Chen, J. P. Jeong, Q. Wu, C. Yang, L. M. Contreras, and G. Fioccola, "Use Cases and Practices for Intent-Based Networking," Internet Engineering Task Force, Internet-Draft draft-irtf-nmrg-ibn-usecases-00, Mar. 2025, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-irtf-nmrg-ibn-usecases/00/

[10] S. K. D. Joseph Redmon and, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015. [Online]. Available: http://arxiv.org/abs/1506.02640

[11] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, "A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas," *Machine Learning and Knowledge Extraction*, vol. 5, no. 4, p. 1680–1716, Nov. 2023. [Online]. Available: http://dx.doi.org/10.3390/make5040083

[12] J. Gao, H. Li, Z. Li, C. Xie, X. Ji, and Y. Zhang, "An algorithm for road target detection of autonomous vehicles based on improved yolov8," *Scientific Reports*, vol. 14, no. 1, p. 17282, 2024. [Online]. Available: https://www.nature.com/articles/s41598-025-06831-y

[13] B. Khalili and A. W. Smyth, "Sod-yolov8—enhancing yolov8 for small object detection in aerial imagery and traffic scenes," *Sensors*, vol. 24, no. 19, p. 6209, 2024. [Online]. Available: https://doi.org/10.3390/s24196209

[14] Open Robotics, "Ros 2 documentation: Humble," https://docs.ros.org/en/humble/index.html, 2025, accessed: 2025-08-18.

[15] S. Pellegrini, A. Ess, K. Schindler, and L. V. Gool, "You'll never walk alone: Modeling social behavior for multi-target tracking," in *2009 IEEE 12th International Conference on Computer Vision (ICCV)*, 2009, pp. 261–268.

[16] Y. Yao, M. Xu, C. Choi, D. J. Crandall, E. M. Atkins, and B. Dariush, "Egocentric vision-based future vehicle localization for intelligent driving assistance systems," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 9711–9717.

[17] A. Lerner, Y. Chrysanthou, and D. Lischinski, "Crowds by example," *Computer Graphics Forum*, vol. 26, no. 3, pp. 655–664, 2007.

[18] AI Hub, "Robot-view driving video (advanced) – social navigation robot driving," https://aihub.or.kr/aihubdata/data/view.do?dataSetSn=71755, 2024, dataset (final open v1.1 on 2024-10-30); modalities: RGB JPG / Depth PNG / LiDAR PCD / IMU CSV; labels in JSON (bounding boxes, cuboids, action). Accessed: 2025-08-18.

[19] J. J. D. Rivera, M. M. S. Sarwar, S. Alam, T. A. Khan, M. Afaq, and W.-C. Song, "An Intent-Based Networking mechanism: A study case for efficient path selection using Graph Neural Networks," in *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*, 2023, pp. 1–6.