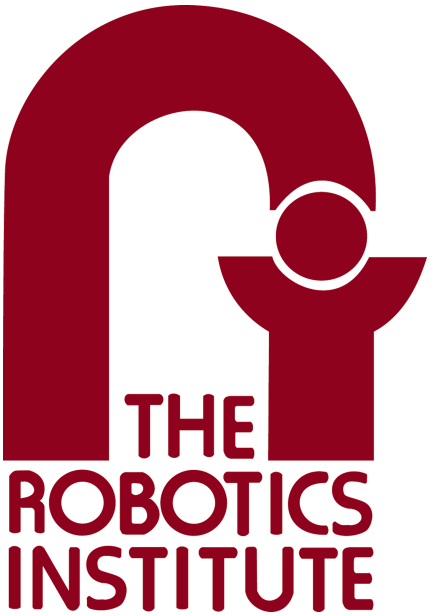


MRSD Project Course

Team I – AIce

Autonomous Zamboni Convoy

Individual Lab Report 3

Team

Rathin Shah

Nick Carcione

Yilin Cai

Jiayi Qiu

Kelvin Shen

Author

Rathin Shah

Mar 3, 2022

Contents

1. Individual Progress

Since the last Progress Review, I majorly worked in three domains:

- Building the RC Car platform for our use-case
- Integration of Lateral and Longitudinal Controller for Zamboni for path following simulation
- Converting remote control of RC Car to Manual and Microcontroller driven

The given activity led in successfully completing tests 4 and test 5.

1.1. Building RC Car :

We decided to get a RC Car that has Ackermann geometry to test and validate our algorithms until we get access to Zamboni. This was an important decision as we had to make sure that the development we do on RC car, is such that it can be built on when we deploy the algorithms on actual Zamboni, with no major change in the core algorithm and sensor integration. I undertook the task of assembling the RC car that was available in the inventory. The assembled RC Car is shown in figure 1 below.

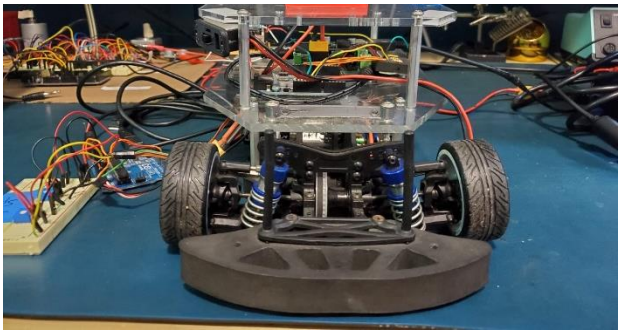
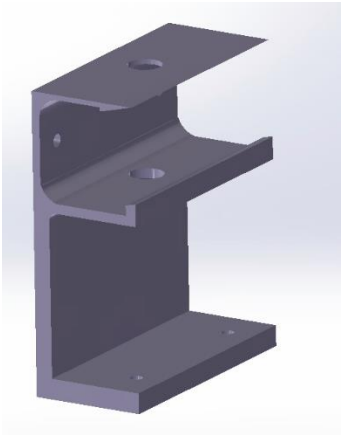


Figure 1. Assembled RC Car

One major thing was to fit the RC Car with RealSense Camera, in the same way, it will fit on Zamboni (to get the TF between vehicle base link and camera link) as close as possible to the actual Zamboni. So based on the URDF and dimensions that we had for the Zamboni in the simulation, we got the scaling ratio for all the dimensions between the RC Car and the Zamboni. I designed a mounting for the RealSense Camera and tried different mounting locations on the RC Car. For each possibility, I calculated the transform between the mounting frame and the RC Car's frame and verified the FOV of the camera that the position will give.

I iterated with 3 designs, as shown in the figure below. The third iteration was finalized as it not only gave a better FOV and a transform but also had a provision to fit a battery without consuming a lot of space on the RC Car. We have very limited space on the RC Car and must fit all sensors and actuators as well as jetson and battery all in one.



