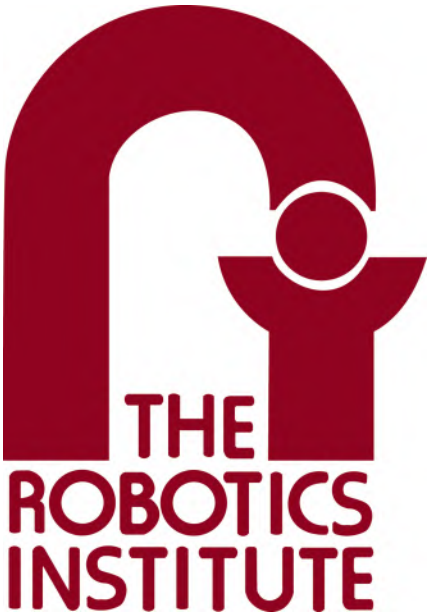

MRSD Project Course

Team I – AIce

Autonomous Zamboni Convoy

Progress Review 3

Team

Rathin Shah

Nick Carcione

Yilin Cai

Jiayi Qiu

Kelvin Shen

Author

Yilin Cai

Mar 24, 2022

Contents

1 Individual Progress

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

1.1 Subsystem integration in simulation

In follower autonomy simulation part, we have already developed follower localization, leader perception, leader velocity estimation, low level controller for follower as well as the whole simulation setup in the previous efforts. In this progress view, I integrated all the algorithms and software subsystems together and finalized the simulation visualization. The follower localization algorithm provides the current pose and velocity of the follower Zamboni. The perception of the leader provides the relative pose between the follower, which can be used to recover the leader's path combined with the follower's localization. The leader velocity estimation provides the velocity of the leader Zamboni which will be set as the target velocity in controller (we then converted the algorithm to follower desired velocity estimation).

The simulation visualization is shown in Fig. 1. I visualize both vehicles in Rviz and gazebo, and also the different paths in Rviz: Green path - Estimated leader path; Red path - Planned follower desired path; Blue path - Planned follower local path to track; Yellow path - Follower actual path from odometry. The Rviz reflects how well the follower tracks its desired path and how well the follower maintain the desired distance between them.

