F1/10 Autonomous Racing

Assignment 3 - CS4501

Madhur Behl (madhur.behl@virginia.edu)

Follow-the-Gap [FTG]

Due Date: May 11, 2020 at Noon (ET)

Overview

In this assignment you will implement the Follow the Gap (FTG) reactive algorithm for autonomous racing. A special race track with static obstacles has been created in Gazebo for this assignment. The goal of the assignment is to sucessfully complete an autonomous lap on the modified race track.

Simulation and track setup

You should use the standalone simulaotr to complete this assignment. Follow the steps below to install the modified track:

[Step 1] Delete any previous versions of the simulator.

If you created a new workspace previously to install the standalone simulator from f1tenth.dev then the easist way is to simply delete that workspace.

Instead, if you had downloaded the simulator as a package in your `src` sub directory in the `catkin_ws` then just remove the directory.

[Step 2] Clone and build the sim version with the new track.

Inside your `catkin_ws/src` directory clone and build the following repo:

```
git clone http://github.com/f1tenth-dev/simulator
```

[Step 3] Copy the track files into the Gazebo directory:

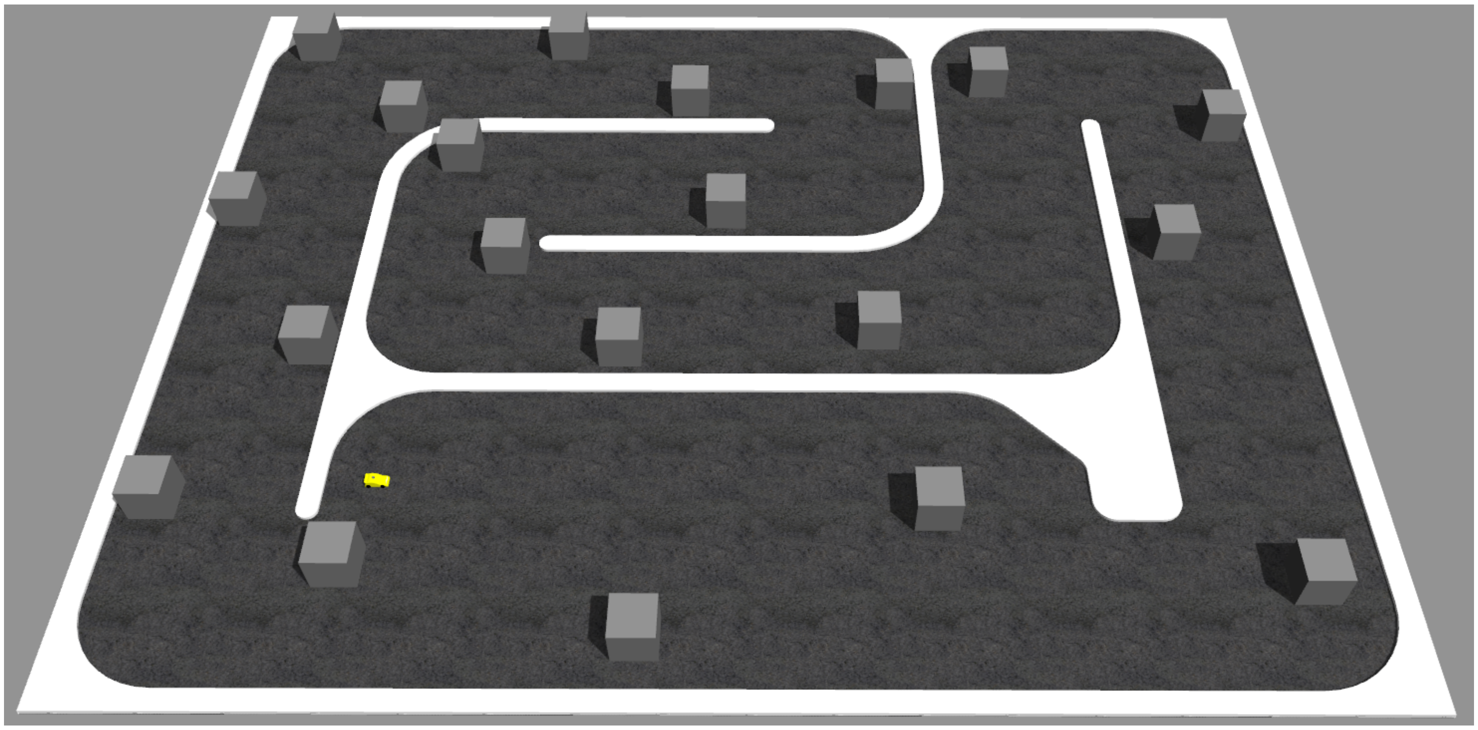
After a sucessful `catkin_make` copy the race track files into the gazebo folder:

```
cp -r catkin_ws/src/simulator/world/race_track .gazebo/models/
```

[Step 4] Launch simulator using

```
roslaunch f1tenth-sim simulator.master world_name:=race_track_obstacles
```

You should see the following track:



[Step 5] Control the racecar using:

```
roslaunch f1tenth-sim racecar.access listen_offboard:=true
```

when the `listen_offboard` arg is set to true, the car will accept commands from a ROS node.

Just like in the previous (Wall Following) assignment, you will need to publish speed and steering values

(AckermannDrive messages) to the `\car_1\offboard\command` topic.

No template code is provided for this assignment.

Here are some suggestions for subroutines and callbacks that you may want to implement:

Subscribe to the LaserScan from the car
For the callback function:

- Prune the ranges array to angles corresponding to the forward looking (say 180 degrees or smaller) field of view only.
- 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.
- 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.
- Calcuate 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 angle
