MRSD Project Course

Team I – AIce

Autonomous Zamboni Convoy

Progress Review 2

Team

Rathin Shah

Nick Carcione

Yilin Cai

Jiayi Qiu

Kelvin Shen

Author

Yilin Cai

Mar 3, 2022

Contents

1 Individual Progress

My responsibility for the Autonomous Zamboni Convoy project mainly focus on setting up and maintenance of the simulation environment. My work also includes defining the whole software architecture, integrating different algorithm modules, and providing algorithm interface from simulation environment.

1.1 Simulation refinement

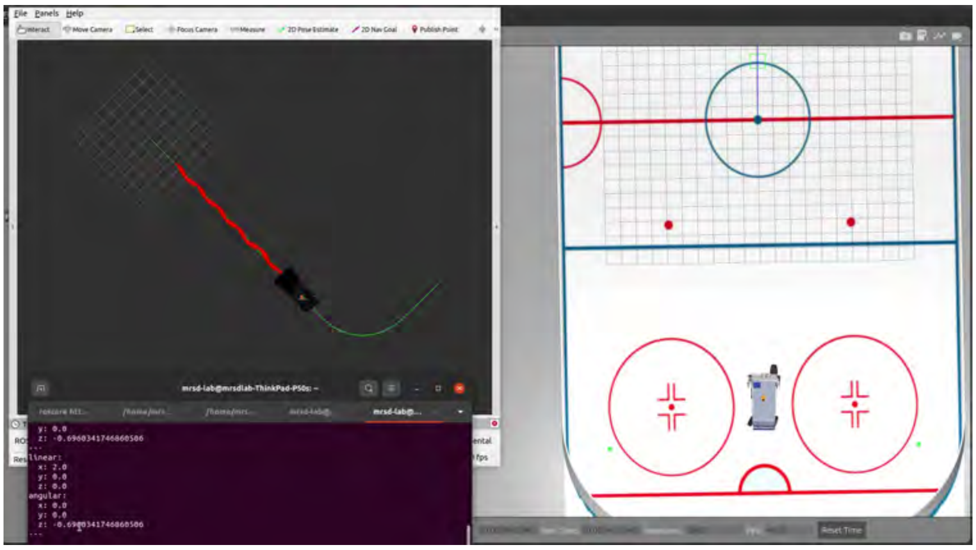
As the Zamboni vehicle with an Ackermann steering mechanism has been defined in simulation, so here I first refined the detailed parameters of the vehicles. I looked up the data sheet of the Zamboni and got the actual mass and physical dimensions. Then I used a box to define the collision property of the vehicle, and defined the inertial based on the cuboid. In addition, the collision, mass and inertial properties for the wheels are also defined according the data sheet for the real vehicle. Another important parameter is the interaction in gazebo, especially the interaction between the ground and vehicle wheels. I found if it is not set to a reasonable value, the wheel will slip in place. The vehicle would not move forward until reaching a very high velocity. I set the friction between wheels as well as the *mindepth*. The *mindepth* is how far the link can penetrate into another model/link before that corrective force is applied by the physics engine. This is useful if I want to ensure continuous contact between surfaces, which is required for wheels.

In addition to the model definition, I also refined the gazebo controller itself. Since I will command the motion of the vehicle by sending steering and velocity command to gazebo, I used gazebo_ros_control Gazebo plugin. It provides a pluginlib-based interface to implement custom interfaces between Gazebo and ros_control for simulating more complex mechanisms. The ros_control packages takes as input the joint state data from your robot's actuator's encoders and an input set point. It uses a generic control loop feedback mechanism, typically a PID controller, to control the output, typically effort, sent to your actuators. ros_control gets more complicated for physical mechanisms that do not have one-to-one mappings of joint positions, efforts, etc but these scenarios are accounted for using transmissions. By tuning the PID, the Zamboni vehicle finally react very fast after I sending the command.

1.2 Waypoint following

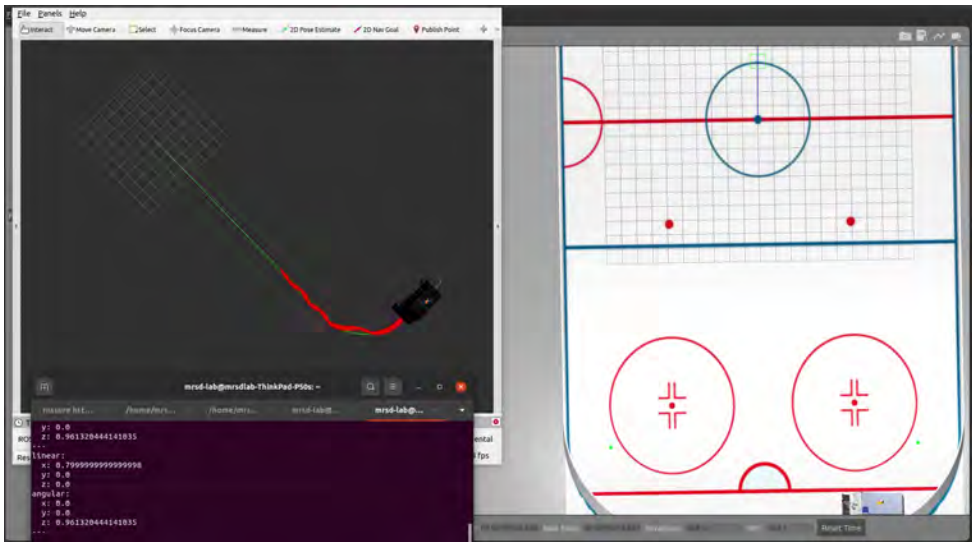
Based on the pure pursuit controller developed before, I integrated the controller and the waypoints loader along with the simulation environment. First, the simulink controller will subscribe the odometry topic published by simulation. Second, we pre-planned the motion for the vehicle, including the waypoints and velocity profile. The controller will also subscribe the waypoints published at a certain rate. Third, the controller will sent velocity and steering angle to the vehicle to drive it following the path.