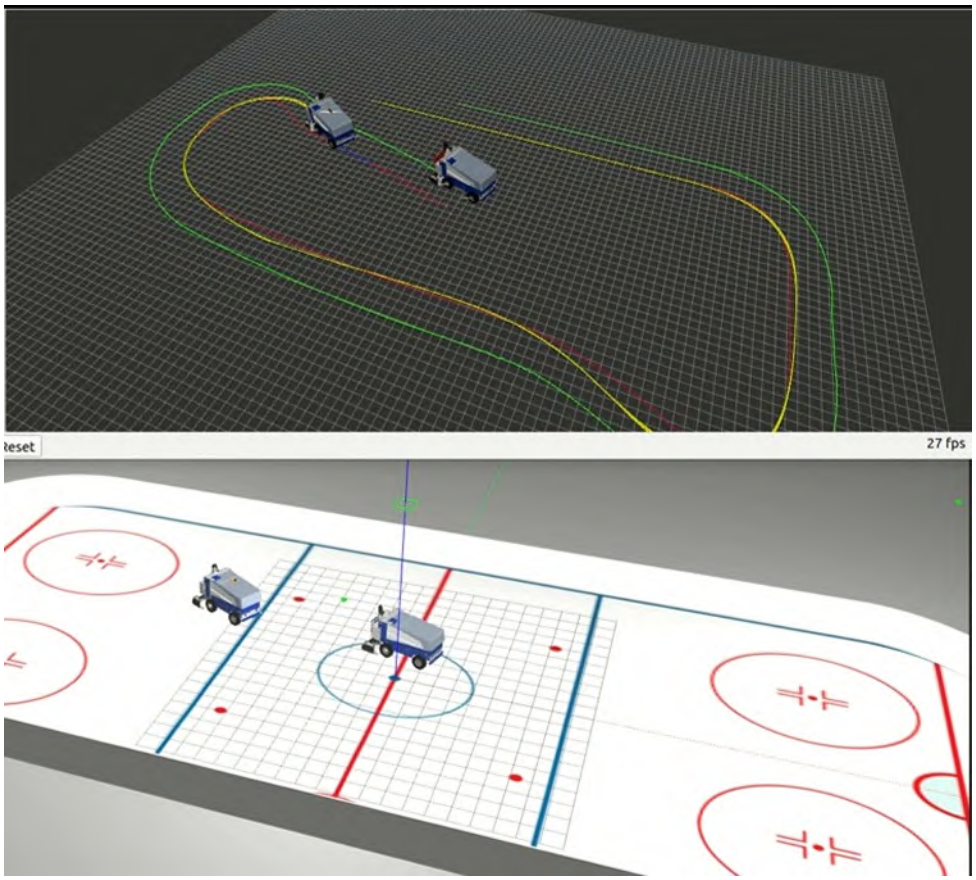

Figure 1: Simulation integration and visualization.

1.2 Motion planning

The motion planning for the follower means that we need to plan the motion profile for based on the localization, perception and estimation, as motioned above. Fig. 2 gives the flow of the motion planning process.

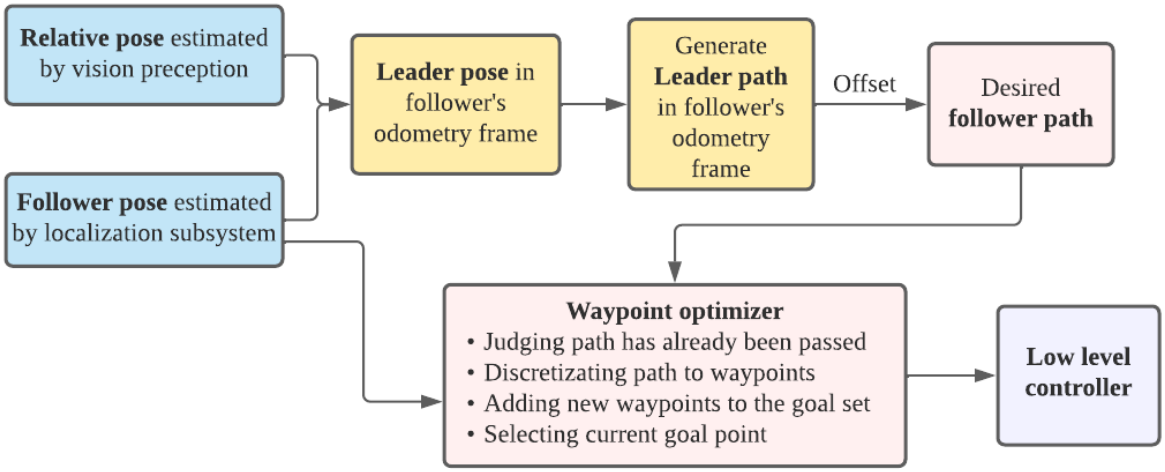

Figure 2: Motion planning for follower Zamboni.

We generate the desired path for follower Zamboni based on relative pose estimation and follower localization. Fig. 3 gives the frame transformation from the follower base footprint link to the leader base footprint link. The pose of follower's footprint base is in the odometry frame derived by wheel odometry. And the transformation involves results from tf broadcast and vision perception. After leader path recovery, we add a lateral offset to the path to get the desired path for the follower, because we want the two Zamboni going along different paths. Then the waypoints optimizer includes waypoints updater and waypoints publisher. It will select waypoints for follower tracking based on the follower's pose and velocity. We then update the current targeting waypoints to the low level controller by finding the closest waypoint in the path to follower's current pose. Then we extract the local path with constant waypoint numbers for controller to track.