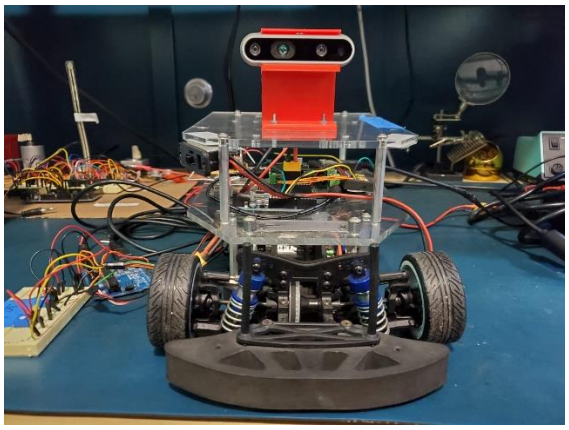


Figure 2. Camera Mount for RealSense Camera

1.2. Lateral and Longitudinal Controller for Zamboni for path following simulation

My major contribution in last two weeks was building a complete vehicle control system (lateral and longitudinal control) in Simulink which can control the longitudinal and lateral motion Zamboni in ROS simulation. Before PR1, I had developed a pure pursuit controller in Simulink and validated its performance in Simulink itself using a 3 DOF model that resembled Zamboni dynamics. I also integrated the ROS toolbox with it to communicate with our simulator. The work I did in this couple of weeks was built on this.

1.2.1. Longitudinal Vehicle Controller

The longitudinal controller is an important part of our drive-by-wire subsystem that can control the velocity of the Zamboni based on the waypoints and relative velocity of the leader Zamboni's velocity provided by the motion planning subsystem.

I developed a longitudinal controller block in Simulink that computes the acceleration and deceleration commands, in meters per second, that control the velocity of the vehicle. The input to the system was the reference velocity, current velocity, and current driving direction. The controller computes these commands using the Stanley method [1], which the block implements as a discrete proportional-integral (PI) controller with integral anti-windup.

The block uses this equation:

$$u(k) = \left(K_p + K_i \frac{T_s z}{z-1}\right) e(k)$$

- $u(k)$ is the control signal at the kth time step.
- Kp is the proportional gain,
- Ki is the integral gain,
- T_s is the sample time of the block in seconds, as set by the Sample time (s) parameter.
- $e(k)$ is the velocity error (CurrVelocity − RefVelocity) at the kth time step. For each k, this error is equal to the difference between the current velocity and reference velocity inputs (CurrVelocity − RefVelocity).

The control signal, u, determines the value of acceleration command AccelCmd and deceleration command DecelCmd.

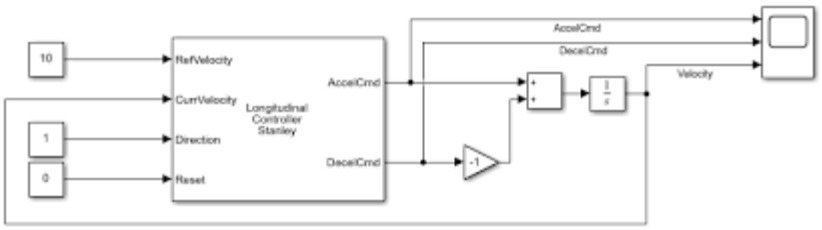


Figure 3. Longitudinal Vehicle Controller

I validated this controller by doing a unit testing of it, by providing reference velocity that leader might have and verified the acceleration output and velocity output with the actual output the vehicle would have.

1.2.2. Integration of Longitudinal and Lateral Controller into single Drive by Wire Controller

With the basic lateral and longitudinal controller developed, I focused on integrating both of them to develop a complete low-level drive by wire controller. The ROS toolbox of Simulink was also added to the controller to make it capable to communicate with our Zamboni Simulator.

The complete layout of the drive-by-wire controller is shown in the figure below.

