ECE 498SMA: Principles of safe autonomy	Fall 2020
Written by: Yangge Li, Minghao Jiang, Sayan Mitra	*MP5: Autonomous Pipeline*
Website 	*Due Date: 2020/12/13*

1 Introduction

In this MP, you will need to combine the knowledge you gained from the previous MPs to implement an Autonomous agent to run on the racing track and compete with other vehicles.

For this MP, there's no theory part. You will only need to work on the programming part. You will be provided with a basic autonomous agent pipeline that contains perception, decision and planning and low-level control modules and you will need to implement the modules so that your Autonomous agent can perform waypoint/lane tracking on a race track while avoiding obstacles and other vehicles. The score you get will be determined by how fast you can finish the track while keeping the vehicle safe.

For this MP, your group will need to submit a single file `MP5_groupnumber.pdf` with the solution to Problem 1-2. Name all the group members and cite any external resources you may have used in your solutions. More details for submission are given in Section 3. All the regulations for academic integrity and plagiarism spelled out in the student code apply.

This is an open-ended and exploratory assignment. You can finish with little work, but digging deeper will reveal many gems

Learning objectives

- Perception
- Decision
- Control

System requirements

- Ubuntu 16
- ROS Kinetic

2 Implementing The Vehicle in The Environment

In this part of the MP, you will need to implement an autonomous agent to run in the Gazebo environment. This environment, as shown in figure 1, is very similar to the race track you see in MP1 and MP2; the difference is that this race track contains various static and dynamic obstacles. Similar to previous MPs, you will continue to work on the GEM car. Your goal this time is to apply the knowledge and concepts learned in this course to build an autonomous-driving pipeline to the GEM car.

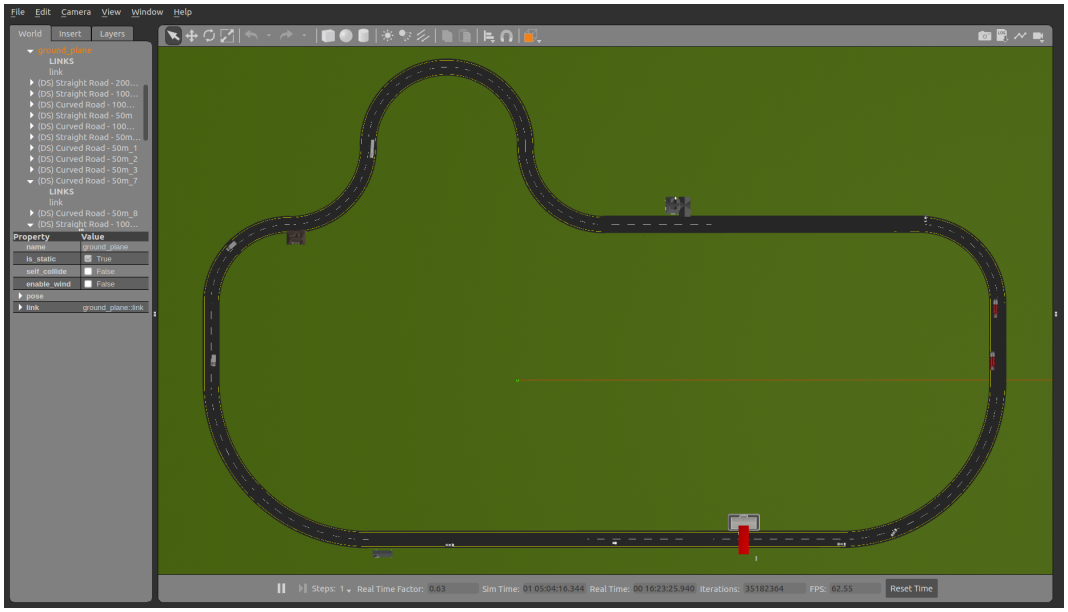

Figure 1: An example environment for this MP.

2.1 Module architecture

This section describes components that are important for this MP. The provided code constructs a basic pipeline including perception, decision and planning and low-level control to drive the GEM vehicle. Feel free to modify any parts listed below increase the performance of your autonomous driving pipeline.

All the files of basic pipeline are located at ./src/mp5/src. **Feel free to change any parts of the pipeline**

2.1.1 VehiclePerception

This module is located in ./perception/perception.py. As its name implies, this is the perception module for the GEM vehicle. The perception module collects and processes the data from sensors on the GEM vehicle. Currently, the available sensors are a front facing camera, LiDAR, IMU, and GPS. In the current provided perception module provides the raw image output from the camera. In addition, the perception module will check LiDAR reading in the box $x \in [0, 20]$, $y \in [-3, 3]$, $z \in [-1.6, 0.5]$ with respect to vehicle frame in front of the vehicle and calculate the average position of the LiDAR point clouds in that box. The position is then projected to the x-y plane by dropping the z component and the distance between vehicle and the position is computed. The GPS readings can provide the current state (pose and twist) of the vehicle.

