F1/10 Autonomous Racing

Team Assignment 3 - CS4501

Madhur Behl (madhur.behl@virginia.edu)

Obstacle Avoidance and Follow-the-Gap [FTG]

Max points: 120

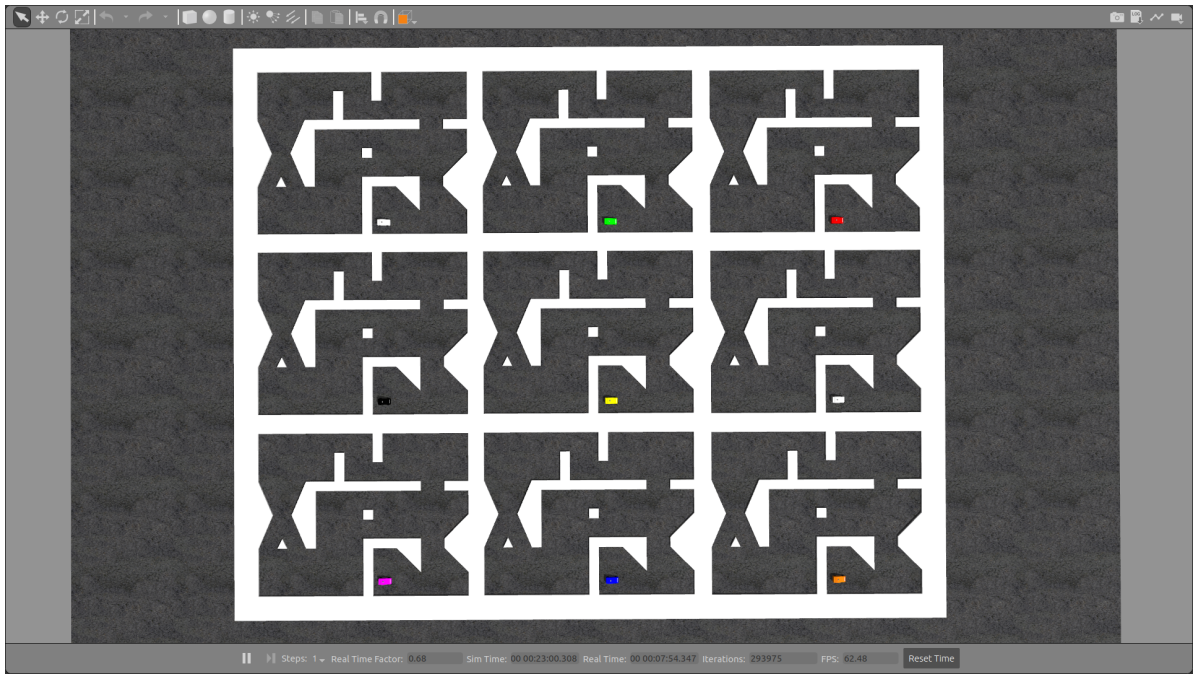
Due Date: May 13, 2021 at Noon (ET)

