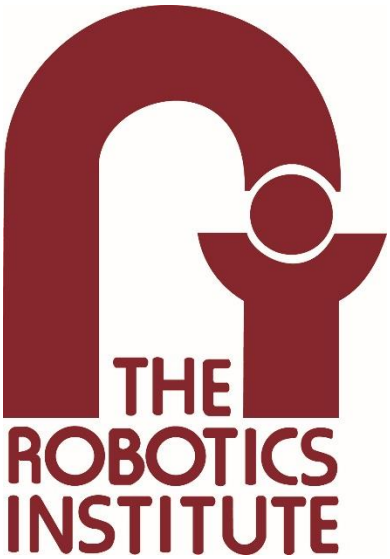

MRSD Project Course

Team I – AIce

Autonomous Zamboni Convoy

Individual Lab Report 04

Author

Nick Carcione

Teammates

Rathin Shah

Yilin Cai

Jiayi Qiu

Kelvin Shen

March 24, 2022

Contents

1. Individual Progress

My work since the last Progress Review has been focused on building out the capabilities of the RC car to make it a usable demonstration platform for our project. One step in doing so was adding an encoder to the car. The RC car as we received it did not contain an external encoder and had a brushed DC motor with an ESC that was unable to output any kind of encoder or odometry information. This meant we had to add our own external encoder, shown below in Figure 1, to the car for use in our localization algorithm.

Figure 1: Encoder added to RC car

To do so, it was necessary to design a mount to rigidly attach the encoder to the vehicle chassis and a gear to connect the encoder shaft to the motor shaft. The final mount design is shown in Figure 2 below and the parameters for the final gear design are included in Table 1 on the next page.

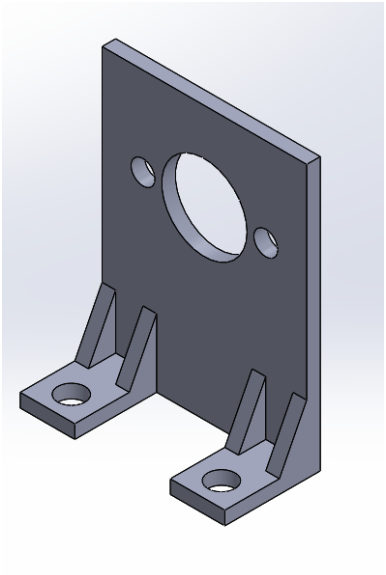

Figure 2: Encoder design

Table 1: Encoder gear parameters

Gear Parameters:	Value	Units
Number of Teeth, N	30	[-]
Diametral Pitch, P	48	[1/in]
Pressue Angle, α	20	[°]
Pitch Diameter, D	0.625	[in]

The mount was designed so that it would sit flush with the encoder face and line up with the bolt holes on the encoder face. It was also designed to allow the encoder to sit high enough to avoid colliding with the side of the chassis and to fit within the short amount of space available between the motor and the edge of the chassis. The gear was designed to mimic the gear that was already on the end of the motor shaft so that each motor shaft rotation would equal one encoder shaft rotation. A picture of the encoder mounted onto the car chassis is shown below in Figure 3.

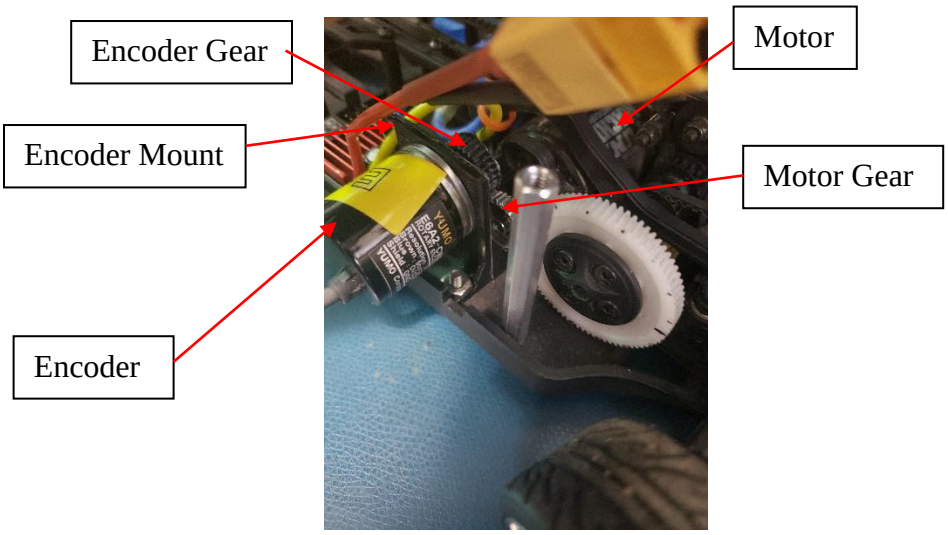

Figure 3: Final mounting of encoder onto RC car

Another step in getting the RC car ready for testing and demonstration was adding some basic waypoint following capability to it. For reasons that I will explain further in Section 2 "Challenges," the RC car initially was unable to follow a straight line; instead, even when commanded to have a 0° steering angle, the RC car would veer to the right. After spending some time testing different offsets for the servo motor that actuates the steering linkage, I found that defining a servo angle of 137° as the zero-steering angle (instead of 135°, or vertical) resulted in the best straight-line-following performance.

