IETF

IETF-123 Hackathon

Interface to In-Network Computing Functions (I2ICF) Project

July 20, 2025, Madrid

Champion: Jaehoon (Paul) Jeong

Members: Yoseop Ahn, Mose Gu, Jiwon Seo, and Jisuk Chae

Department of Computer Science and Engineering at SKKU

Email: {pauljeong, ahnjs124, rna0415, sjw6136, sue030124}@skku.edu

IETF-123 Interface to In-Network Computing Functions (I2ICF)

Champion: Jaehoon Paul Jeong (SKKU)

IETF-123 Interface to In-Network Computing Functions (I2ICF) Hackathon

Professors:

- Jaehoon Paul Jeong (SKKU)
- Younghan Kim (SSU)
- Yong-Geun Hong (DJU)
- Joo-Sang Youn (DEU)
- Yiwen Chris Shen (AJU)

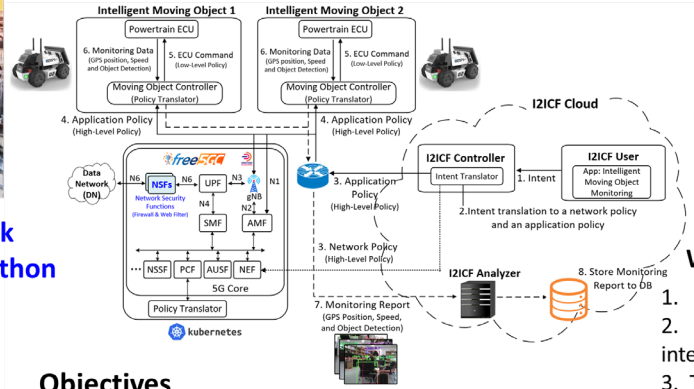
Researchers:

- Jung-Soo Park (ETRI)
- Soohong Daniel Park (Samsung Research)

Students:

- Mose Gu (SKKU)
- Yoseop Ahn (SKKU)
- Jiwon Suh (SKKU)
- Jisuk Chae(SKKU)

I2ICF Architecture



Objectives

- To demonstrate Interface to In-Network Computing Functions (I2ICF).
- To develop an integrated framework for intent-driven orchestration of networking, security, camera-based object recognition, and application services in mobile systems (e.g., Robot Car) operating in distributed computing environments.

Future Work

- The intent translation module will be designed and implemented using advanced techniques such as Knowledge Graph Embedding and Reasoning Model.
- A real-time analytics system powered by Machine Learning and Deep Learning will be developed to assess the operational status of Service Functions (SFs) in mobile platforms, with the aim of enhancing safety and security.

I2ICF Developing Environment

- OS: Ubuntu 20.04
- Kubernetes: Microk8s v1.32.2
- Object Detection: YOLO v8
- ROS Version: Humble
- GitHub Repository: https://github.com/jaehoonpauljeong/I2ICF/tree/main/IETF-123

Workflow of the I2ICF Testbed

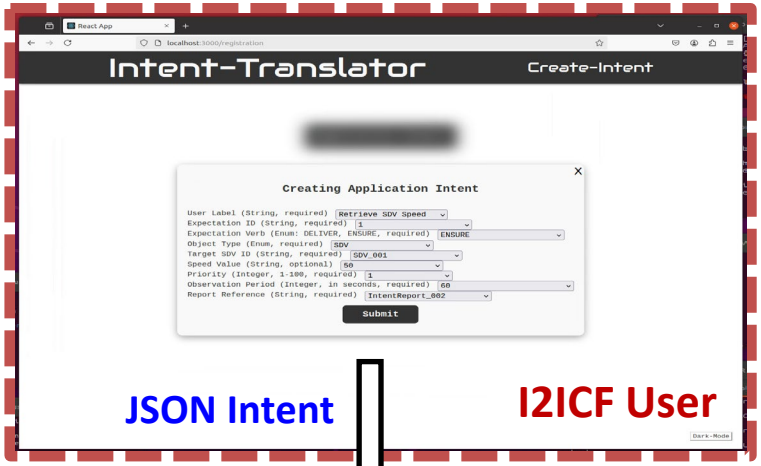
1. I2ICF User sends an intent to the I2ICF Controller.
2. The I2ICF Controller's Intent Translator converts the intent into Network and Application Policies.
3. The translated Network Policy is forwarded to the wireless network components (e.g., 5G NEF).
4. The High-Level Application Policy is sent to each Moving Object Controller.
5. Each Moving Object Controller translates the received Application Policy and applies it to the Powertrain ECU, adjusting the operational parameters.
6. The Moving Objects monitor operational data (e.g., GPS position and speed) and detect objects or obstacles in front of the Robot Car with YOLOv8.
7. The Moving Object Controllers set these data into a Monitoring Report (e.g., JSON File) and forward it to the I2ICF Analyzer.
8. The I2ICF Analyzer processes the Monitoring Report to assess the performance of the policies and stores the results in the database (DB) for further analysis.

