EE405A Integrate Your Control System and F1tenth Project

This project is based on the ROS-based Control system and gazebo-based Eurecar F1TENTH project.

- Week5 Simple Vehicle Controller https://github.com/hynkis/EE405A/tree/main/Week5/Materials/waypoint_follower (https://github.com/hynkis/EE405A/tree/main/Week5/Materials/waypoint_follower)
- Week6 Gazebo-based Simulator https://github.com/Leedk3/EE405_a_eurecar_f1_tenth_project (https://github.com/Leedk3/EE405_a_eurecar_f1_tenth_project)
- Week7 (This week) Waypoint Recorder https://github.com/hynkis/EE405A/tree/main/Week7/Materials/waypoint_recorder (https://github.com/hynkis/EE405A/tree/main/Week7/Materials/waypoint_recorder)

You will be able to integrate your controller package with the gazebo-based simulator.

We will introduce the following topics

1. Record waypoint trajectory in simulator
2. Integrate your controller with simulator

1. Record waypoint trajectory in simulator

Package for recording waypoint trajectory in the gazebo simulator.

Requirements

1. For python packages

```
pip2 install numpy --user
pip2 install pandas --user
pip2 install roslib --user
pip2 install matplotlib --user
```

2. For ROS Melodic messages

```
sudo apt-get install ros-melodic-geometry-msgs
sudo apt-get install ros-melodic-visualization-msgs
sudo apt-get install ros-melodic-nav-msgs
```

3. For Gazebo

```
sudo apt-get install ros-melodic-gazebo-ros-pkgs
sudo apt-get install ros-melodic-gazebo-ros-control
```

Record waypoint trajectory (waypoint_recorder/recorder.py)

Subscribe the current position of ego vehicle and save the position every specific distance. Simply run package and save waypoint in real-time. For stop recording, turn off this package using `ctrl + c`

You can change these parameters

- WPT_CSV_PATH : the path for saving your waypoint trajectory
- WPTS_GAP : the gap distance between each waypoint (Waypoint trajectory resolution)

```
rosrun waypoint_recorder recorder.py
```

For manual control using your keyboard inputs, run the keyboard node in the gazebo simulator package: https://github.com/Leedk3/EE405_a_eurecar_f1_tenth_project/blob/master/f1tenth-sim/scripts/keyboard_teleop.py (https://github.com/Leedk3/EE405_a_eurecar_f1_tenth_project/blob/master/f1tenth-sim/scripts/keyboard_teleop.py)

```
rosrun f1tenth-sim keyboard_teleop.py car_1
```

Click the terminal where the keyboard_teleop node is running, and press your keyboards to control your vehicle.

- w a d s : forward, left, right, backward
- space_bar : brack

After recording the trajectory, you can plot the recorded data using Matplotlib.

You may need to edit and smooth your recorded trajectory by your hand, spline interpolation, etc.

Change the path for saved waypoint trajectory.

- FILENAME : the name of the saved waypoint trajectory in the directory 'waypoint_recorder/wpt_data'

```
roscd waypoint_recorder
cd scripts
python2 wpt_plotter.py
```

2. Integrate your controller with simulator

Visualize pre-built waypoint trajectory in Rviz (waypoint_follower/wpt_loader.py)

Publish the recorded waypoint.

You need to change the waypoint path parameter.

- WPT_CSV_PATH : the path for your saved waypoint trajectory

```
rosrun waypoint_follower wpt_loader.py
```

Matching the topic message

In the Gazebo-based simulator, the vehicle's longitudinal behavior is controlled by speed command, not acceleration command.

Also for control, the simulator uses the message type as AckermannDrive, not AckermannDriveStamped.

Therefore, for integrating your controller with the simulator, you need to modify some codes in 'waypoint_follower/controller.py' as follows:

from

```
from ackermann_msgs.msg import AckermannDriveStamped
~~~

class WaypointFollower():
    ~~~

    def publish_command(self, steer, accel):
        """
        Publish command as AckermannDriveStamped
        ref: http://docs.ros.org/en/jade/api/ackermann_msgs/html/msg/AckermannDriveSta
mped.html
        """
        msg = AckermannDriveStamped()
        msg.drive.steering_angle = steer / np.deg2rad(17.75)
        msg.drive.acceleration = accel
        self.pub_command.publish(msg)
~~~

self.pub_command = rospy.Publisher('/control', AckermannDriveStamped, queue_size=5)
self.sub_odom    = rospy.Subscriber('/simulation/bodyOdom', Odometry, self.callback_od
om)
```

to

```
# from ackermann_msgs.msg import AckermannDriveStamped
from ackermann_msgs.msg import AckermannDrive
~~~

class WaypointFollower():
    ~~~

    def publish_command(self, steer, accel):
        """
        Publish command as AckermannDriveStamped
        ref: http://docs.ros.org/en/jade/api/ackermann_msgs/html/msg/AckermannDriveSta
mped.html
        """
        # msg = AckermannDriveStamped()
        # msg.drive.steering_angle = steer / np.deg2rad(17.75)
        # msg.drive.acceleration = accel
        msg = AckermannDrive()
        msg.steering_angle = steer / np.deg2rad(17.75)

        # You may need to dynamically change this speed command for optimize your lap
 time.
        # Just a large speed command is not a proper solution for good lap time due to
unstable control.
        msg.speed = 0.5
        self.pub_command.publish(msg)
~~~

# self.pub_command = rospy.Publisher('/control', AckermannDriveStamped, queue_size=5)
# self.sub_odom    = rospy.Subscriber('/simulation/bodyOdom', Odometry, self.callback_
odom)
self.pub_command = rospy.Publisher('/car_1/command', AckermannDrive, queue_size=5)
self.sub_odom    = rospy.Subscriber('/car_1/ground_truth', Odometry, self.callback_odo
m)
```

Running the Controller (waypoint_follower/controller.py)

Before running the controller node, change the waypoint path to your recorded one.

- WPT_CSV_PATH : the path for your saved waypoint trajectory

Open a terminal and run the following node.

```
rosrun waypoint_follower controller.py
```