

# IETF-124 Hackathon



## Interface to In-Network Computing Functions (I2ICF) Project

November 2, 2025, Montreal

**Champion: Jaehoon (Paul) Jeong**

**Members:** Yoseop Ahn, Mose Gu, Jiwon Suh, Jisuk Chae

Department of Computer Science and Engineering at SKKU

Korea Electronics Technology Institute

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

# IETF-124 Interface to In-Network Computing Functions (I2ICF)



## IETF-124 Interface to In-Network Computing Functions (I2ICF) Hackathon

### Professors:

- Jaehoon Paul Jeong (SKKU)
- Yong-Geun Hong (DJKU)
- Joo-Sang Youn (DEU)

### Researchers:

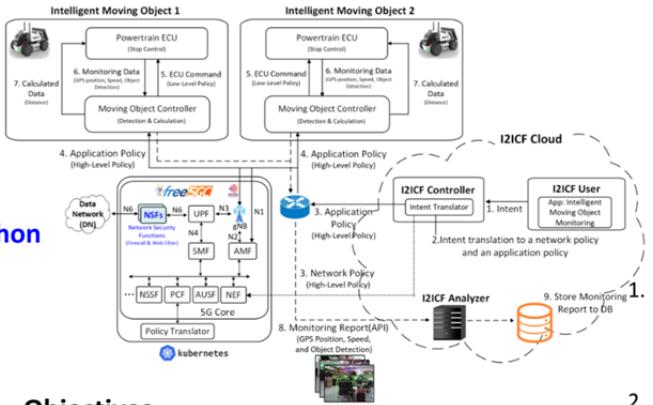
- Jung-Soo Park (ETRI)

### Students:

- Mose Gu (SKKU)
- Jiwon Suh (SKKU)
- Jisuk Chae (SKKU)
- Yeonjoo Lee (SKKU)
- Kihyun Shim (SKKU)
- Isaac Choi (SKKU)
- Wonjae Lee (SKKU)
- Deokhyeon Ryu (DJKU)
- Eunjin Hwang (DJKU)
- Sumin Park (DJKU)
- Eunsan Seong (DEU)
- Taehyun Kwan (DEU)

Champion: Jaehoon Paul Jeong (SKKU)

### I2ICF Framework



### Objectives

- To demonstrate an intent-driven, closed-loop safety pipeline in which a moving object receives an intent, this I2ICF framework (i) performs on-board camera-based object detection and camera–LiDAR calibration/fusion for distance estimation, (ii) streams detections to a cloud server, and (iii) executes real-time stop/avoidance for obstacles and pedestrians.

### Future Work

- To optimize End-to-End performance (e.g., inference accuracy, and inference latency) and extend to two robot cars that exchange on-board inference data to cooperatively plan collision-free, optimal paths when approaching each other.

