Automated Driving Using External Perception

Individual Lab Report

MRSD Project Course I

Spring 2023

Shreyas Jha

Team E: OuterSense

Teammates: Atharv Pulapaka, Dhanesh Pamnani, Jash Shah, Ronit Hire

ILR05

Apr. 6, 2023

Contents

Individual Progress

In my previous ILRs, we noted that the servo motor exhibited jitter and the BLDC motor displayed a non-smooth response with our ESC on the RC car. While this wasn't a significant issue while teleoperating the vehicle, it hindered precise low-level control of the RC car, which is crucial for ensuring robust subsystem functionality. To address this, I spent the last couple of weeks refining the car subsystem. I integrated the VESC (Vedder Electronic Speed Controller) onto the car. The VESC is an ESC that employs ERPM (Electronic RPM) feedback from the BLDC motor to drive the motor at the desired RPMs, features a built-in IMU, reliable power electronic drive circuits, and a large online community. The physical hardware and connections are shown in Figure 1.

To integrate the VESC, I studied the F1-tenth autonomous racing software repository, which uses the VESC. The repository was extensive and complex, and identified specific components relevant to our requirements. Then I began calibrating the BLDC motor configuration using the VESC tool. With the motor connected to the VESC, I navigated to the "Motor" tab in the VESC tool and selected the appropriate motor type and number of poles. Next, I proceeded to the "Detection" tab and followed the instructions to detect the motor parameters such as resistance, inductance, and back-EMF constant. The VESC tool guided me through the process, and once the motor parameters were detected, I entered them into the motor configuration settings. I also adjusted other parameters such as current limits, acceleration, and braking settings under the "App Configuration" tab. Once I had made the necessary adjustments, I saved the configuration by clicking the "Write Configuration" button. With the calibration complete, I felt confident that the motor was operating efficiently and that I had fine-tuned the motor settings to match our specific requirements.

Finally, I began developing the software in ROS to control the VESC as per the commands received from the central MPC block. Here I had to handle following the correct command in the profile sent by the MPC as per the control horizon defined in the MPC block. I also developed a software-based E-stop to instantaneously stop the RC car and supersede all other control commands received.

As part of my work on the distributed subsystems, I focused on configuring Chrony (an open-source implementation of the Network Time Protocol (NTP) used for synchronizing the system clock with remote NTP servers or local reference clock) to attain relative synchronization among the sub-systems. This helps us synchronize events (state estimation, control actions) and accurately collect logs across the distributed subsystems. As there was no internet access in our network, I ensured that the clients were forced to use the master time as the reference. The approach was to configure the clients to step match the master on reboot and continue slew tracking to gradually synchronize local system-time, rather than abruptly changing it to avoid disruptions to time-sensitive processes that rely on the system clock.

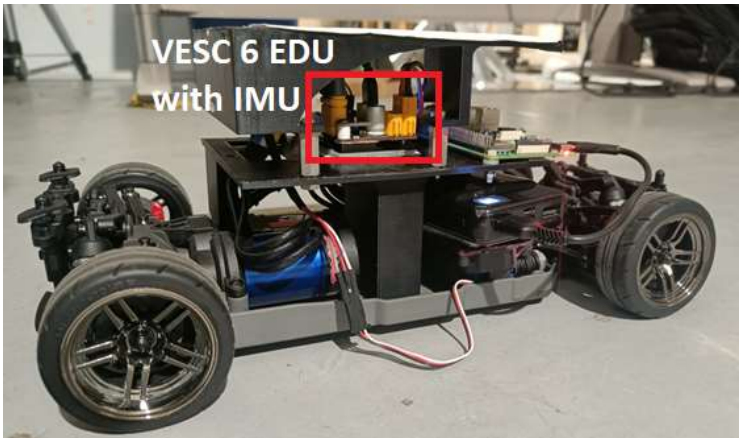

Figure 1: VESC integrated in the RC Car subsystem.

Challenges

VESC IMU driver: The current driver to interface with the VESC IMU is written for ROS2 and it is challenging to integrate this with our stack. We need to configure this to work in the ROS Noetic environment.

Driving the vehicle at speeds below 20cm/s: Currently the RC car can track the commands by the MPC block very accurately, but only above 20 cm/s. Due to the inertia in the motor and inertia of the car, it is physically not possible to drive the RC car below a certain speed smoothly. The issue is that when the upstream MPC control commands a velocity of below 20cm/s the RC car is not able to track it. One possible solution is to incorporate feedback from the IMU and build a cascaded low-level controller.

Teamwork

Atharv Pulapaka: Atharv had to update the visualization module to obtain real-time performance whilst testing the control subsystem. He also worked on integrating the state estimates from the perception module with the controls block and setup the central decision system to act on state estimates from the perception and generative a control profile to safely drive the RC car to the end goal. He also worked on building the PDB (Power Distribution Board).

Dhanesh Pamnani: Dhanesh worked on the lane detection algorithm. I also collaborated with him to build the mechanical support structures on the RC to develop a finalized mechatronic subsystem design for the RC Car.

Jash Shah: Jash has been working on setting up the embedded infrastructure. He faced a lot of challenges as the performance on the edge devices was not real-time. He thus tried other methods for object detection to lower the computational cost of the perception subsystem.

Ronit Hire: Ronit refined the tracking algorithm, velocity estimation module, and refactored the perception pipeline. He also collaborated with Atharv to ensure smooth integration of the perception and control subsystems.

Plans

I intend to work on the following:

1. Integrated system testing
2. Tune the low-level controller and use IMU feedback.