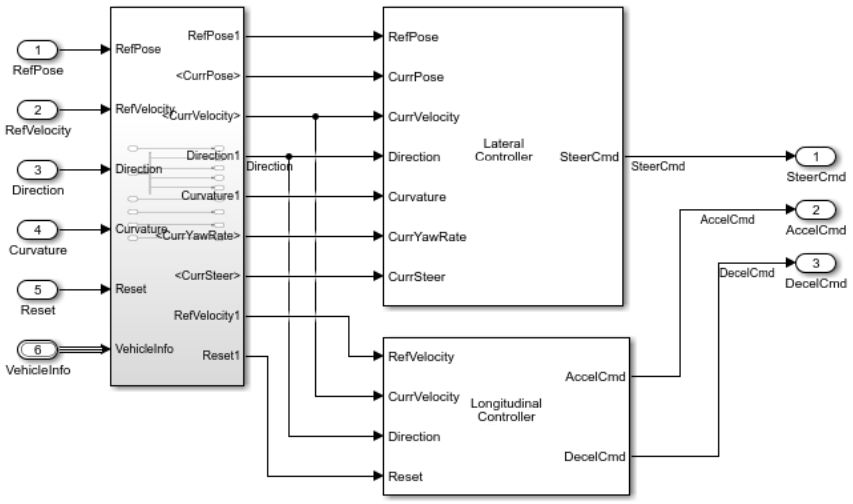


Figure 4. Drive by Wire Vehicle Controller

This integrated controller was unit tested by giving the input of reference pose and velocity and verifying the output of steering command and longitudinal command with the actual commands generated in gazebo by driving the Zamboni with same velocity and the same pose/curvature.

This integrated controller was equipped with ROS toolbox and a multi-ros network control was implemented by adding necessary publishers and subscribers based on our simulation rqt graph as shown in the figure below.

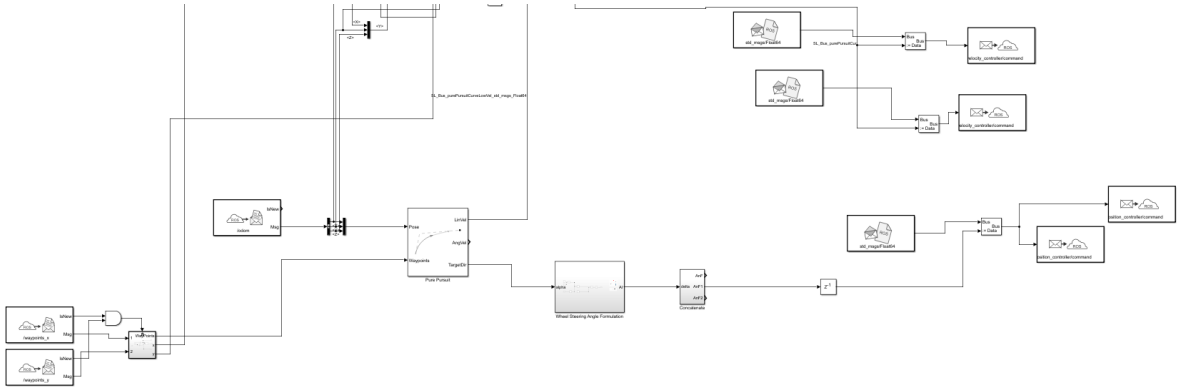
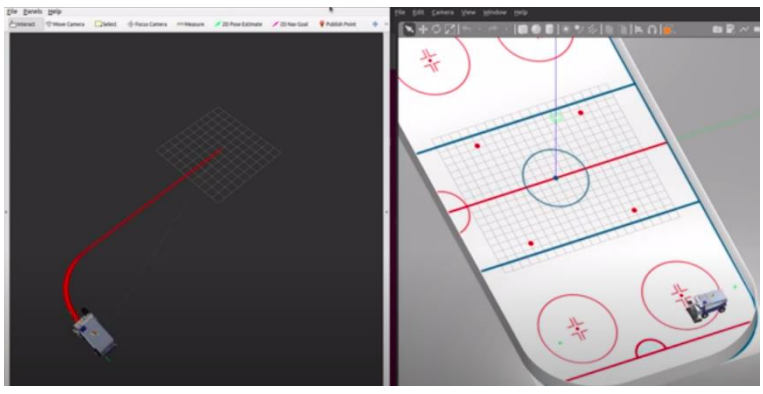


