Automated Driving Using External Perception

Teammates: Ronit Hire, Dhanesh Pamnani, Shreyas Jha, Jash Shah

ILR03

6th April, 2023

Sponsor: Nissan Automotive Inc.
Liam Pederson, Najam Baig, & Viju James

Table of Contents

1. Individual Progress

1.1 MRSD Project: Team OuterSense

Plotting, run car with simulated states, integrated with perception, pdb

Generate Velocity commands: Previously MPC steering output was sent to RC car and the raspberry pi on the RC car commanded the servo to follow steering commands. This time I worked on using acceleration commands to generate velocity commands that can be given to the ESC on the car. Using the dynamics of the vehicle the velocity values were calculated based on the acceleration commands. The next step was to generate a profile that can be given to the RC car for it to follow if there is a communication loss. The profile is defined in such a way that it will follow the MPC command for few time steps and then gradually come to a stop. The code was then written on the controller of RC car to follow the profile over the time period defined by the MPC execution.

Solving Plotting issue: Earlier matplot lib was used for plotting data and visualize MPC outputs. But plotting caused delay in execution so switched to publishing all the data that needs to be plotted on a topic and used plotjuggler to visualize the data. Currently working on RVIZ to have a better visualization of the data. The delay issue was solved since now plotting is done parallelly. Refer figure 1.1.1 for plots

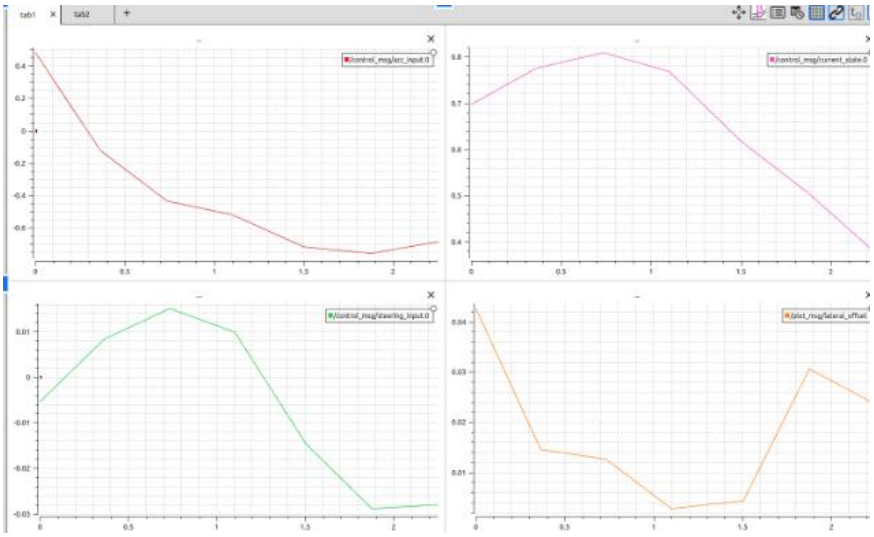

Figure 1.1.1 : MPC output command plots

Operate RC car using simulated data: The goal was to integrate RC car and MPC controller. A desired trajectory was defined and given to MPC and states were simulated that closely mimics the RC car behavior. Using these simulated states commands were generated and given to the RC car. We first had to get the computer with MPC running and the RC car on the same network and configure the computer as master. The RC car successfully followed the given trajectory at a constant velocity and desired steering values.

Integrate perception data with MPC: Now the control system will take the data from the perception unit and update the states which will be used by the MPC. To test, the car was moved manually and it was checked the states estimated by the camera were correct. The states have x and y coordinate, velocity and yaw of the vehicle. The car was moved the estimates were verified on the plotjuggler visualizer. I had to modify the control system code to incorporate perception data from the camera, convert in world frame and get used by MPC as inputs. Based on the current state of the vehicle and the desired trajectory defined the MPC outputs were verified.

Complete integrated test: the entire system was operated. All the subsystems – perception, controls, RC car were integrated and tested. A straight line path was given as desired trajectory ,states were used from the perception unit and commands were given to the RC car. Successfully tested the experiment where the RC car followed the straight path. We ran it multiple times by giving different orientations and starting positions. There were many failure cases as well, currently we are working on identify those issues and making it more robust.

Power Distribution Board: Started soldering on the power distribution board.

2. Challenges

2.1 MRSD project

- The challenge was to identify that plotting was causing delay and other issues in execution.
- The other challenge was to get all execution in sync during the test with RC car on simulated states.
- Debug issues in state estimation from camera. We need to run multiple times and test different scenarios to understand the problem.

3. Team work

Ronit Hire: Ronit worked on making the tracking algorithm more robust. He also worked on finding the velocity of the vehicle. I collaborated with him to integrate the perception subsystem with the control subsystem and test the performance.

Shreyas Jha: Shreyas worked on resolving the ESC issue and integrated the new ESC (VESC) with Raspberry PI and ROS. He ad to study the entire

architecture of VESC to understand how to integrate with our system. He also worked on time synchronization on all ROS machines in our project.

Dhanesh Pamnani: He worked on lane detection algorithm and also developed mechanical parts to complete building the RC car.

Jash Shah: Jash worked on setting up the embedded system for edge compute on the infrastructure unit. He faced a lot of issues in running ROS and ubuntu on these devices. He had to try on multiple devices such as Raspberry Pi, nano and jetson.

4. Future Plan

- The next step is to have multiple integrated testing and modify the code accordingly for more robust operation.
- Complete PDB soldering and test the board
- Set up a better visualization platform