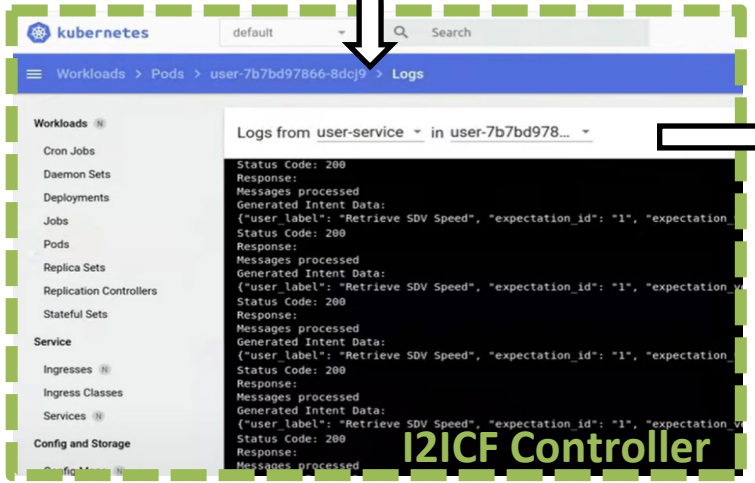
What differences from previous work?

- Previous Goal focused on Intent Translation and Policy Provisioning in Interfaces to In-Network Computing Functions (I2ICF) and its Framework.
  - Creation of a YAML Intent based on 3GPP 28.312 and its Deliverance to Moving Objects.

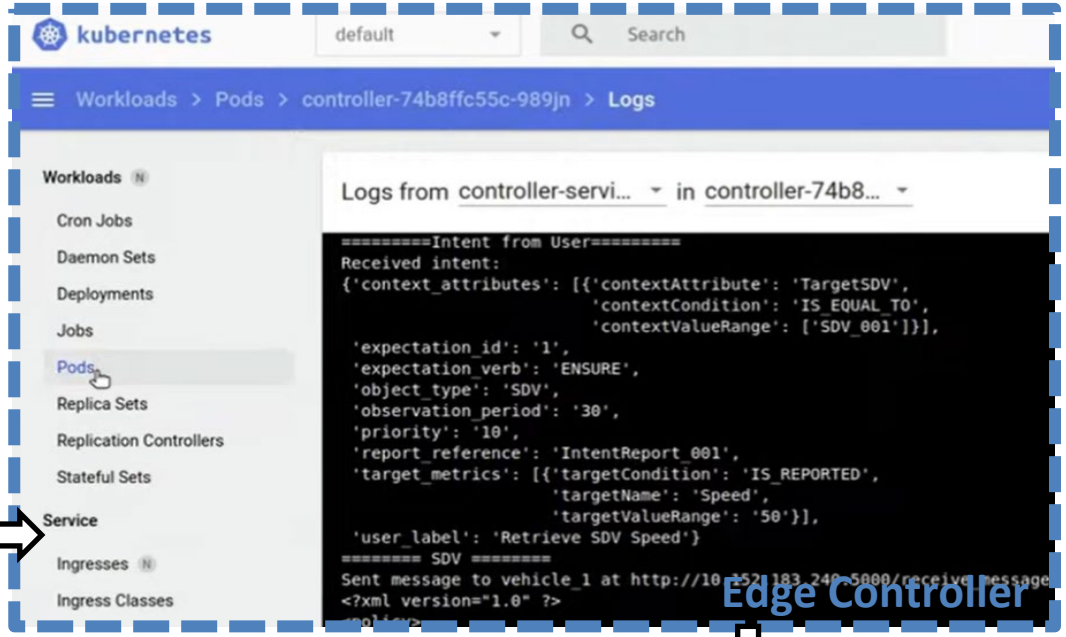
- Present Goal is to implement a User Equipment (UE)'s data collecting ability and monitoring interface to collect data from the UE which is directed by a user intent.
  - We collect data from a UE (e.g., Robot Car) and deliver it to an Analyzer Component for the framework's closed loop purpose.