Figure 5. ROS Toolbox integrated controller

1.2.3. Co-Simulation with the ROS based simulator to implement autonomous path following

Once I had the vehicle controller setup, the next task followed was using it to implement the autonomous path following for the follower Zamboni in the simulation in Gazebo. The task was completed by subscribing to a topic on ROS that generated the waypoints and velocity profiler needed for controller. The vehicle controller generated the steering commands and velocity commands so that the Zamboni follows the reference trajectory with respect to the velocity profile that it needs to follow. Here, the necessary inputs needed for the controller were generated by manually publishing on the respective topics but in the final setup, it will be calculated by the perception and motion planning subsystem and then publish on topics. The figure below shows the path following of the Zamboni using the controllers developed in Simulink. Green path is the reference path provided and red path is the actual path taken by the zamboni.



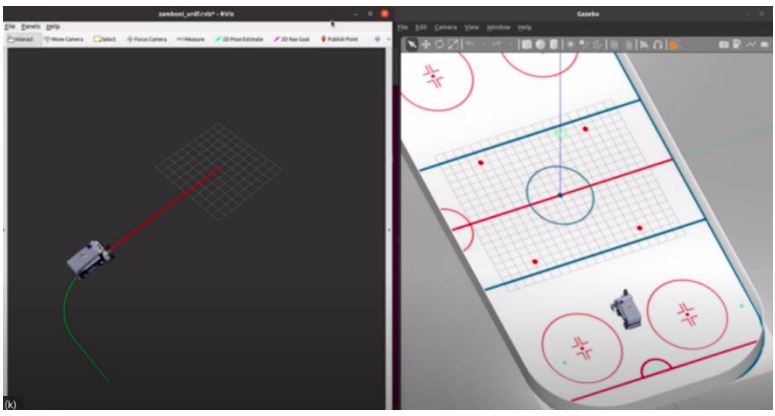


Figure 6. Autonomous Path Following of the Zamboni

Once the setup was complete, we validated the performance of the pure pursuit lateral controller and the longitudinal controller with the performance requirements we have. To

meet these requirements, we had to tune the pure pursuit controller which was heavily dependent on lookahead distance and how the waypoints are getting to be published. This had no feedback on the current steering and yaw angle of the vehicle. So, I also implemented a Stanley steering controller and validated the vehicle's performance against this controller.

By tuning the lookahead distance for the pure pursuit controller, we were able to get satisfactory performance for the path following. The Stanley controller also did a good job and with both the controllers the cross-track error was below 50 cm. The final controller will be decided based on hardware computation requirements later.

1.3. Conversion of remote control into microcontroller-controlled RC Car

I helped Nick in converting the RC Car into an electronically controlled car with respect to the steering servo control and the driving Brushed Motor Control.

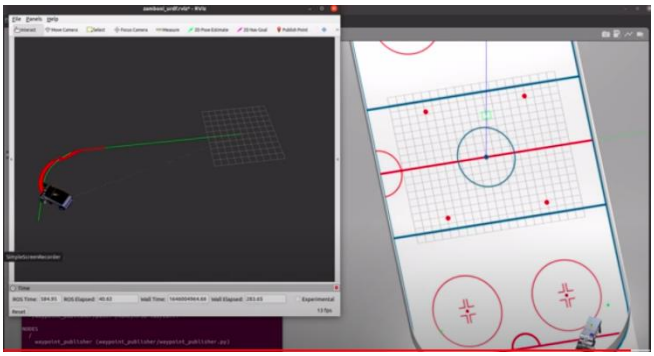
I helped him in mapping the steering angle of the RC Car to the servo motor angle. This activity was necessary to parametrize the lateral steering controller developed as per the RC Car kinematics so that we can autonomously control the steering of the vehicle

Figure 7. Steering angle mapping on RC Car

I also helped him with the microcontroller coding to control the actuators on the vehicle, specifically the ESC Speed controller for the Brushed Motor to control the velocity and also the servo motor for the steering angle.