### I2ICF Developing Environment

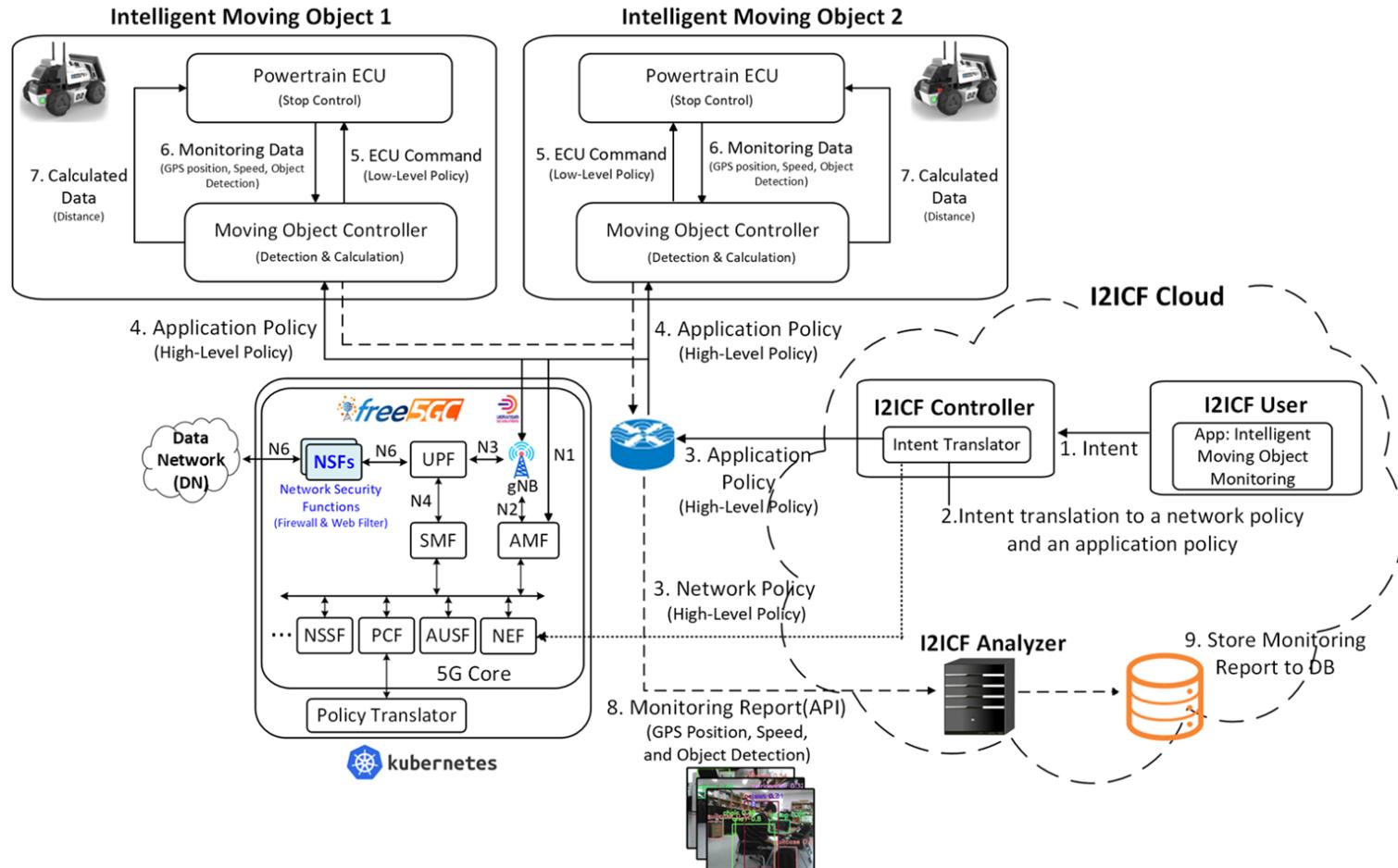
- OS: Ubuntu 20.04
- Kubernetes: MicrOK8s v1.32.2
- Object Detection: YOLO v8
- ROS2 Version: Humble
- GitHub Repository:  
[https://github.com/jaehoonpauljeong/I2ICF/tree/main/IETF-124/SDV\\_Robocar](https://github.com/jaehoonpauljeong/I2ICF/tree/main/IETF-124/SDV_Robocar)

### Workflow of the I2ICF Testbed

1. Intent Submission: A User provides a safety/perception intent (e.g., avoid obstacles/pedestrians; stop within a configured distance).
2. Intent Application: A Moving Object sets detection/stop parameters from the intent.
3. Calibration: Camera–LiDAR calibration; synchronization for distance estimation.
4. Perception & Fusion: Camera detection and LiDAR range fusion to classify objects and estimate distance.
5. Cloud Streaming: Detection results and motion metadata are sent to a cloud server (I2ICF Analyzer).
6. Safety Action: If a proximity threshold is met, the Moving Object stops to avoid a collision and detours for a safe driving.
7. Monitoring: Logs (e.g., JSON) are stored in the cloud database (DB) for further analysis.
8. Monitoring Report(API): GPS Position, Speed, and Object Detection
9. Store Monitoring Report to DB