ROS topics for sensors on GEM car:

```
LiDAR: /velodyne/points
Camera: /gem/front_single_camera/front_single_camera/image_raw
GPS*: /gazebo/model_states
IMU: /gem/imu
```

* : $/gazebo/model\_states$ topic will provide position, orientation and velocity for all models in Gazebo. Refer to provided code for how to use this topic.

Please use command **rostopic info** $/topic\_name$ to view the message type of each topic and refer to ros.org for how to utilize them.

2.1.2 VehicleDecision

This module is located in ./decision/decision.py. This module will contain the decision module of the GEM vehicle. It should take result from the perception module and decide what the car should do next. Currently, the decision module holds a list of waypoints. It will take GPS and LiDAR readings from the perception module and decide which waypoint should be feed to the vehicle. If anything is detected from the Lidar with in range 15, it will send command to stop the vehicle.

This module provides core functionality to your autonomous driving pipeline. You may apply knowledge learned in lectures and previous MPs here or explore other techniques.

2.1.3 VehicleControl

This module is loated in ./controller/controller.py. This module contains a low level controller to control the vehicle motion. The provided controller is similar to the controller you implemented in MP2. It takes the reference from the decision module and the current state of the vehicle and compute required speed and velocity to run the vehicle.

2.1.4 generate_waypoint.py

This is a utility function to compute lane waypoints. You will need to provide the filename to the world file you want to generate the waypoints. The generated waypoints will locate at the center of the track, basically along the white dashed lane marking.

2.2 Development instructions

For this MP, there's no gold solution. You will need to develop your own algorithm to solve the problem of running on the track safely. The algorithm you developed should be general enough that it can handle variation of scenario, for example, different position of static obstacles, different position and moving pattern of dynamic obstacles, different shape of tracks with waypoints.

2.2.1 Editing world in Gazebo

You can edit the world using Gazebo GUI to generate different scenarios to test the algorithm you implemented. You can use the following buttons as shown in figure 2 to move or rotate a selected model. You can also copy and paste models to add more obstecles or create more complicated trajectories. You can save the model in ./src/gem_simulator/gem_gazebo/worlds for future use and you need to modify the launch file to point to the proper world. Note that when saving the model, you may need to consider deleting the vehicle model.

Figure 2: The buttons in the red box from left to right are selection mode, translation mode and rotation mode.

2.2.2 Performance Metric

As you already knew, there will be a race by the end of this semester. The ranking is based on a scoring metric evaluating how far, how fast, and how safe your GEM car performs. The final score is calculated by:

$$S_{Final} = S_k = (\sum_{i=0}^{k} \bar{v}_i) - \Psi_{safe}$$

where kth node is the last node your GEM car reached.

$\bar{v}_i$ is the average speed from $i-1$th node to ith node, which can be calculated as:

$$\bar{v}_i = \frac{x_i - x_{i-1}}{t_i - t_{i-1}} \tag{1}$$

Safety score Ψ_{safe} is determined by 2 factors: how many obstacles GEM car hit and how long GEM car deviates from the track. For each obstacle your GEM car hit, your points will be deducted by 100 points; for each second your GEM car deviates from the race track, your points will be deducted by 50 points (up to 200).

$$\begin{aligned} \Psi_{safe} &= \Psi_{obs} + \Psi_{deviates} \\ &= 100 * N_{obstacleHit} + min(200, 50 * t_{deviates}) \end{aligned} \tag{2}$$

Scoring metric interpretation: The score is calculated based on the distance traveled by and speed of your GEM car. The more waypoints GEM car reaches and the faster GEM car drives, the higher your score would be. Meanwhile, you have to make sure your GEM car drive safely. If it hits obstacles or deviates from the race track, your score will be negatively impacted.

2.3 Report

Problem 1 (25 points)**.** Describe the algorithm you are using for each module. You need to describe each component of your design potentially with pseudo code or diagrams. Note, answer of this question should not exceed 2 pages.

Problem 2 (25 points)**.** Record a video of your GEM car driving on the race track and include the link to the video in the report.

2.4 Race

After the deadline of this MP, there will be a in-class race among all the groups. The race rule is that we will run GEM car with each group's solution on the race track (not exactly the same as provided in MP-release). After the GEM car finishes one loop, a score will be generated based on the same scoring metric in 3.2.2. The ranking will be based on this score, and extra credit will be assigned.

- The Champion will receive 25 points of extra credit
- 2nd place will have 20
- 3rd place will have 15
- 4th place will have 10
- 5th place will have 5

In addition to that, any teams that comes up with an solution that can drive the GEM car to finish the entire loop safely(meaning $\Psi_{safe} = 0$) can receive another 25 points of extra credit.

Details of how the race is conducted will be released on Piazza later.

3 Report and Submission

For problem 1-2, each group should write a report that contains the solutions, plots, and discussions. This report should be submitted to Gradescope **per group** to assignment **MP5** with filename `MP5_report_#.pdf`. Please note that for this MP, it is not required to submit code unless you would like to participate in the race. If you would like to, please submit the code to Compass2g. The content in folder src/mp5/src should be submitted to Compass2g **pergroup** to assignment **MP5 code** in a zip file with filename `MP5_code_#.zip`.