F1tenth Racing

This class activity is to be done as individuals, not with partners nor with teams.

How to get started →

Begin by reading this entire writeup and making sure you have a good understanding of it. Next, spend some time planning how you’re going to approach the problem.

Introduction

In this class activity, you will race on a large racetrack centerline using an approach of your choice.

Activity Information

Objectives

- Learn to create and tune racing controllers

Resources

- Simulator Setup
- Bicyle Model
- Pure Pursuit
- Ackermann Message
- Bool Message

What to submit

You must submit your repo in the follow file structure

```
├──f1tenth_racing
    ├── config
        ├── params.yaml
    ├── launch
        ├── race.launch
    ├── csv
        ├── gp_centerline.csv
    ├──CMakeLists.txt
    ├──package.xml
    # Any additional folders/scripts
```

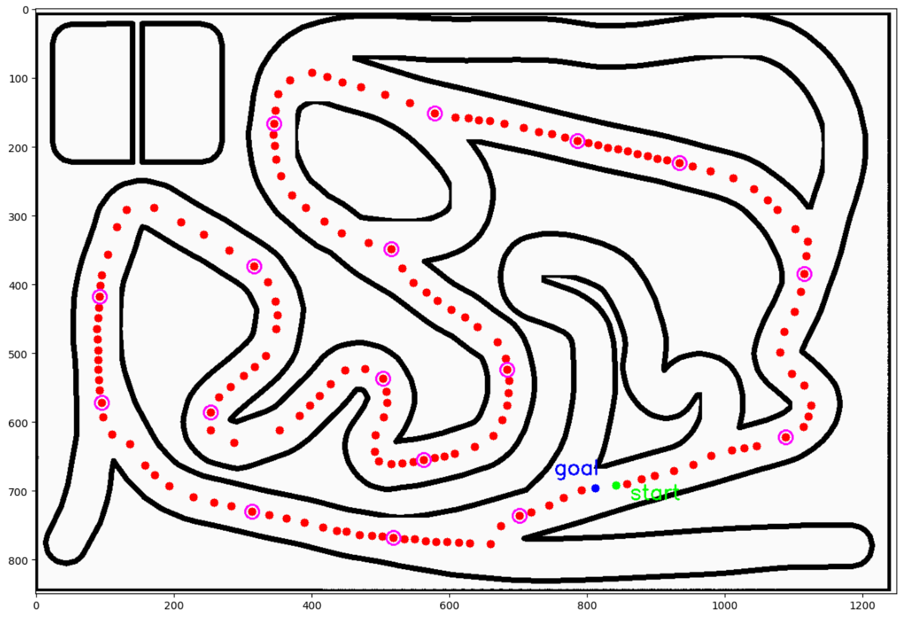
For this activity you can create your own set of scripts, configs. The only requirement is that `race.launch` must launch all the necessary nodes

Grading considerations

- **Late submissions:** Carefully review the course policies on submission and late assignments. Verify before the deadline that you have submitted the correct version.
- **Environment, names, and types:** You are required to adhere to the names and types of the functions and modules specified in the release code. Otherwise, your solution will receive minimal credit.

Racing

For this activity you are provided with a csv detailing the centerline of the RAFall23GP environment called `gp_centerline.csv` . Your goal is to create and run a controller capable of completing the entire set of waypoints in the least amount of time.



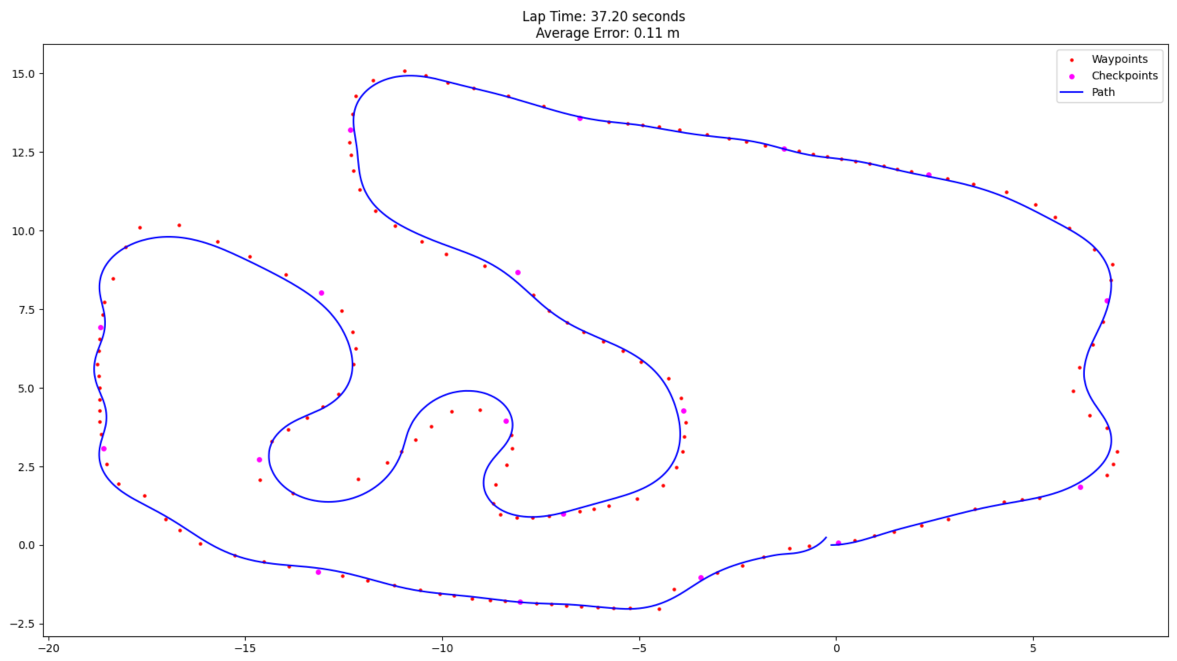
In the image above, the red dots indicate waypoints that you will navigate. The actual number of waypoints you have are 1757. Circles with a magenta ring indicate checkpoints. In the csv, a checkpoint is placed every 100 waypoints. **You are required to pass through all checkpoints. Skipping any checkpoint will be considered as an invalid submission.**

Your racer must listen to a topic called `/race_start` which is of the type `std_msgs/Bool` . An external script will be used to publish a `true` on this topic. When this turns true, you may start your racing. Starting pre-maturely will cause reduction in grading and non-consideration in the final grand prix.

The `params.yaml` also contains the `odom_topic` and `command_topic` . **To allow for multiple racers to run simultaneously, you will need to use these parameters in your node.**

Part 1 - Empty world

First you may test your controller on the empty world. As there are no obstacles you are free to control as necessary. The only requirement is to go close (<1m) to each checkpoint



The above picture can be used as a baseline for comparison. A successful run must

- Have lap time less than 1 minute
- Have an average distance error to a checkpoint of < 0.5m

Part 2 - Walled world

You will now need to tune your controller and account for the walls. Test your controller in the RAFall23GP environment and see your laptimes. If number of collisions > 5 it will be considered as invalid.

Video

Demonstration: https://youtu.be/ZVCemDSc7CI

Submission and Assessment

Submit your code using the Github upload feature on autolab.

Note: Make sure your code complies to all instructions, especially the naming conventions. Failure to comply will result in zero credit.

You will be graded on the following. Penalties are listed under each point, absolute values, w.r.t activity total.

1. Empty World (70%)
2. Walled World (30%)