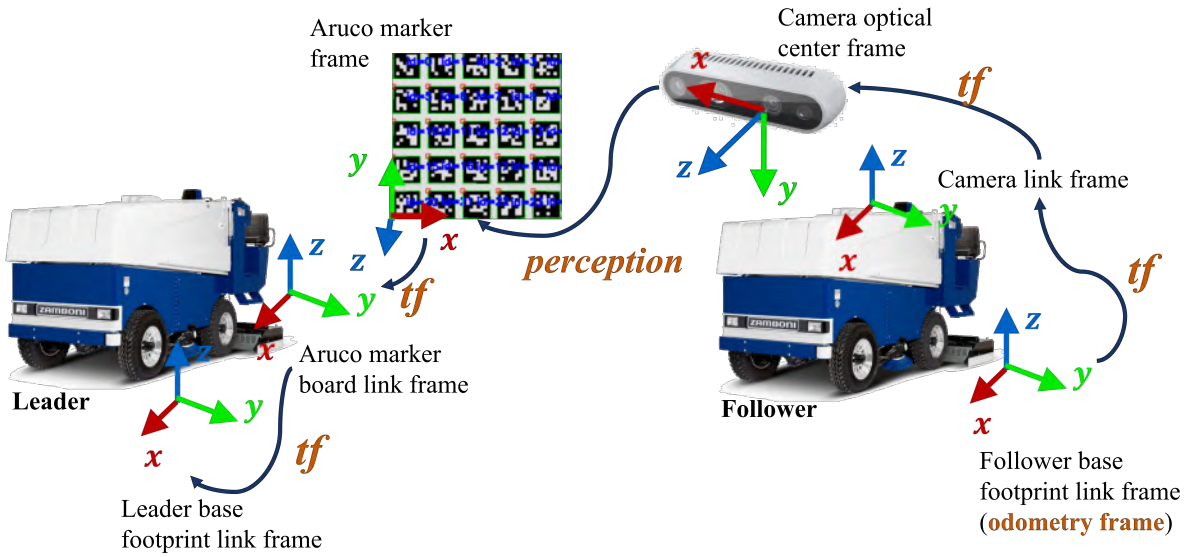

Figure 3: Frame transformation in leader path generation.

1.3 Autonomous convoy

After integration and simulation setup, I conducted the demo of the autonomous Zamboni convoy. Figure 4 shows the three moments of the demo of leader following. The leader Zamboni's steering angle and velocity were controlled by keyboard teleportation. And the autonomy software subsystem will retrieve the leader's path, do motion planning for follower and control the follower to track the planned path to follow behind the leader in real time. So the actual input to the follower's pure pursuit controller are the local waypoints (x, y and yaw), follower's current pose and follower's desired velocity.