As shown in Figure 1 and Figure 2, the green line is the planned path for the vehicle and the red line indicates the real path. It is found that alone the straight line, the steering angle is actually fluctuate. Although the vehicle is moving forward, the real path is not actually straight. Same failure happens in the turning phase, as shown in Figure 2. Although the steering angle is not that stable, it still follow path turn arounf.



```
  y: 0.0
  z: -0.6960341746860506
---
linear:
  x: 2.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: -0.6960341746860506
---
```

Figure 1: Zamboni path following failure 1.



```
  y: 0.0
  z: 0.961320444141035
---
linear:
  x: 0.7999999999999998
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.961320444141035
---
```

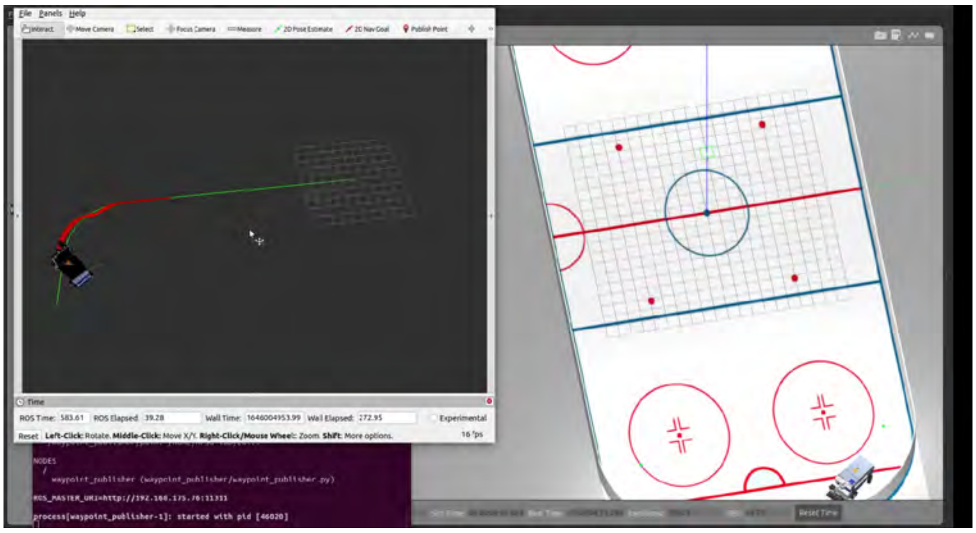
Figure 2: Zamboni path following failure 2.

The reason for this failure is the incorrect look ahead distance. Figure 3 indicates another failure that the the vehicle fail to turn 90 degrees along the path because the controller get the new desired waypoints before it actually reaches the correct waypoint.

By tuning the controller parameters, we finally successfully controlled the vehicle to follow the path, as shown in Figure 4.

2 Challenges

The first major challenge lies in the URDF setup and gazebo engine. To reflect the dynamics of the Zamboni in gazebo simulation, I tuned the physical parameters in URDF, e.g. mass, inertial and their origin points. The mass and inertial values are taken from the Zamboni data sheet which reflect their actual properties. However, after changing the parameters, the Zamboni in gazebo will sink into the ground. Sometimes, the wheels even fly into the sky. By testing repeatedly the influence of each parameter, I found the problem lies in the definition of inertial.



ROS Time: 583.61 ROS Elapsed: 39.28 Wall Time: 1646004953.99 Wall Elapsed: 272.95

Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel:: Zoom. Shift: More options.

```
NODES
  /
    waypoint_publisher (waypoint_publisher/waypoint_publisher.py)

ROS_MASTER_URI=http://192.168.175.70:11311

process[waypoint_publisher-1]: started with pid [46020]
```

Figure 3: Zamboni path following failure 3.

