1 Introduction

In this status, we focused on creating the overall software structure. Based on the Code Review, we thought about how to create the structure using Brain and Embedded Code. To ensure smooth collaboration between the parts that will be assigned to each team member, each module and interface were clearly divided. Because it was before the vehicle arrived, Perception and Localization, which are related to actual sensors, focused on structure, algorithm, and model selection. On the other hand, Path Planning and Control conducted tests and actual implementations using simulators, F1tenth Gym and WeBOT2.0.

2 Planned activities

(1) Perception (assigned to Minhyek Choi)
- a. Search Possible Detection Model
- b. Implement Lane Detection with Conventional CV

(2) Localization (assigned to Myungeun Cho)
- a. Search possible Localization Algorithm
- b. Implement Extended Kalman Filter with Reference

(3) Path Planning (assigned to Hyoseok Bang)
- a. Make Binary Masked map
- b. Test Global Planner with Checkpoint
- c. Test Local Path Planner with Visualization tool

(4) Control (assigned to Geunwoo Kim)
- a. Test Local Path Planner with Visualization tool
- b. Divide Left and Right Lane in any Perception case
- c. Implement Pure Pursuit Controller in Simulator

3 Status of planned activities

(1) Perception

a. Search Possible Detection Model (delayed)
Decide to compare between training a YOLOv8 model from scratch and using transfer learning with YOLOv5. YOLOv8 generally outperforms YOLOv5 when it comes to small parameter models. However, YOLOv5 offers stable support for transfer learning methods, and a model trained in this way can potentially perform better than a YOLOv8 model trained from scratch. Therefore, We will compare performance and computational power of each model to determine the most suitable approach.

b. Implement Lane Detection with Conventional CV (completed)
By using HSV Filtering and Hough Line Transfrom in OpenCV Library, a traditional Lane Detection algorithm was developed and applied. It was confirmed that it was successfully recognized through parameter change in real world using WeBOT2.0.

(2) Localization

a. Search possible Localization Algorithm (completed)
To accurately estimate the vehicle's position based on the given sensor data, we plan to utilize either Particle Filter or Extended (or Unscented) Kalman Filter.

b. Implement KF with example code (completed)
Currently working on implementing a code using example code of Particle Filter and Kalman Filter to receive sensor data sent from the ROS simulator and apply filtering to it.

(3) Path Planning

a. Make Binary Masked map (completed)
Different information on the map (solid lines, dotted lines, stop lines, etc.) were mapped to unique numbers and export in .txt format.

b. Test Global Planner with Node (completed)
By reading txt file, a map is created again, and global path planning is performed based on the designated checkpoint. But now it coded by python, should be change in C++.

c. Test Local Path Planner with Visualization tool (completed)
The overall driving will be planned by comparing these two (Lane Following based on Global Path / Using Race Track Information with Localization), but Local Path Planning is required in obstacle and overtaking situations. Therefore, BFS, Dijkstra, and A* were used to implement the planner and visualized. Currently, non-Holonomic constraint is not considered, so it will be applied in future projects.

(4) Control

a. Implement Pure Pursuit Controller in Simulator (completed)
Based on Path made by dividing the left and right lanes of Path Planning, a controller that Purely pursuit the average center point was implemented. It was also confirmed with normal speed WeBOT2.0 that it works well in real situations. However, on the Simulation, if vehicle runs in fast speed, the fact that there is possibility of diverging is checked. In the future, we plan to apply another Controller (Stanley, Optimal) to compare the performance.

4 General status of the project

(1) Vehicle can Set Local Path in Grid Map with Obstacles

-Calculate the path that can respond to obstacle situations in real time through the Local Path Planner that can cope with various obstacle situations. It will consider the constraint of the vehicle later.

(2) Vehicle can Follow Lanes in Real World (with Camera)

-Based on lane detection through Conventional Computer Vision, Pure Pursuit Controller was tested, and lane tracking performance was checked in the test track environment.

5 Upcoming activities

(1) Perception
- a. Select Camera (Stereo) to meet Perception Performance
- b. Detect Traffic Sign and Measure Performance
- c. Detect Lanes and Measure Performance

(2) Localization
- a. Create a out Test Track Map
- b. Develop ICP for Localization
- c. Develop UKF for Localization
- d. Test Algorithm with Simulator

(3) Path Planning
- a. Make Global Planning function
- b. Implement Occupancy Grid Map Class
- c. Implement Path Planner (Hybrid A*) with Holonomic Constraint

(4) Control
- a. Measurement Constraint of real vehicle
- b. Vehicle Calibration
- c. Maneuver Test with Joystick