Previous Demonstration



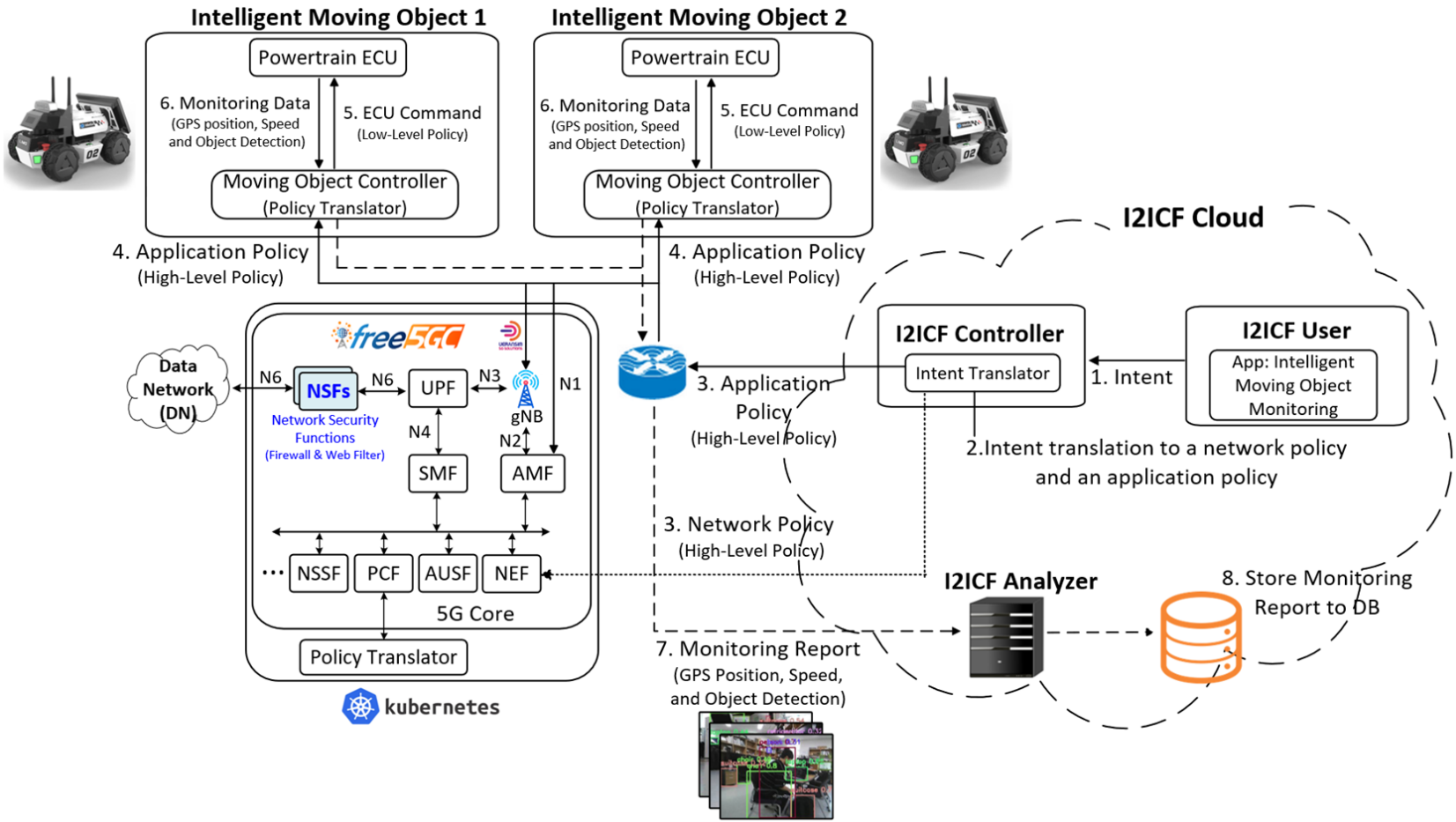


YAML Intent



Moving Object Service Functions

Interface to In-Network Computing Functions (I2ICF) for Moving Objects



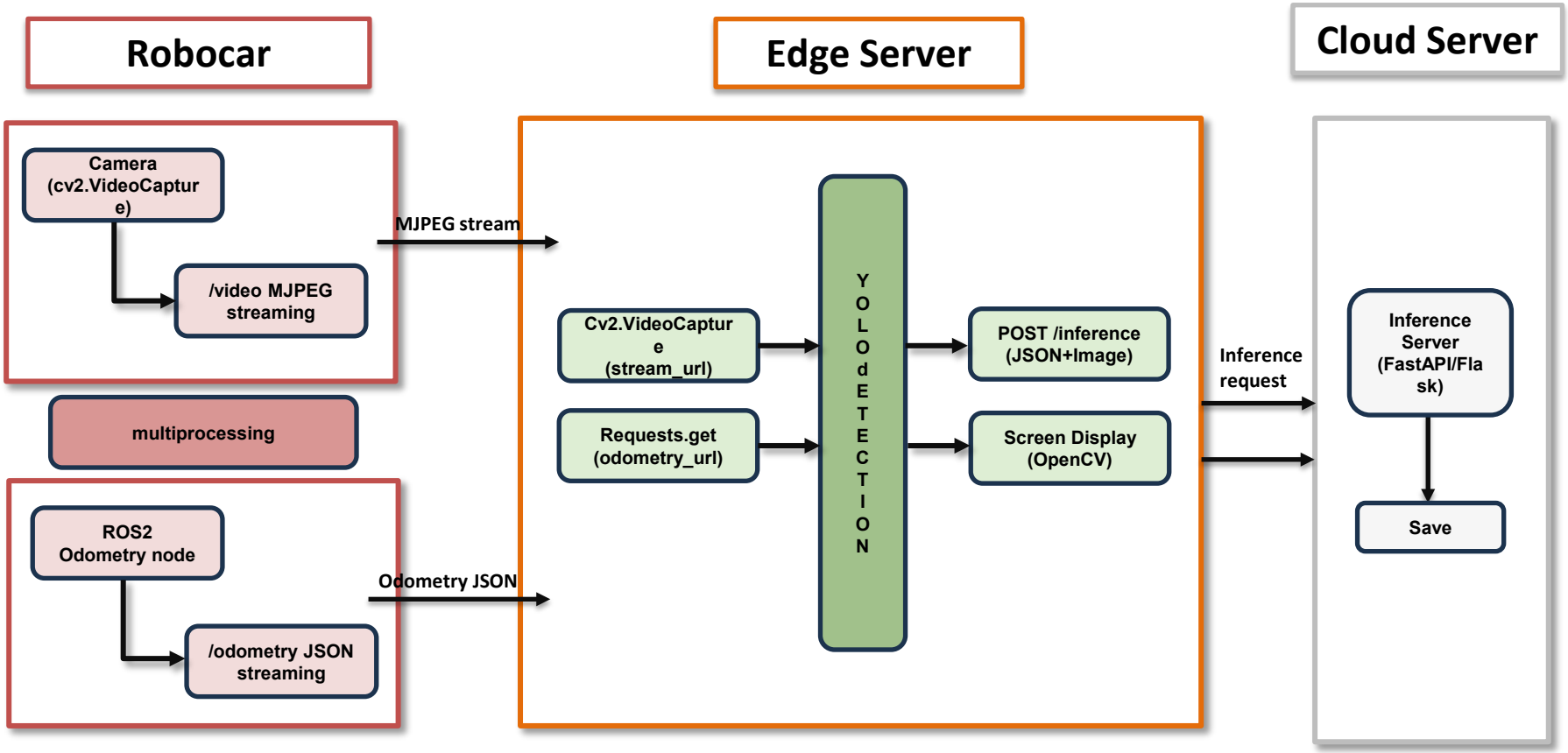
Goal of Hackathon Project

- The goal is to show the feasibility of a framework and interfaces for configuring and monitoring Moving Objects (e.g., robot cars and drones).
  - **I2ICF (Interface to In-Network Computing Functions) Framework**
    - An intelligent framework for moving-objects with an intent translator and closed-loop control.
    - **Three categories of actions** of moving objects: Sensing, Movement, and Actuation.