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

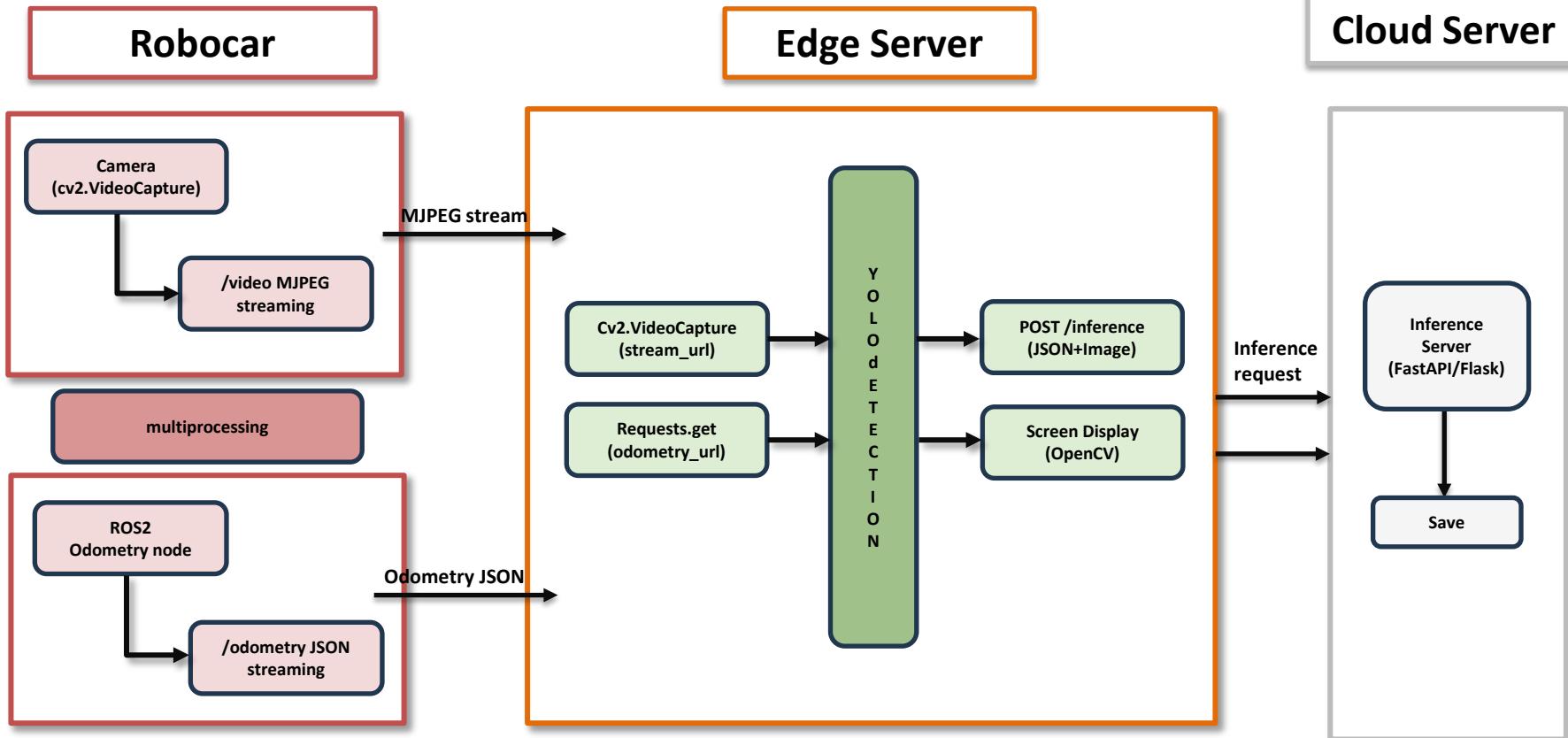




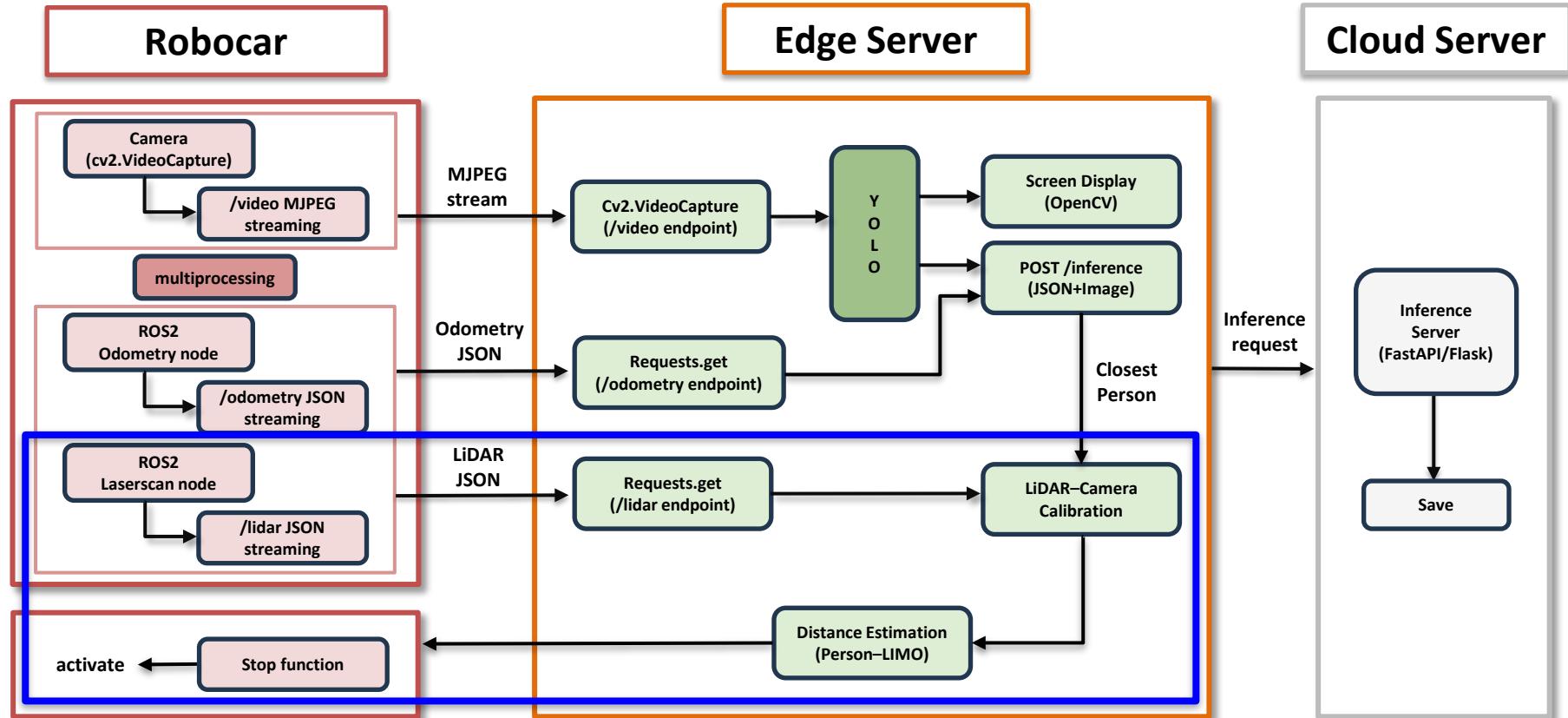
# Goal of Hackathon Project

- The goal is to show the feasibility of a Configured moving objects to Collision avoidance by self-determining obstacles and pedestrians
  - **I2ICF (Interface to In Computing Functions) Moving Object Controller**
    - The robot car fuses the sensing data of both camera and LiDAR in real time to measure distance and stops automatically when it is too close.
- 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-gu-nmrg-intent-translator/>

# IETF-123 I2ICF Framework (Before)



# IETF-124 I2ICF Framework (Now)



New Feature: Calibrating LiDAR and Camera for Distance Measurement

# Demonstration (1/3)

```
agilex@agikex-NUC12WSK17:~ $ ros2 launch limo base limo_base.launch.py
[INFO] [launch]: All log files can be found below /home/agilex/.ros/log/2025-07-14-14-29-35-182625-agikex-NUC12WSK17-3201
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] [limo_base-1]: process started with pid [3202]
[INFO] [limo_base-1]: Loading parameters:
[INFO] [limo_base-1]: - port name: ttylimo
[INFO] [limo_base-1]: - odom frame name: odom
[INFO] [limo_base-1]: - base frame name: base_link
[INFO] [limo_base-1]: - pub odom tf: 0
[INFO] [limo_base-1]: -use_mcnamu: 0
[INFO] [limo_base-1]: connect the serial port:'/dev/ttylimo'
[INFO] [limo_base-1]: enableCommandedMode :
[INFO] [limo_base-1]: Open the serial port:'/dev/ttylimo'
```

