System Integration (Capstone)

Write up

1st submit: September 19th, Kenta Kumazaki

1. Goals

In this project my goal is to enable Carla to drive around the test track using waypoint navigation.

I implement components of the Perception, Planning and Control subsystems.

The vehicle should run the test track according to the waypoints and stops at red traffic lights.

2. Steps

The steps of this project are the following. I followed the walkthrough lessons.

(1) Waypoint Updater Node (partial)

At first, I subscribe only "/base_waypoints" and "current_pose" and check the whether Waypoint Updater can publish "/final_waypoints".

(2) DBW (Driver By Wire) Node

Implement the controller and check whether Carla can run according to the waypoints.

(3) Traffic Light Detection

(a) Detect the traffic light and its color from "/image_color".

(b) Convert traffic light position to a waypoint index and publish "/traffic_waypoint".

(4) Waypoint Updater Node (full)

Use "traffic_waypoint" to change the waypoint target velocities, then publish "/final_waypoint" to stop Carla if the traffic light color is red.

(5) Change points

(a) Number of waypoints to be published

(b) PID gains

(c) Deceleration profile

3. Submission

(1) GitHub

https://github.com/kkumazaki/Self-Driving-Car_Project9_Capstone[

(2) Directory

I cloned the basic repository from Udacity https://github.com/udacity/CarND-Capstone

and modified the following files.

- **Writeup_of_Lesson18.pdf**: This file
- **/ros/src/waypoint_updater/**
  - waypoint_updater.py: Updates the target velocity property of each waypoint based on traffic light.
  - waypoint_loader.py: Generates and publishes the static waypoint data.
- **/ros/src/twist_controller/**
  - dbw_node.py: Publishes throttle, steering, brake commands to control the vehicle.
  - twist_controller.py: Generate target values using PID controller, low pass filter, etc.
  - pid.py: PID controller. (I only modified this module to add loggings)

4. Reflection

(5) Change points

(a) Number of waypoints to be published

I suppose there's a latency issue because Carla couldn't run according to the default setting.

When I changed the **LOOKAHEAD_WPS** from 200 to **50**, Carla was able to follow the waypoints.

```
26  #LOOKAHEAD_WPS = 200 # Number of waypoints we will publish. You can change this number
27  LOOKAHEAD_WPS = 50 # Number of waypoints we will publish. You can change this number
28  MAX_DECEL = 0.5 # Max deceleration
```

(b) PID gains

I tuned PID gain as following:

P gain: 0.3 (recommended in the lesson)

I gain: 0.3 (originally it was 0.1, but there was about 0.4~0.5 integral error)

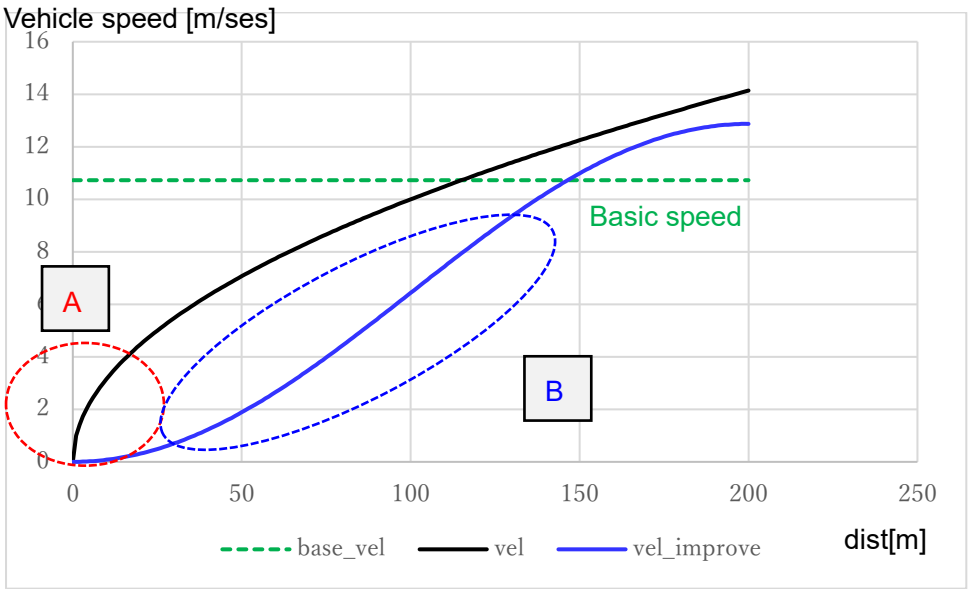
D gain: 0.1 (originally it was 0, but there was less than 0.1 differential error)

(c) Deceleration profile

The deceleration was kind of quick when it stopped, so I changed it as below:

dist [m] is the distance from the red light.

The original vehicle speed change was big around stop timing **(A),** so I made it smoother as the following codes **(B).**



waypoint_update.py (B)

```
115             # Smoothing the deceleration profile
116             #vel = math.sqrt(2 * MAX_DECEL * dist) # This can be improved by smoothing the deceleration profile.
117             if dist < 200:
118                 vel = wp.twist.twist.linear.x * 0.6 * (1.0+math.sin(math.pi*(dist/200. - 0.5)))
119             else:
120                 vel = wp.twist.twist.linear.x * 1.2
```

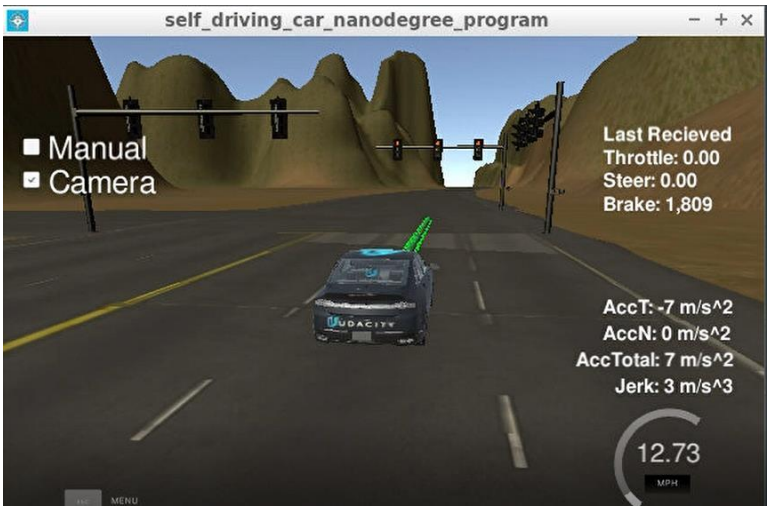
5.Result

The final code was able to achieve the Project Rubric as below.

- The code is built successfully and connects to the simulator.
- Waypoints are published to plan Carla's route around the track.
- Controller commands are published to operate Carla's throttle, brake, and steering.
- Successfully navigate the full track more than once.

Carla started to run after the traffic light became from red to green.

When Carla reached the red light, it started to decelerate and stop in front of the intersection.



I think if I was able to set bigger LOOKAHEAD_WPS, Carla was able to stop earlier than this.