Automated Driving Using External Perception

Individual Lab Report

MRSD Project Course II

Fall 2023

Shreyas Jha

Team E: OuterSense

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

ILR06

Sep. 14, 2023

Table of Contents

Individual Progress

Building upon the foundations of my previous efforts on the RC car subsystem, networking, time synchronization, and embedded systems, my individual progress over the recent weeks has been focused on making some minor yet important improvements to the RC car subsystem.

One of the key pieces of work has been the development of startup scripts designed to streamline the configuration process for multiple RC cars. They enable us to test with more robustness, uniformity, and efficiency as they minimize manual configuration efforts. These scripts begin by conducting communication tests over the OuterSense WLAN network, ensuring secure communication with the RC cars. This capability is crucial for coordination and data exchange among the vehicles, especially as our project's scale grows. Moreover, the scripts automate the process of updating configurations for two critical components: Chrony and Multi-machine ROS. Chrony configuration is particularly vital for achieving time synchronization among our distributed subsystems. By automatically adjusting the configurations to align with the designated IP address of the cloud laptop for OuterSense, it provides flexibility for anyone with the OuterSense software to test on their own machines. It also ensures that all RC cars synchronize system clocks with the cloud machine (as well as the same time zone). Furthermore, the startup scripts initiate the essential processes for each RC car. They launch the VESC packages, responsible for motor control, and the control packages governing the vehicle's behavior. This automated startup sequence not only saves valuable time but also minimizes the risk of manual errors, ensuring that each RC car starts with the correct configurations.

I'd like to highlight a key feature of the RC Car subsystem: the timer-based trajectory follower. It addresses one of the project's critical challenges: inherent latency in our setup. By incorporating this trajectory follower, we account for this latency and enable our RC cars to follow predefined trajectories with precision. I dedicated efforts to enhancing the logging and debugging capabilities of our RC car software over the past week. Effective logging helps the team diagnose issues and monitor the system's performance. This logging uses the different verbosity levels provided by ROS so that it is easy to filter and prioritize messages based on their severity. I also addressed warnings related to the Raspberry Pi GPIO cleanup following an abrupt shutdown.

Another aspect of my individual progress involved configuring the Jetson Nano to stream data to the OuterSense network over LAN. This setup (shown in Fig. 1) was essential to optimize our data processing pipeline and reduce our reliance on multiple laptops during testing.

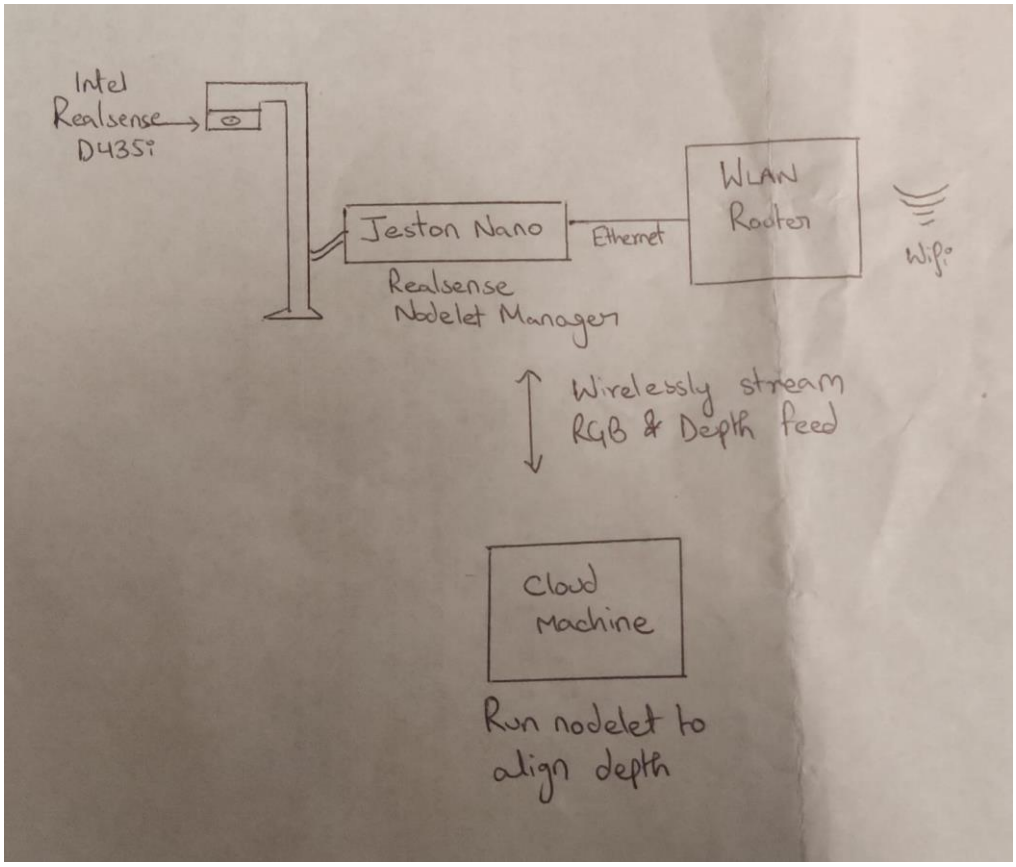

Figure 1: Using Jetson Nano to stream the perception feed.

Challenges

Jetson Nano as the edge device: The challenge was particularly related to frame rate drops when using the RealSense D400 series cameras in conjunction with the Jetson Nano. This challenge had been previously identified by Jash in Spring '23 and had proven to be an ongoing concern. To address this issue, I explored various avenues, including overclocking the computer to support vision frameworks and installing a compatible Jetpack and Ubuntu 20. However, despite these efforts, the frame rate drops persisted when attempting to use the RealSense package to output RGB frame-aligned depth feed, which is used by our pose estimation software.

Another thing I tried was to run the ROS nodelet, which fuses the data on the cloud (depicted in Fig 1.). Still, since it was still tied to the nodelet manager on the Jetson Nano, the processing happened on the Jeston Nano, and the frame rates dropped. One future option is to try porting the nodelet to a ROS node and decoupling it from the nodelet manager to run it locally on the cloud machine.

Notably, the frame rate issue with the Jetson Nano and RealSense camera has also been observed by others in the community, as evidenced by open issues on GitHub for Jetson + RealSense users. A definitive solution remains elusive.

Teamwork

Atharv Pulapaka: Atharv's work on scaling the trajectory follower has been invaluable. He has been exploring optimization techniques, both as a joint optimization problem and for individual car stacks. Our upcoming integration tests with multiple cars will provide insights into the effectiveness of these optimizations.

Dhanesh Pamnani: Dhanesh's efforts were focused to fix the low ride height of the cars which has contributed to their stability and functionality. He also provided logistical support for the team and performed the calibration of camera intrinsics. We collaborated to unit test all the mechatronic components of the customized RC cars, ensuring their reliability and performance.

Jash Shah: Jash has undertaken the responsibility of setting up the Gazebo environment for our project. This environment serves as a critical testing ground for planning and decision-making algorithms. His efforts include importing the Ackermann model and customizing it to work seamlessly with our control messages.

Ronit Hire: Ronit's work on the multi-object tracker, specifically looking into the implementation of Hungarian matching for data association to maintain stable tracking IDs. He has also been studying approaches to perform the global data association task.

Future Work

I intend to work on the following:

1. Managing the communication framework for multiple devices and subsystems
   This is centered around integration testing with cloud-based software.

2. On-board IMU data for the RC cars
   To further enhance our state estimation pipeline, I intend to explore Madgwick and Mahony fusion algorithms for IMU data. This step will also involve calibrating the VESC IMU.