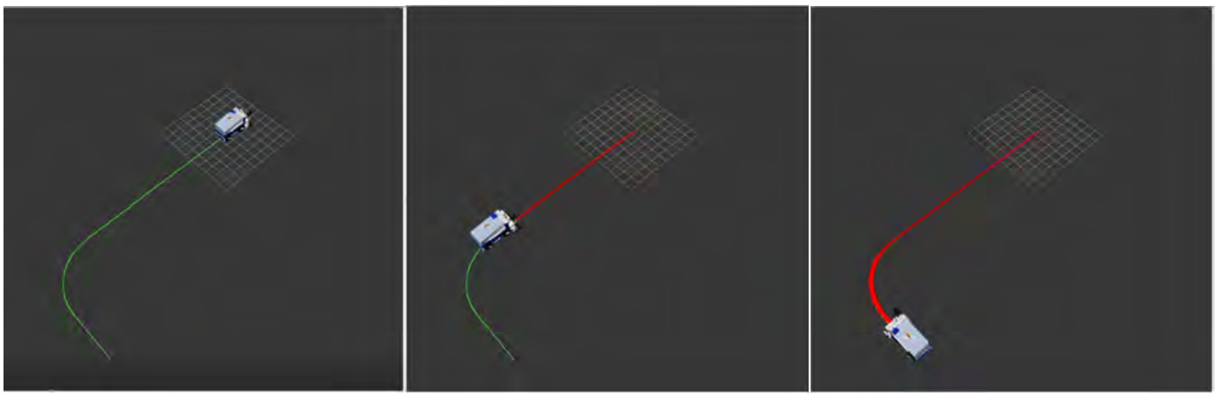


Figure 4: Success Zamboni path following.

For simplification, only the wheel links and the chassis link in the URDF are given the mass and inertial. The parameters for other links like "steering hinge links" are ignored. I defined a Marco called "Null inertial element" which is needed to make the model work with Gazebo. The mass and inertial element value is set to 0.01 which is extremely small compared to other parts. Due to the gazebo inbuilt physics engine, there will be computing numerical errors. The order of magnitude of such errors is comparable to 0.01, causing the failure of definition of the links. When I change 0.01 to 0.1, problems then get solved and the vehicle will drive normally. The experience we learnt from this tuning is that we should always take care of the small value in simulation system, which could cause numerical errors.

The second challenge lies in the integration of pure pursuit controller developed in simulink into the gazebo simulation environment. pure pursuit will send steering and velocity commands to the vehicles to make it follow the waypoints. To test the performance of the controllers in gazebo, we need to reduce the error caused by gazebo itself. However, when I use rob_control_plugin with bad PID parameters, the steering angle will fluctuate after getting the command from the pure pursuit. Such fluctuated value will also be feed back into the controller, causing more error. By tuning the parameters, U finally achieved a great performance.

3 Teamwork

Kelvin Shen is in charge of perception and recognition. His progress includes:

- Remade the URDF of the marker board using custom Gazebo texture and attached it to the rear of the leader Zamboni
- Retrieved ground truth positions of the marker board and the camera in Gazebo by looking up tf transforms
- Tested marker board pose estimation algorithm when both Zamboni's are moving
- Tested interpolated depth algorithm when both Zamboni's are moving as well as when the board is partially occluded
- Calibrated RealSense D435i and validated the aforementioned pose estimation algorithm with the printed marker board

Rathin Shah is in charge of the controller development. His contribution includes:

- Implemented Path Following for the Zamboni using the waypoints provided and the vehicle controller developed in simulink.
- Designed and 3d-printed RealSense Camera mount.
- Mapped Steering servo angle to RC Car steering angle

Nick Carcione is in charge of the RC car hardware. His contribution includes:

- Got the RC car up and running.
- This included figuring out how to steer the wheels and send speed commands to the motor.
- Made it so that the RC car can be controlled via manual inputs or computer commands.

Yilin Cai is in charge of the simulation setup. His contribution includes:

- Modify the dynamics parameters of the Zamboni in URDF to reflect the real physical parameters.
- Tuned the PID parameters in ROS- Controller to improve the reaction performance when sending steering and velocity command in Gazebo.
- Integrated waypoints loading and pure pursuit controller to realize path following in gazebo simulation along with Rviz visualization.

Jiayi Qiu is in charge of the leader's velocity estimation and wayponits setup. Her contribution includes:

- Developed the leader velocity estimation algorithm using Extended Kalman Filter based on relative position and relative heading angle.
- Tuned the parameter of EKF and improved the estimation performance.
- Integrated the algorithm with simulation as a ROS node.
- Designed and published waypoints including positions, heading angles and velocities for controller testing and PID tuning.

4 Plans

The next step of my MRSD project works will still focus on the simulation part. I will work on the perception integration. Specifically, I will retrieve the waypoints based on the relative pose of the leader vehicle estimated by camera of the follower. Then based on the follower's localization and the relative, generate the waypoints for the follower. Then I will design a waypoints optimizer to check and publish the waypoints to the controller. Moreover, I will integrate the velocity estimation of the leader into the simulation and provide such information to controller. Then for next progress review, I believe we can realize a basic leader-following motion.