Overview

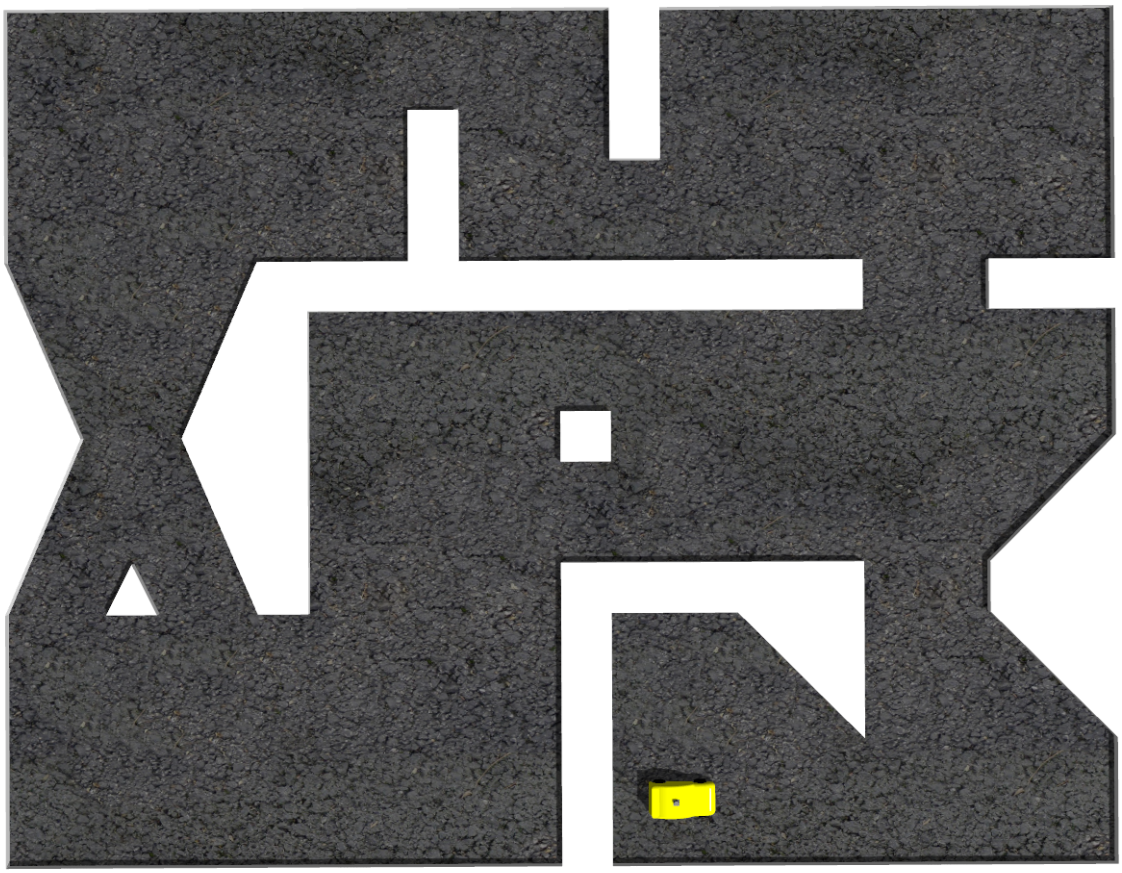
In this assignment you will extend your Wall Following autonomous driving method to implement obstacle avoidance, possibly using the Follow the Gap (FTG) reactive algorithm for autonomous racing. A special race track with static obstacles has been uploaded to your team account for this assignment. The goal of the assignment is to successfully complete ONE autonomous lap on the modified race track.

Modified Race track

The track layout for the online F1/10 simulator has been changed to the following:



Here is the top view of the single race track:



No template code is provided for this assignment.

Here are some suggestions for subroutines and callbacks that you may want to implement. You are free to implement your own method, this is just a recommendation.

Subscribe to the LaserScan from the car just like with Wall Following.
For the callback function:

- **Forward Field of View**: Prune the ranges array to angles corresponding to the forward looking (say 180 degrees or smaller) field of view only.
- **Gap Calculation/Disparity Detector:** Calculate the edges or the disparities in the LIDAR data. Scan each pair of subsequent LIDAR range values, and return an array of indices where the difference between the two values is larger than a threshold. The returned array contains the list of indices where such edges/disparities exist. For a refresher on gap calculation and disparity detection refer to Lecture 11 video on Follow the Gap (FTG): https://youtu.be/AywLjJTPRGQ
- Alternatively, scan the range array to determine the closest obstacle.
- Inflate or extend the obstacle/track boundaries to account for the car width. Go through the disparities or the closest obstacle and extend the disparities. Use the length of an arc at a distance to estimate how many values need to be overwritten and in which direction - left or right of the edge.
- Calculate heading angle - return the angle FTG will be targeting depending on which index corresponds to the farthest distance in the filtered/inflated ranges data. Also calculate the distance to the farthest target.
- Threshold the angle if it's larger than the sharpest turn you can make.
- Scale the speed in accordance to the forward distance.
- Publish the speed and steering angle

- You are free to use the /team_name/ground_truth topic to help you out.

As always:

- [ ] Create a launch file for your demo
- [ ] Try to implement variable velocity control instead of constant velocity control
- [ ] Add whatever ROS nodes that are required to the existing `race` package.
- [ ] Update the code on your team's Git repo.
- [ ] Record your successful runs in case of technical issues during the live demo phase.
- [ ] Start early on the assignment
- [ ] Work as a team
- [ ] Try not to crash !