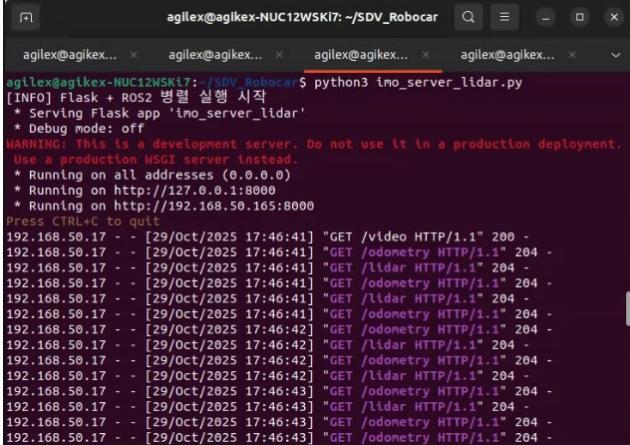
## 1. [LIMO] Run 'limo\_base.launch.py'

```
agillex@agikex-NUC12WSK17:~ agillex@agikex-NUC12WSK17:~/SDV_Ro... agillex@agikex-NUC12WSK17:~$ ros2 launch ydlidar_ros2_driver ydlidar.launch.py [INFO] [launch]: All log files can be found below /home/agillex/.ros/log/2025-10-29-17-10-32-928581-agikex-NUC12WSK17-12205 [INFO] [launch]: Default logging verbosity is set to INFO [INFO] [ydlidar_ros2_driver_node-1]: process started with pid [12216] [INFO] [static_transform_publisher-2]: process started with pid [12218] [static_transform_publisher-2] [WARN] [1761729832.987635221] []: Old-style arguments are deprecated; see --help for new-style arguments [static_transform_publisher-2] [INFO] [1761729832.996229496] [static_tf_pub_laser]: Spinning until stopped - publishing translation [static_transform_publisher-2] translation: ('0.000000', '0.000000', '0.028000') [static_transform_publisher-2] rotation: ('0.000000', '0.000000', '0.000000', '1.000000') [static_transform_publisher-2] from 'base_link' to 'laser_link' [ydlidar_ros2_driver_node-1] [INFO] [1761729832.997712109] [ydlidar_ros2_driver_node]: [YDLIDAR] INFO! Current ROS Driver Version: 1.0.1 [ydlidar_ros2_driver_node-1] [ydlidar_ros2_driver_node-1] YDLidar SDK initializing [ydlidar_ros2_driver_node-1] YDLidar SDK has been initialized [ydlidar_ros2_driver_node-1] [YDLIDAR]:SDK Version: 1.1.3 [ydlidar_ros2_driver_node-1] LIDAR successfully connected [ydlidar_ros2_driver_node-1] [YDLIDAR]:Lidar running correctly ! The health status: good [ydlidar_ros2_driver_node-1] [YDLIDAR INFO] Current SDK does not support current
```

## 2. [LIMO] Run ‘ydlidar.launch.py’

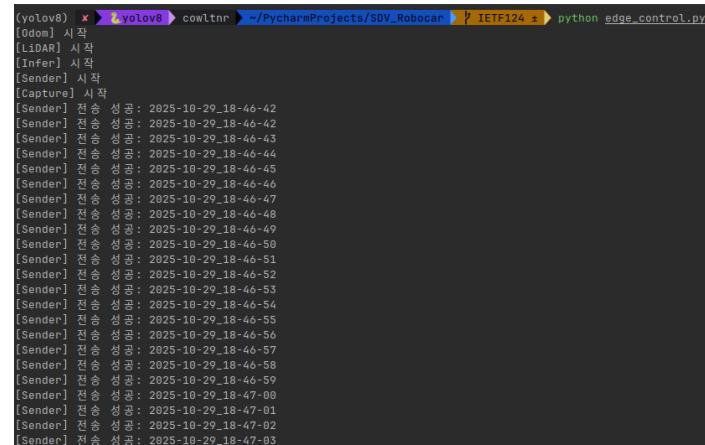
### 3. [LIMO] Run ‘imo server lidar.py’