- Internet Drafts for the I2ICF Project
  - https://datatracker.ietf.org/doc/draft-jeong-opsawg-i2icf-problem-statement/
  - https://datatracker.ietf.org/doc/draft-jeong-opsawg-i2icf-framework/
  - https://datatracker.ietf.org/doc/draft-ywj-opsawg-i2icf-data-center-networking/
  - https://datatracker.ietf.org/doc/draft-ahn-opsawg-i2icf-cits/

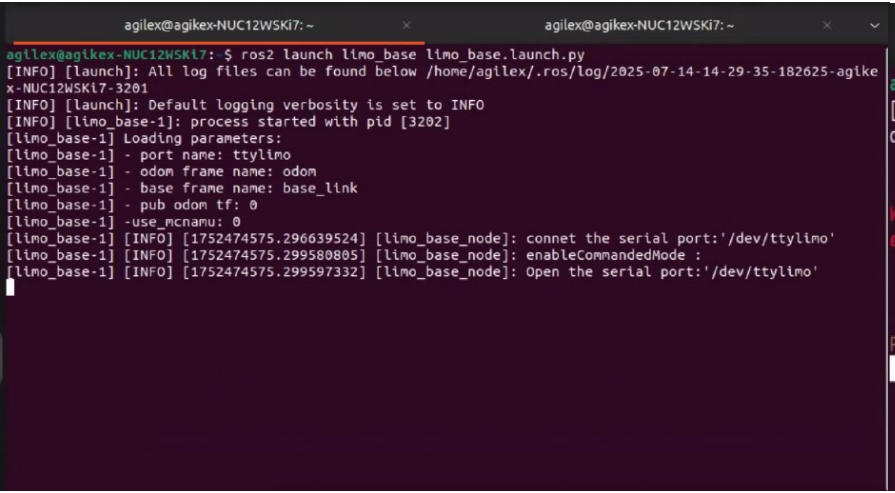
3 Kinds of Actions of Moving Objects

- Sensing
  - Camera Sensor
  - Odometry Sensor
- Movement
  - Four-wheel movement
  - Source, intermediate positions, and destination movement
- Actuation
  - Sensing data delivery
  - Fire alarm

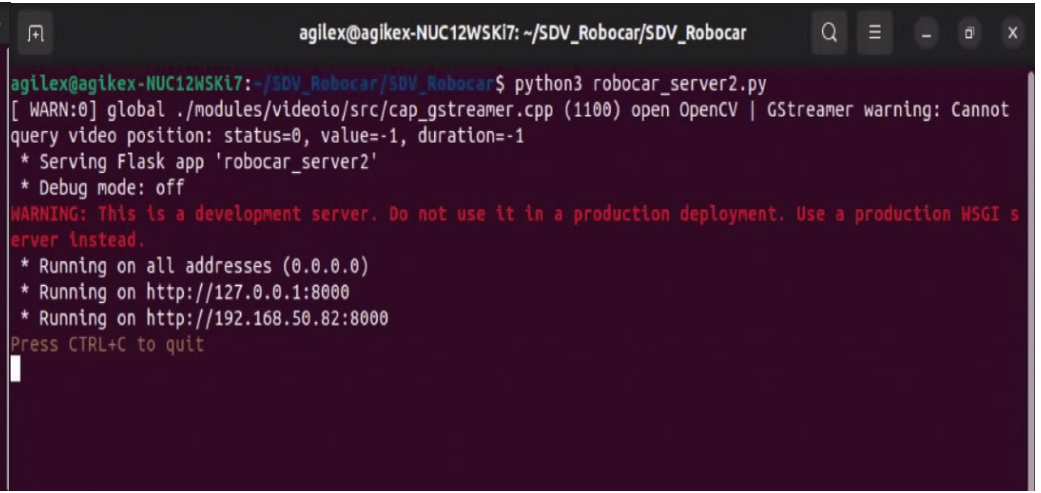
Goal Flow Diagram



Demonstration

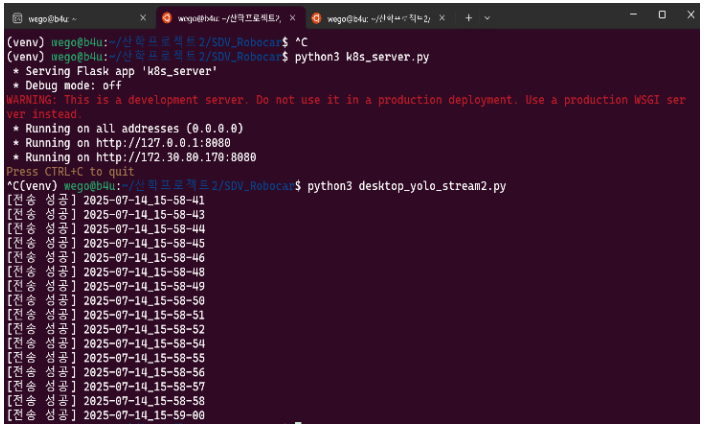


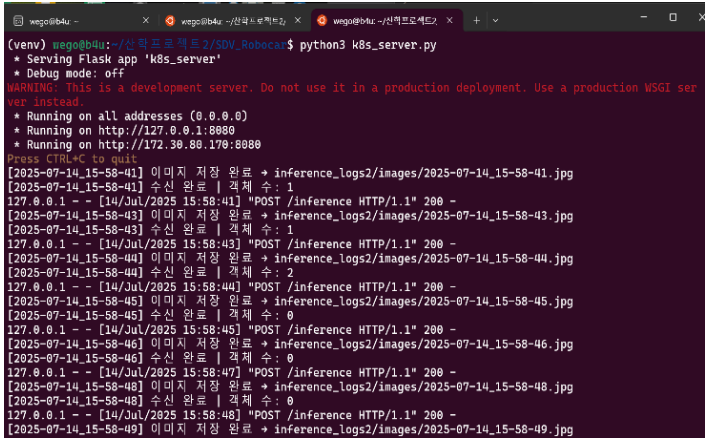
1.LIMO ROS2 Launch



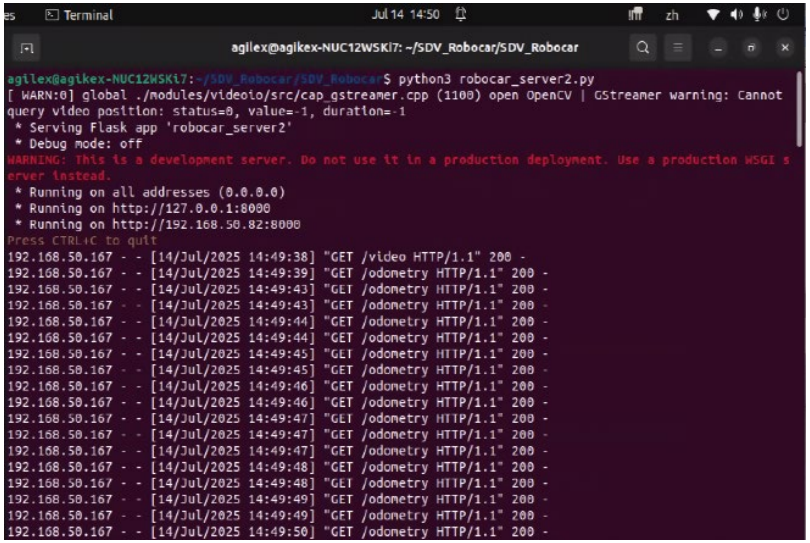
2. Run robocar_server2.py

Demonstration



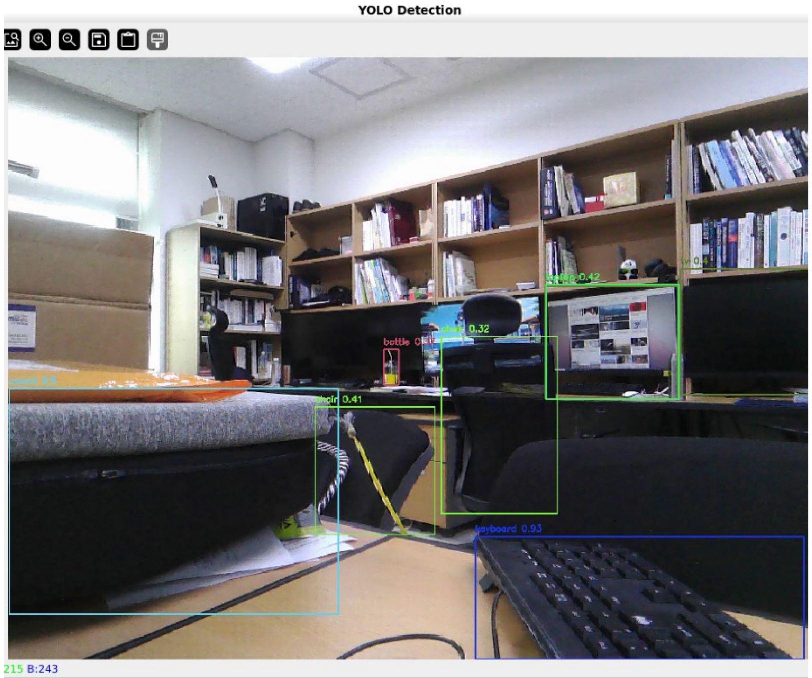


3. Run desktop_yolo_stream2.py & k8s_server.py

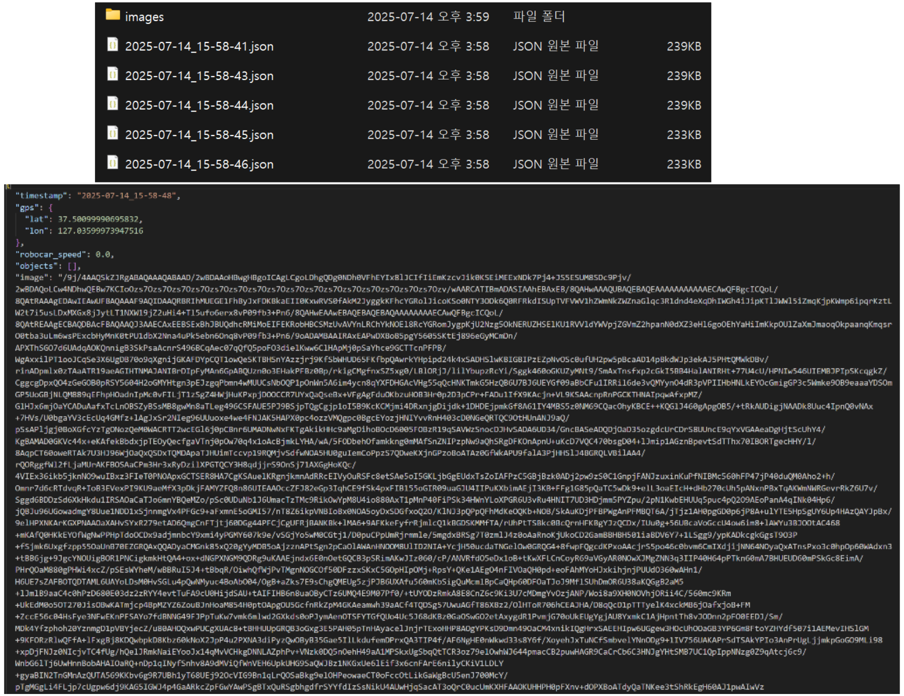


Robocar_server when stream success

Demonstration



YOLO
Detection



Logs

What we learned

- We implemented a Monitoring Interface for collecting and analyzing the Moving Objects' data in the I2ICF Framework in Wireless Networks.

- We demonstrated Intent-Based Networking (IBN) for the configuring and monitoring of Moving Objects through the I2ICF Framework.

Next Steps

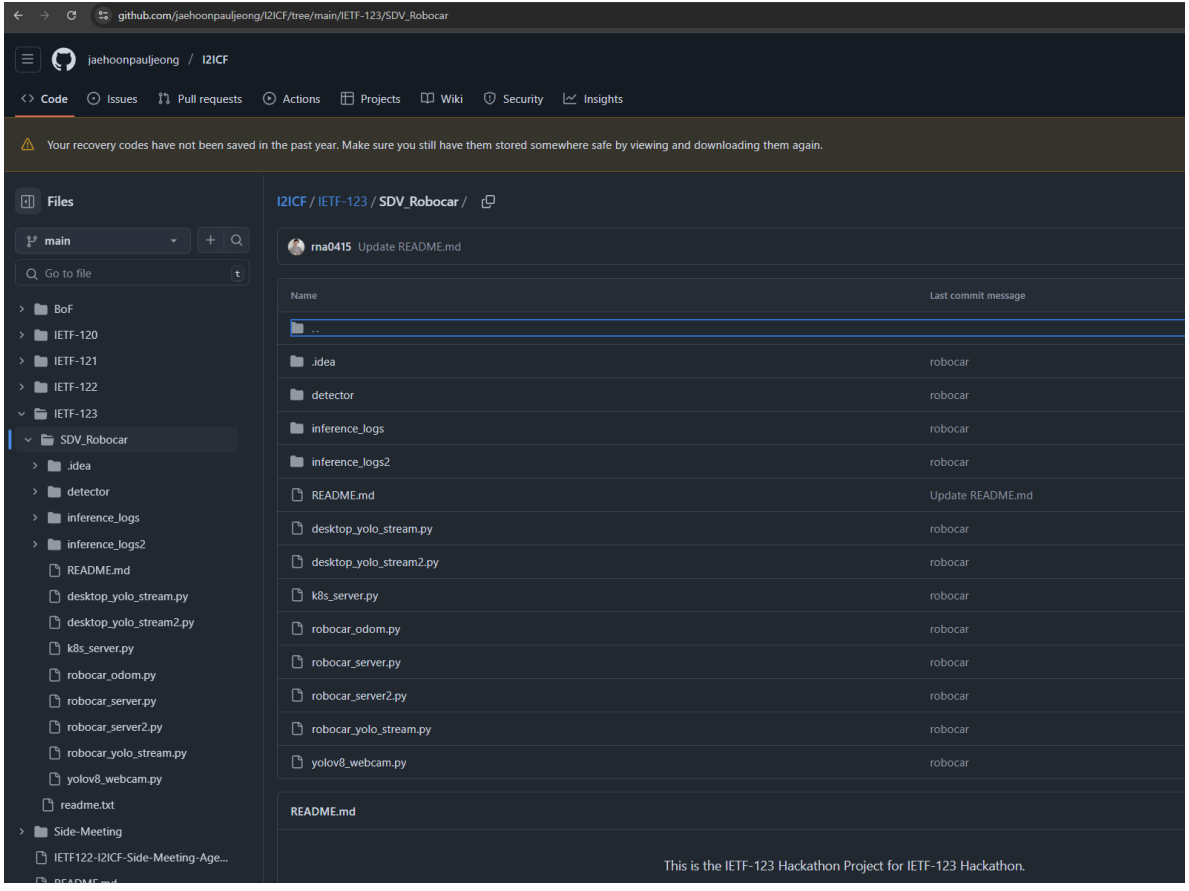
- We will support **three categories of actions** of moving objects: Sensing, Movement, and Actuation.

- In IETF 124, we will adopt **an Intent Translator** in advances from a Rule-based scheme to an AI-based scheme (e.g., Large Language Model: LLM) on Kubernetes Container Orchestration System.

- Also, we will design **YANG Data Models for three categories of actions of Moving Objects.**
  - Refer to https://datatracker.ietf.org/doc/draft-jeong-opsawg-i2icf-framework/

Open-Source Project for I2ICF

[URL] https://github.com/jaehoonpauljeong/I2ICF/tree/main/IETF-123



Demonstration Video Clip for I2ICF

[URL] https://www.youtube.com/watch?v=OKZS2LH-OmA

I2ICF Hackathon Team

- **Professors**:
  - Jaehoon Paul Jeong (SKKU)
  - Younghan Kim (SSU)
  - Yong-Geun Hong (DJU)
  - Joo-Sang Youn (DEU)
  - Yiwen Chris Shen (AJU)
- **Researchers**:
  - Jung-Soo Park (ETRI)
  - Soohong Daniel Park (Samsung Research)
- **Students**:
  - Mose Gu (SKKU)
  - Yoseop Ahn (SKKU)
  - Jiwon Suh (SKKU)
  - Jisuk Chae(SKKU)

Hackathon Team Photo

Appendix

Set-up and Detailed-Procedure

LIMO ROS2 Specification

Physical appearance of the LIMO ROS2 platform

Specifications:

Dimensions	Weight	Payload	Ground Clearance
322 x 220 x 251mm	4.8kg	4kg	24mm
Steering Structure	Screen	IPC	Depth Camera
40N·m	7 inches	Intel NUC	Orbbec Dabai
LiDAR	Battery	Operating Time	Standby Time
EAI T-mini Pro	10Ah 12V	2.5hours	4hours
System	Speed	ROS Version	
Ubuntu 22.04	1m/s	ROS2 Humble	
Control Distance	Control Method		
10m	Mobile App / Command		

Key hardware specifications of LIMO ROS2

System Architecture & Technology Stack

Input data:

- Real-time MJPEG video stream from /video
- Odometry data via /odometry

Output Data (sent every 1 second):

- YOLOv8 object detection results
- Robot speed and GPS (converted from odometry)
- Base64-encoded JPEG image

Storage Example (SERVER):

- inference_logs2/2025-07-14_16-22-01.json
- Inference_logs2/images/2025-07-14_16-22-01.jpg

All sensor data is processed every second and saved as structured JSON with detection results and image.

Programming Language:

- Python 3.x

Core Libraries & Frameworks:

- ROS2 Humble (odometry data)
- Flask (HTTP API server)
- OpenCV (video processing & display)
- Ultralytics YOLOv8 (object detection)
- Multiprocessing (ROS2 + Flask parallel execution)
- Base64 JSON (data encoding & structuring)
- Requests (HTTP transmission)

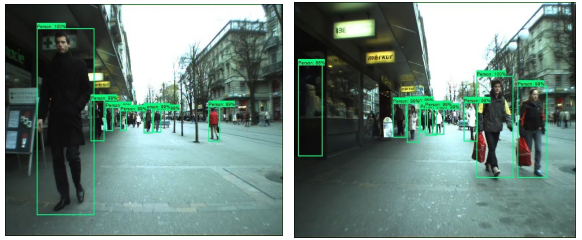
A lightweight software stack built with Python and ROS2 for real-time sensing and inference.

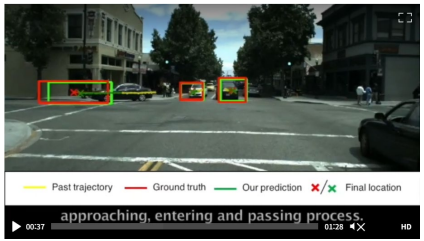
Datasets

- ETH -> Bird Eye View
- HEV-I(Honda Egocentric View-Intersection Dataset) -> outdoor dataset
- UCY -> outdoor dataset
- **AI Hub robot perspective driving video (advanced) - Social navigation robot driving -> Suitable for indoor navigation**
- **Test Dataset 60k, Validation Dataset 6k**

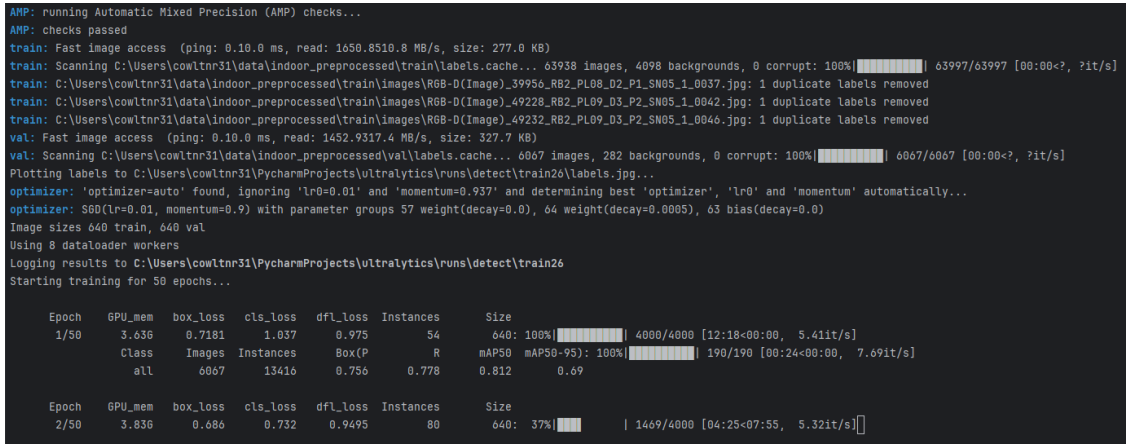
(a) Leader Follower (b) Collision Avoidance (c) Group (d) Others

Fig. 9: Sample scenes from our benchmark. In each of the samples, we illustrate a different social interaction between the primary pedestrian (yellow) and the corresponding interacting neighbours (red) in real world datasets.





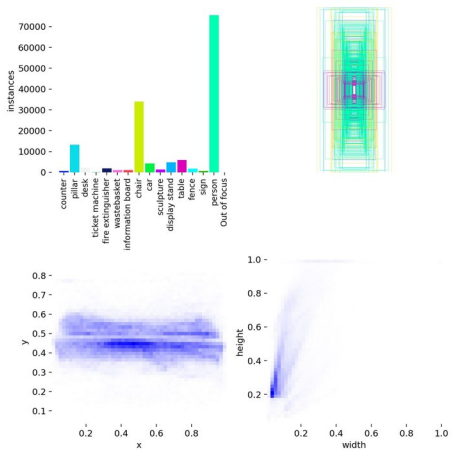
Hyperparameter



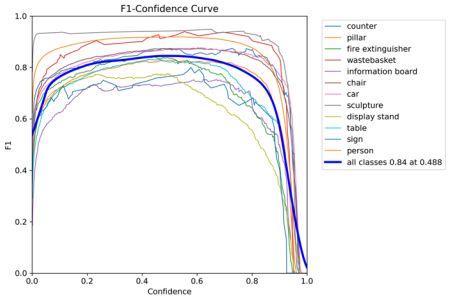
항목	값
model	YOLOv8s (Pretrained)
Rask	Object Detection
Input Size	640 x 640
Epochs	50
Batch Size	16
Optimizer	SGD (momentum = 0.937)
Learning Rate	0.01
Weight Decay	0.0005
Train Images	63,997
Validation Images	6,067

Result

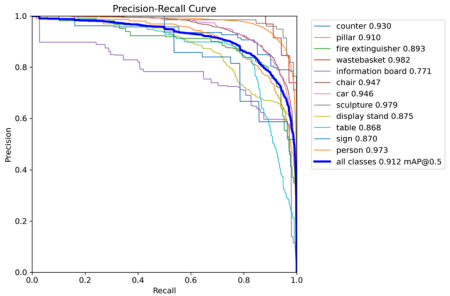
- YOLOv8s fine-tuning performance



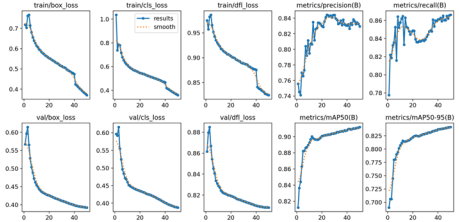
Class distribution in training data



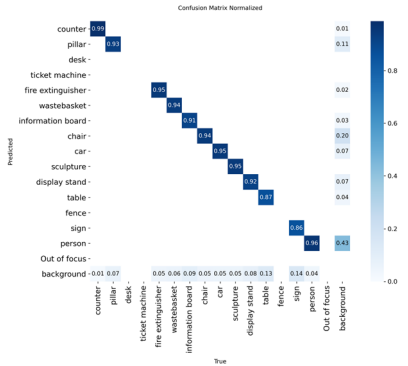
F1 score variation by confidence threshold



PR Curve



loss & accuracy



Normalized confusion matrix

showing per-class prediction accuracy

Result

- YOLOv8s fine-tuning performance

```
Model summary (fused): 72 layers, 11,131,776 parameters, 0 gradients, 28.5 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95):
                   all       6067      13416       0.83      0.867      0.912      0.841
               counter         81         81      0.784      0.984       0.93      0.845
                pillar        604        815      0.798      0.855       0.91      0.855
     fire extinguisher        155        167      0.838      0.838      0.893       0.78
           wastebasket         34         34      0.923      0.912      0.982      0.915
     information board         55        145      0.657      0.828      0.771      0.623
                 chair       1171       2029      0.846      0.901      0.947      0.904
                   car        427        705      0.844      0.897      0.946      0.911
             sculpture         78         78      0.935      0.949      0.979      0.933
         display stand        314        485       0.83      0.722      0.875      0.794
                 table        401        426      0.839       0.81      0.868      0.801
                  sign         28         28      0.743      0.786       0.87      0.841
                person       4202       8423       0.92      0.917      0.973      0.895
Speed: 0.1ms preprocess, 1.8ms inference, 0.0ms loss, 0.6ms postprocess per image
```

-> Epoch 50 => mAP50: 0.912/ mAP50-95: 0.841

Result

- Prediction vs Ground Truth

ground truth on validation images

predicted bounding boxes on validation images

Result

- Real-time result visualization via camera streaming