With the car able to follow a reasonably straight line, Rathin and I then proceeded to do some basic waypoint testing. We would mark a physical point on the ground at a known location relative to the center of the rear axle of the RC car and pass these coordinates along with a desired yaw into a Stanley controller that Rathin had developed. The Stanley controller would

compute the constant steering angle that the front right wheel had to be at to achieve the desired pose and we would pass that angle to the RC car using an Arduino Bluetooth module. The RC car would then be given a constant velocity and we would watch to see if it reached the desired location. We repeated this process multiple times at multiple locations to test the output of the Stanley controller and determine if any further steering angle offsets were necessary.

2. Challenges

A major challenge that I encountered while trying to get the RC car to follow a straight line was the wheels of the car not being aligned or straight. When looking at the car straight on, there was a slight but visible offset on one of the front wheels. This caused the RC car to drift to the left even when a zero steering angle command was sent. After looking through the code and documentation from previous MRSD teams that used this platform, I found a note that the car had been in a crash in the spring of 2016 which caused the team that was using it to add an offset to their steering controls. After similarly adding an offset to our steering controls, I was able to get the car to drive straighter, but it still tends to slightly drift. This is because the servo motor that actuates the steering linkage is what gets controlled, and the motor is only capable of taking integer inputs. At the current offset, the vehicle drifts slightly left and at an offset one integer higher, it will drift right.

Finding a way to mount the encoder onto the RC car also posed a challenge. As it is, the body of the car is quite crowded and there is not much free space to add extra components. On top of that, the encoder would have to mate with one of the gears of the drive train, and many of these are not exposed in a way that is easily accessible. The solution I came up with was a shortened mount that could fit within the perimeter of the chassis and would hold the encoder at a height so that it could both mate with the gear at the end of the motor shaft and avoid being in collision with the side of the chassis. The physical mounting of the encoder was itself another challenge. Since the encoder mount and gear had to be small, 3D printing these components to a suitable accuracy was difficult and required multiple prints. To mount the encoder, I had to drill mounting holes through the plastic body of the RC car and doing so ended up being tougher than I originally envisioned. The holes that I did drill ended up being slightly misaligned, which meant I also had to go back and widen the drilled holes to get them to line up with the holes in the mount.

3. Teamwork

Rathin Shah

Rathin helped me in getting the encoder gear designed and the encoder mounted onto the RC car. He also finalized the design of a PDS PCB that will be included on the RC car to provide power to the various sensors and components. Finally, Rathin worked with me in

conducting the RC car waypoint following tests and testing out the Stanley controller that he designed.

Yilin Cai

Yilin worked on finishing up the simulation of the leader-follower convoy. He set up the simulation and visualization for the demonstration of the simulated system. He also generated the path taken by the leader by integrating localization and perception data and used this to test the accuracy of the simulated perception system. He then planned the follower's motion based on the path of the leader that he found.

Jiayi Qiu

Jiayi implemented an algorithm to estimate the desired velocity of the follower in real-time using an Extended Kalman Filter (EKF) and published the output of this algorithm using a ROS node. She also created an algorithm that would generate a local trajectory for the follower that will be passed to the low-level controller. Finally, she worked with Yilin to simulate the leader-follower convoy using a pure pursuit controller.

Kelvin Shen

Kelvin continued development of the perception subsystem. He expanded the perception system's capabilities in simulation by adding a Velodyne Puck to the Gazebo simulation and processing the resulting point cloud to generate two distinct clusters, one for the leader and one for the follower. Additionally, he helped to fix the transforms generated by the camera in the simulation. In real world development, he created a physical ArUco board and worked with Rathin on calibrating the RealSense IMU with Rathin.

4. Plans

My goal for the next Progress Review is to finish developing the RC car, which will include integrating three main components. The first of these is the encoder. Now that it has been successfully mounted onto the chassis, I will work on integrating it into the Arduino's circuit and adding its functionality to the Arduino code. The other two items I will integrate include the Jetson and the RealSense. Additionally, I will help design a mount for the Jetson. Finally, I will help run real-world validation tests for the localization and leader velocity estimation algorithms on the RC car.