# Demonstration (2/3)



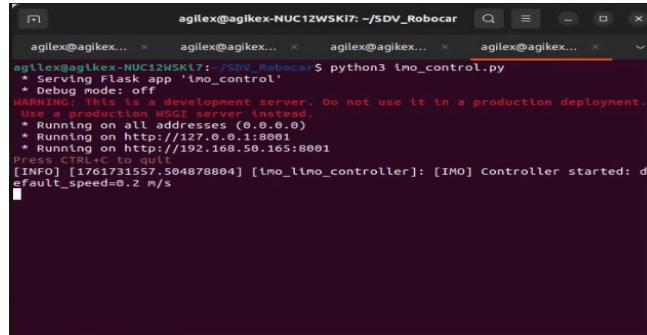
```
agilex@agikex-NUC12WSK17:~/SDV_Robocar$ python3 imo_server_lidar.py
[INFO] Flask + ROS2 병렬 실행 시작
* Serving Flask app 'imo_server_lidar'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:8000
* Running on http://192.168.50.165:8000
Press CTRL+C to quit
192.168.50.17 - - [29/Oct/2025 17:46:41] "GET /video HTTP/1.1" 200 -
192.168.50.17 - - [29/Oct/2025 17:46:41] "GET /odometry HTTP/1.1" 204 -
192.168.50.17 - - [29/Oct/2025 17:46:41] "GET /lidar HTTP/1.1" 204 -
192.168.50.17 - - [29/Oct/2025 17:46:41] "GET /odometry HTTP/1.1" 204 -
192.168.50.17 - - [29/Oct/2025 17:46:41] "GET /lidar HTTP/1.1" 204 -
192.168.50.17 - - [29/Oct/2025 17:46:41] "GET /odometry HTTP/1.1" 204 -
192.168.50.17 - - [29/Oct/2025 17:46:42] "GET /odometry HTTP/1.1" 204 -
192.168.50.17 - - [29/Oct/2025 17:46:42] "GET /lidar HTTP/1.1" 204 -
192.168.50.17 - - [29/Oct/2025 17:46:42] "GET /odometry HTTP/1.1" 204 -
192.168.50.17 - - [29/Oct/2025 17:46:42] "GET /lidar HTTP/1.1" 204 -
192.168.50.17 - - [29/Oct/2025 17:46:42] "GET /odometry HTTP/1.1" 204 -
192.168.50.17 - - [29/Oct/2025 17:46:43] "GET /lidar HTTP/1.1" 204 -
192.168.50.17 - - [29/Oct/2025 17:46:43] "GET /odometry HTTP/1.1" 204 -
192.168.50.17 - - [29/Oct/2025 17:46:43] "GET /odometry HTTP/1.1" 204 -
```

4. Run 'k8s\_server.py'



```
(yolov8) x ➜ yolov8 cowltnr ~/PycharmProjects/SDV_Robocar $ iETF124 ± python edge_control.py
[Dodom] 시작
[LiDAR] 시작
[Infer] 시작
[Sender] 시작
[Capture] 시작
[Sender] 전송 성공: 2025-10-29_18-46-42
[Sender] 전송 성공: 2025-10-29_18-46-42
[Sender] 전송 성공: 2025-10-29_18-46-43
[Sender] 전송 성공: 2025-10-29_18-46-44
[Sender] 전송 성공: 2025-10-29_18-46-45
[Sender] 전송 성공: 2025-10-29_18-46-46
[Sender] 전송 성공: 2025-10-29_18-46-47
[Sender] 전송 성공: 2025-10-29_18-46-48
[Sender] 전송 성공: 2025-10-29_18-46-49
[Sender] 전송 성공: 2025-10-29_18-46-50
[Sender] 전송 성공: 2025-10-29_18-46-51
[Sender] 전송 성공: 2025-10-29_18-46-52
[Sender] 전송 성공: 2025-10-29_18-46-53
[Sender] 전송 성공: 2025-10-29_18-46-54
[Sender] 전송 성공: 2025-10-29_18-46-55
[Sender] 전송 성공: 2025-10-29_18-46-56
[Sender] 전송 성공: 2025-10-29_18-46-57
[Sender] 전송 성공: 2025-10-29_18-46-58
[Sender] 전송 성공: 2025-10-29_18-46-59
[Sender] 전송 성공: 2025-10-29_18-47-00
[Sender] 전송 성공: 2025-10-29_18-47-01
[Sender] 전송 성공: 2025-10-29_18-47-02
[Sender] 전송 성공: 2025-10-29_18-47-03
```

5. Run 'edge\_control.py'



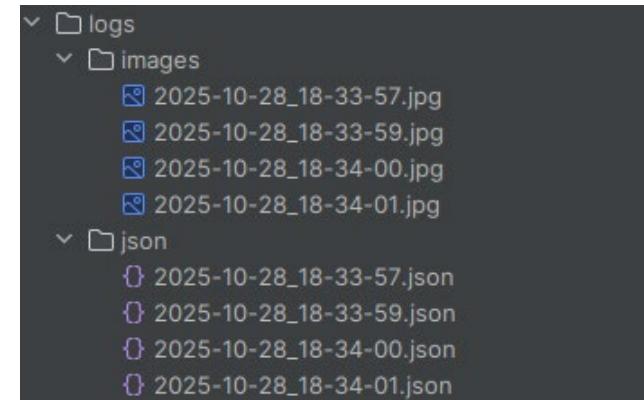
```
agilex@agikex-NUC12WSK17:~/SDV_Robocar$ python3 imo_control.py
[INFO] Flask + ROS2 병렬 실행 시작
* Serving Flask app 'imo_control'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:8001
* Running on http://192.168.50.165:8001
Press CTRL+C to quit
[INFO] [1761731557.504878804] [imo_limo_controller]: [IMO] Controller started: default_speed=0.2 m/s
```

6. Run 'imo\_control.py'

# Demonstration (3/3)



Result 1: YOLO Detection & Distance



Result 2: Logs



# What we learned

- We implemented a real-time perception-to-stop pipeline.
  - Distance Measurement with Camera and LiDAR and Auto-Stopping for an Obstacle.
- Intent-Based Driving
  - Robot stops when an obstacle is near.



# Next Steps

- **Design & Implementation of Intent Translator and Policy Translator**
- **Optimization of End-to-End Pipeline**
  - Reduction of frame drops and inference latency
  - Stabilization of distance estimation
- **IETF-125 Hackathon**
  - Extension of I2ICF Testbed
    - Exchange inference data and cooperative maneuver control.
  - YANG Data Models for I2ICF Moving Object Management
    - <https://datatracker.ietf.org/doc/draft-jeong-opsawg-i2icf-framework/>



# Open-Source Project for I2ICF

[URL] [https://github.com/jaehoonpauljeong/I2ICF/tree/main/IETF-124/SDV\\_Robocar](https://github.com/jaehoonpauljeong/I2ICF/tree/main/IETF-124/SDV_Robocar)

The screenshot shows a GitHub repository interface. The left sidebar displays a tree view of the repository structure:

- main
- BoF
- IETF-120
- IETF-121
- IETF-122
- IETF-123
- IETF-124/SDV\_Robocar
  - .idea
  - detector
  - logs
    - README.md
    - edge\_control.py
    - imo\_control.py
    - imo\_server\_lidar.py
    - k8s\_server.py
- Side-Meeting
  - README.md
  - Readme

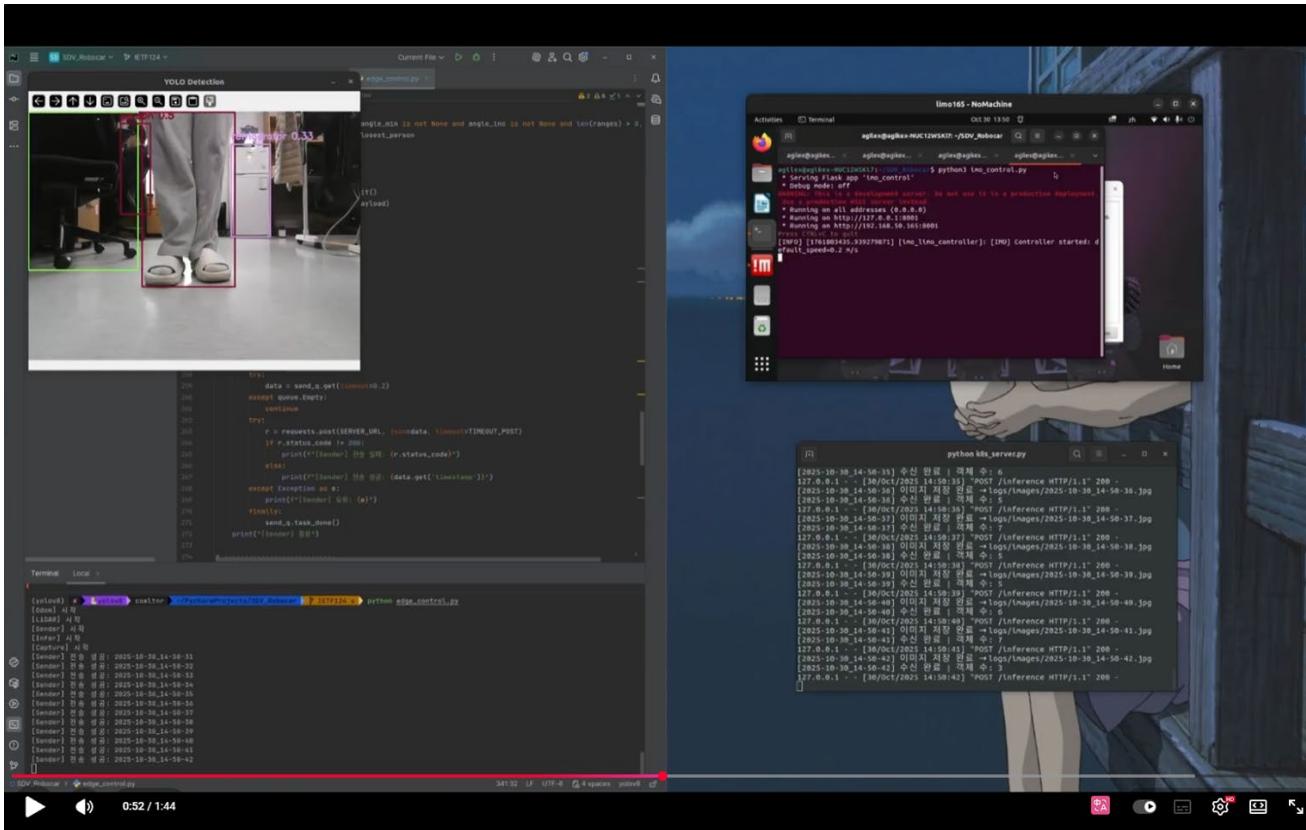
## Main steps

### LIMO

- Run server
- Stream camera

# Demonstration Video Clip for I2ICF

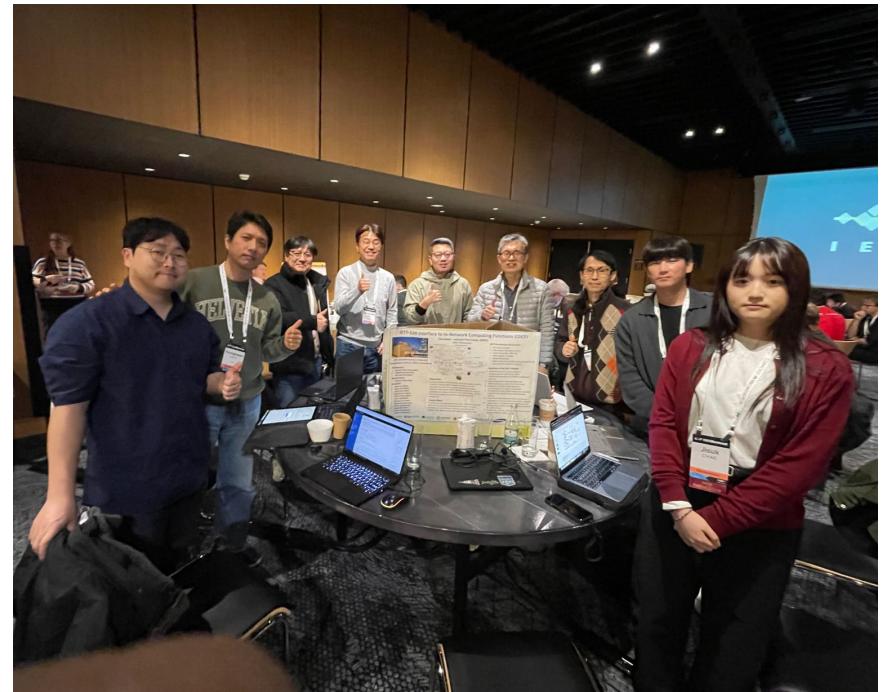
[URL] <https://www.youtube.com/watch?v=lw0GV3yN5Nw>



# I2ICF Hackathon Team

- **Professors:**
  - Jaehoon Paul Jeong (SKKU)
  - Yong-Geun Hong (DJU)
  - Joo-Sang Youn (DEU)
- **Researcher:**
  - Jung-Soo Park (ETRI)
- **Students:**
  - Mose Gu (SKKU)
  - Jiwon Suh (SKKU)
  - Jisuk Chae (SKKU)
  - Yeonjoo Lee (SKKU)
  - Kihyun Shim (SKKU)
  - Isaac Choi (SKKU)
  - Wonjae Lee (SKKU)

## Hackathon Team Photo



# Appendix



# Differences from the previous work

- The Previous Goal
  - ✓ To implement User Equipment (UE)'s data collecting ability and a monitoring interface to collect data from the UE which is directed by a user's intent.
  - ✓ We collect data from a UE (i.e., Robot Car) and deliver it to an Analyzer Component for the framework's closed loop control.
- The Present Goal
  - ✓ To have a moving object (i.e., Robot Car) receive an intent and autonomously drive.
  - ✓ To perform onboard camera-based object detection and camera-LiDAR calibration/fusion for distance estimation.
  - ✓ To let the moving object deliver the detection results to a cloud server to perform real-time stopping to avoid the collision with obstacles.
  - ✓ Camera-LiDAR data is calibrated and fused; collision control is performed via the powertrain ECU; detection results are transferred to a cloud server.