ECE484 FA24 MP2 Walkthrough

Vehicle Model and Control

Will Chen, John Pohovey, Daniel Zhuang, Eric Ji, Guang Yin

9/20/2024

Outline

- **Logistics**

  HW2 (Individual Work)

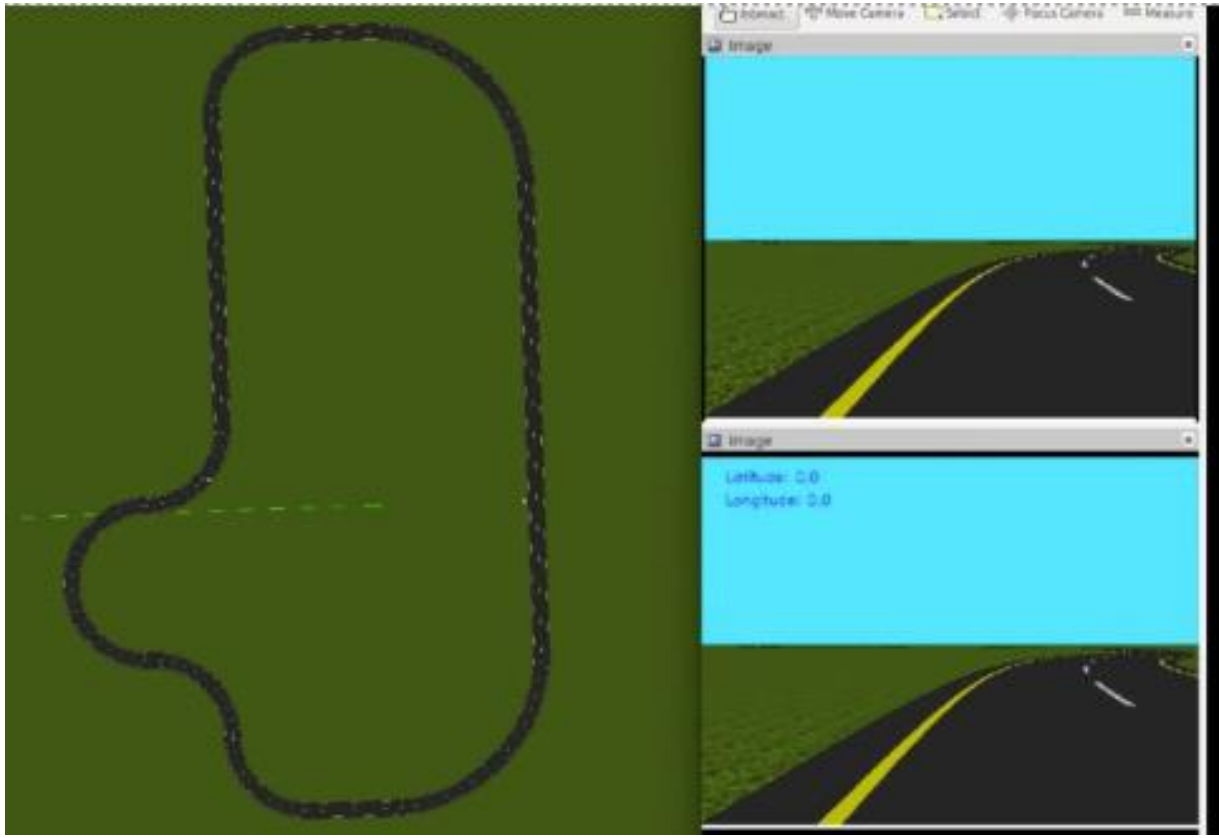
  MP2 (Group Work)
  - Both due on 10/4

  MP1 and HW1 will be due in one week (9/27)
  - Must show up for the demo, otherwise get 0 points
  - Similar to the MP0 demo, there will be demo and individual questions

MP2

Overall Objective: Implement a controller to navigate your car around the track(a list of waypoints to follow).

Task 1: Learn ROS
Task 2: Longitudinal Control (Set Velocity)
Task 3: Lateral Control (Set Heading)



Task 1: ROS

- Recall *rosnode* and *rostopic*, *subscriber* and *publisher*
- ROS also has its own message type, E.g. *ModelState*
- You need to read the documentation to learn how to extract info(E.g. position) from those ROS message

File: **gazebo_msgs/ModelState.msg**

Raw Message Definition

```
# Set Gazebo Model pose and twist
string model_name           # model to set state (pose and twist)
geometry_msgs/Pose pose     # desired pose in reference frame
geometry_msgs/Twist twist   # desired twist in reference frame
string reference_frame      # set pose/twist relative to the frame of this entity (Body/Model)
                            # leave empty or "world" or "map" defaults to world-frame
```

Compact Message Definition

```
string model_name
geometry_msgs/Pose pose
geometry_msgs/Twist twist
string reference_frame
```