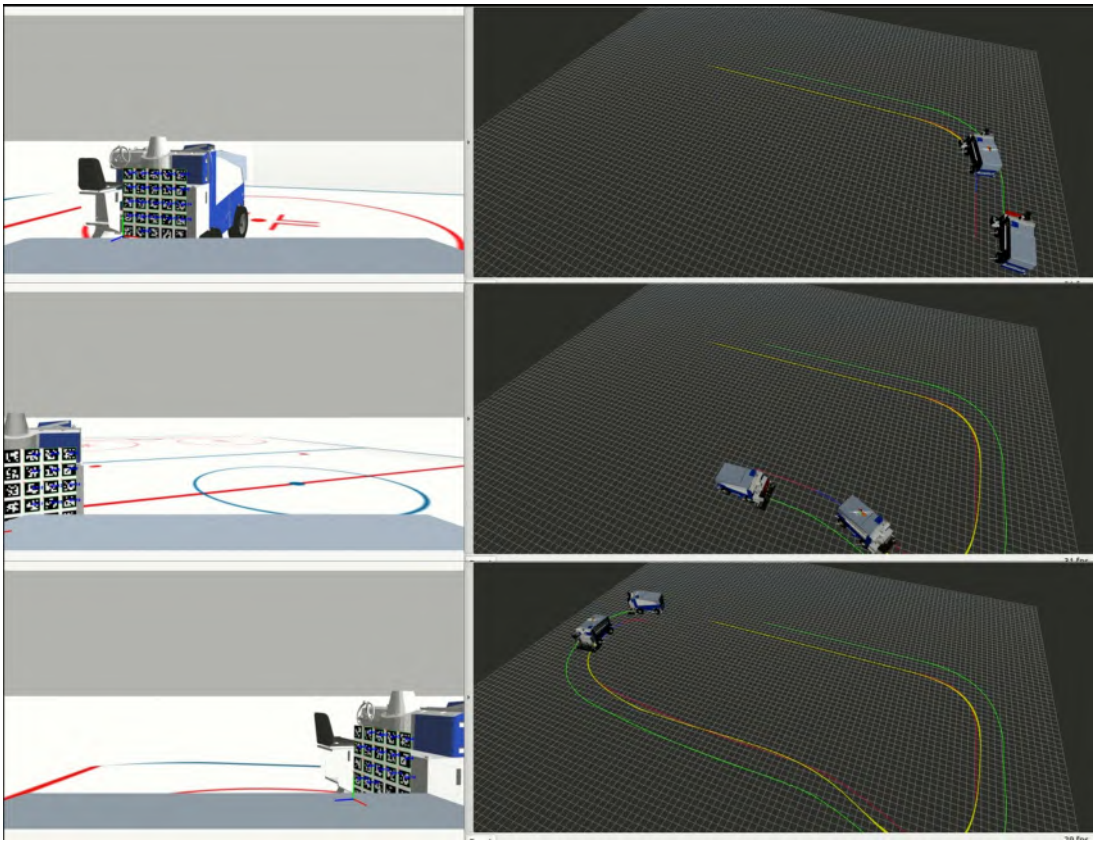

Figure 4: Autonomous convoy of the Zamboni.

2 Challenges

The first major challenge lies in the coordinate frame transformation as shown in Fig. 3. At the beginning, we took the transformation matrix returned by the perception algorithm as the transform from the board frame to the camera frame, according to the documentation of OpenCV. However, we found that there was very high error when comparing the estimated leader pose to the ground truth. After debugging, we realized what the OpenCV document really means is that it returns the pose of the broad frame in the camera's optical frame. Then we corrected the transform matrix chain and get correct and accurate estimation.

In the autonomous convoy demo, we still use the ground truth value of the relative pose between the leader and follower. Because we actually figured out the frame transformation mistake after we finished the demo using the ground truth. Another reason is that we found sometime

the follower lost the marker board of the leader in the camera's field of view. The solution to this will be discussed in the section of Plans.

The second major challenge is the motion planning and controller. The pure pursuit controller takes in the waypoints and the velocity profile at the waypoints. However, we set the initial position of the leader in the front of the follower. And since the follower's desired path is generated by add the lateral off set to the leader's pose, there will be blank gap between the follower's initial position and the starting point of its desired path. We solved this by sending a constant speed to make the follower go straight to the starting point to the path. However, this could cause variation of the longitudinal distance of the leader and the follower. We are targeting a better solution of the longitudinal controller.

The third challenge is the velocity profile of the follower. We realize that if we directly send the estimated velocity of the leader to the follower directly, it could help the follower to keep a constant distance with the leader when going along a straight path. However, if they are taking turns, the radii of the leader and follower's paths are different. So if they still have the same speed, we distance between them will change. So we directly estimate the desired speed of the follower based on estimated desired path.

3 Teamwork

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

- Set up Velodyne's Puck in simulation. Converted the published PointCloud2 message into PCL format. Downsampled, filtered, and clustered the point cloud to get only two clusters, one for the leader and the other for the follower.
- Fixed the incorrect transform from the board frame to the camera frame with Yilin.
- Printed a 42in-by-42in marker board with the help of SCS Poster Printing.
- Calibrated IMU on RealSense with Rathin.

Rathin Shah is in charge of controlling the real RC car. His contribution includes:

- Gear design for encoder.
- PDS design for rc car power distribution.
- Waypoint follower testing and stanley controller validation.

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

- Designed mount and gear for encoder.
- Mounted encoder onto RC car.
- Figured out how to get RC car to move in (sort of) a straight line.
- Did waypoint testing with Rathin.

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

- Setup simulation and visualization for autonomous convoy demo.
- Generate the leader's path by integrating the localization, perception and tf broadcast. Tested with Kelvin the perception accuracy in path generation by comparing with ground truth.
- Planned motion for follower Zamboni based on the leader's path.
- Integrated autonomy subsystems to conduct the convoy demo.

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

- Implemented real-time follower desired velocity estimation using EKF.
- Implemented ROS node to publish the follower's desired trajectory.
- Generated follower's local trajectory for low-level control.
- Simulated leader follower convoy using pure pursuit control with Yilin

4 Plans

The next step of my MRSD project works will still focus on both the simulation and hardware part. I will improve the controller to actually controller the longitudinal distance between the leader and follower, which I hope can also solve problems with the blank gap of the follower's initial position and the starting point of the path. I will also integrate the ROS code to the hardware using Navidia Jetson. We also plan to add another camera to solving the issue of losing the leader in the camera view.