2. Challenges

One of the major challenges was tuning of Lateral Controller for the Zamboni Dynamics for smooth and correct path following (Refer Figure)



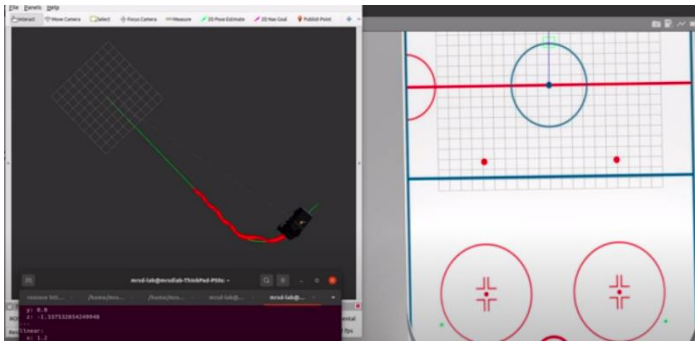


Figure 8. Poor Path following by vehicle

Also developing a velocity profiler with inline to Zamboni Dynamics was a little tricky as we had no tested data on the actual Zamboni.

Also there were challenges in ROS Subscriber Nodes on master server, they couldn't connect to the Nodes on MATLAB and we had to debug this issue for quite some time. Eventually we realized that it was an inbuilt issue of the ROS Toolbox and we had to physically set the environment in the ROS workstation and the MATLAB workstation.

Another significant challenge was the loop rate that MATLAB run with and the ros simulation run with, so we had to manually set the looprate of MATLAB by publishing the rostime and calculating the loop rate from that for the MATLAB.

Also due to complex Ackermann steering geometry , finding a mapping between servo angle and steering angle was difficult and we have to create small jig and approximately produce the results.

3. Teamwork

a. Kelvin Shen

Kelvin worked on making the URDF of the marker board using custom Gazebo texture and attached it to the rear of the leader Zamboni.He Retrieved ground truth positions of the marker board and the camera in Gazebo by looking up tf transforms and thus successfully implemented the perception stack in the simulation. He also calibrated RealSense D435i and validated the aforementioned pose estimation algorithm with the printed marker board

b. Nick Carcione

Nick majorly worked on getting the RC Car up and running. He coded the Arduino to send the steering and velocity commands to the RC Car from the program instead of RC remote. He also contributed in mapping the steering angle on the RC Car to the servo commands.

c. Yilin Cai

Yilin helped me integrate the waypoints loading and pure pursuit controller to realize path following in gazebo simulation along with Rviz visualization. He also Modified the dynamics parameters of the Zamboni in URDF to reflect the real physical parameters and tuned the PID parameters in ROS- Controller to improve the reaction performance when sending steering and velocity command in Gazebo

d. Jiayi Qui

Jiayi developed the leader velocity estimation algorithm using Extended Kalman Filter based on relative position and relative heading angle and also tuned the parameter of EKF and improved the estimation performance. She also helped in path following implementation by generating the waypoints and velocity commands needed for the controller.

4. Future Work/Plans

Following is the plan for the coming weeks:

1. Getting waypoints from the leader zamboni with lateral and longitudinal offsets
2. Integrate the perception package with localization and velocity estimation modules
3. Implementing a complete Leader-Follower Convoy in Simulation to validate all the algorithms
4. Validating Localization on the RC Car using Real Sense IMU and Wheel Odometry

5. References

1. https://www.mathworks.com/help/driving/ref/longitudinalcontrollerstanley.html#mw_e0c62d0f-29b0-4b4e-b27e-04180cae6694
2. https://dingyan89.medium.com/three-methods-of-vehicle-lateral-control-pure-pursuit-stanley-and-mpc-db8cc1d32081
3. http://ai.stanford.edu/~gabeh/papers/hoffmann_stanley_control07.pdf