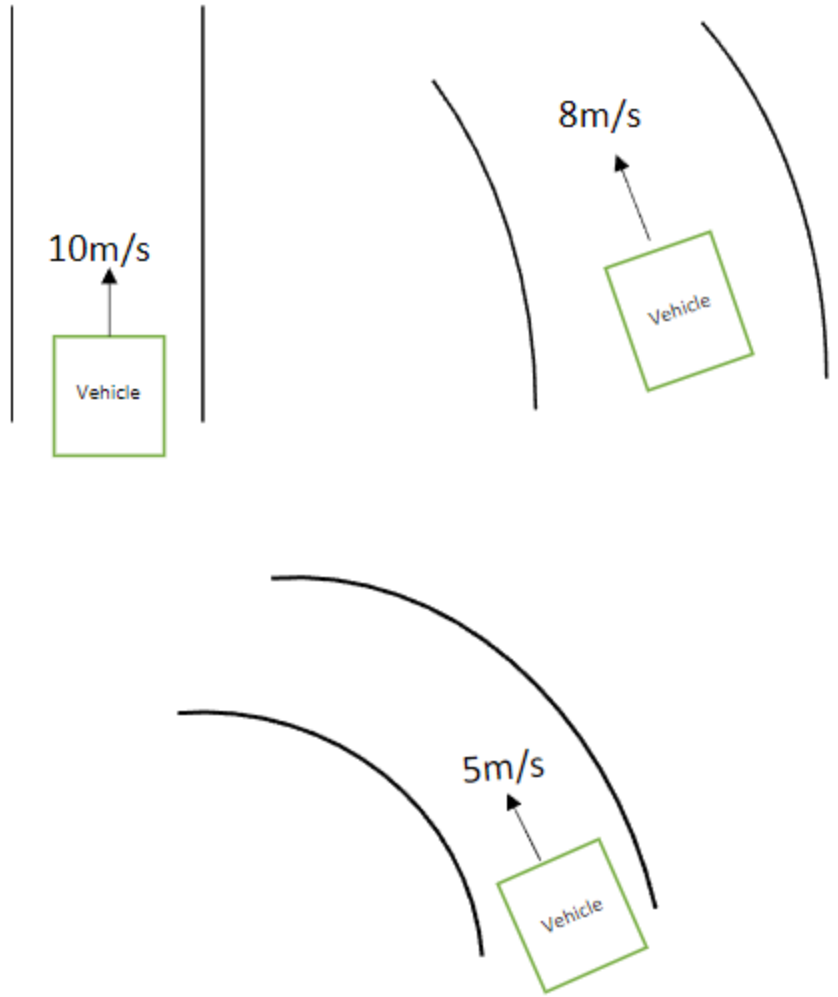
Task 1: ROS

Recall some useful ros commands:

- *rosnode list*: Displays all active nodes
- *rosnode info <node_name>*: Show details about a specific node, including its publisher, subscribers, and services
- *rostopic list*: Provide details about a particular topic, including its publisher, and subscriber nodes
- *rostopic echo <topic_name>*: Display the data being published to a given topic in real-time

Task 2: Longitudinal Controller

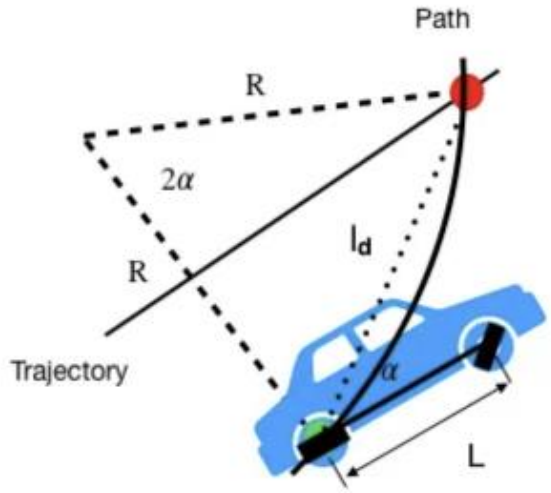
- Based on a list of future waypoints, return the corresponding target speed.
- You need to design the <state, curvature, target_speed>mapping.
- speed = f(curvature, vehicle_state), f is a function up to you to design, e.g. could be piece-wise linear
- For example, drive fast during straights and slow during turns



Task 3: Lateral Controller (Pure Pursuit)

- Derivation of the underlying mathematics is not a requirement for this MP, but strongly recommended for exams and projects.
- In short, you need to implement the below formula to get the steering value

$$\delta = \arctan\left(\frac{2L\sin(\alpha)}{l_d}\right)$$



Several Methods to find the lookahead Target Point

(check the documentation for details)

- Directly setting as the nearest waypoint
- Setting a fixed lookahead distance and interpolating between provided waypoints
- Setting a dynamic lookahead distance (as a function of vehicle states) and interpolating between provided waypoints

Metric

Safety

- You have 4 simulation seconds to reach each waypoint sequentially, otherwise will be considered deviating from the track

Efficiency

- While maintaining safety, your total simulation Time to Finish(TTF) should not exceed 130 seconds

Comfort

- While maintaining safety and efficiency, we also do not want the passengers to experience uncomfortable, therefore we want to avoid sudden acceleration and deceleration.
- A flag in controller.py to enable/disable acceleration logging, provided you implement Task1 correctly

Grading

HW2 (Individual Work) total 100pts

- Problem1. Simple Dubins car model (Nonlinear Dynamics) 30pts
- Problem2. Stability Analysis (Hurwitz Matrix, Eigenvalue Analysis) 30pts
- Problem3. Feedback Control 40pts

MP2 (Group Work) total 100pts

- Report: 70pts
- Autograder: 20pts
- Demo: 10pts

Refer to the documentation for more details

The Grainger College of Engineering

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN