System Integration of Autonomous Vehicle

yu-shen

November 28, 2017

Contents

1 Introduction **1**

2 Plan and Schedule **1**

3 Questions/Problems **2**

4 Major nodes or modules **2**

4.1 `./waypoint_tracker.org` 2

4.2 Waypoint_Updater 2

4.3 DBW_Node 2

4.4 `./tl_detector.org` 2

4.5 `./tl-classification.org` 2

5 Overall Architecture **2**

6 TODO TODO **2**

1 Introduction

The document uses literate programming approach to present the design and implementation. It presents software construction as prose of writing natural language narrative. The key distinction is that all text are comments to code, while code are presented in special code block. For details of it, please check the link for it. With this style of narrative, key chunks of code relevant to higher level design are presented with much more thought process presented. This would make the source code much more understandable, removing the challenges of providing updated and appropriate comments to the source code.

In this document, `«to_local_coordinates»` standards for the named source code block to be inserted in the place.

In this design document, "my_car" refers to the car being controlled by the path planner.

The full source code can be generated from this literate programming document:

If you prefer to read the source code, you can still read them, but there may not be enough comments.

2 Plan and Schedule

Outdated

Tas	Owner	Due	Status	Remarks
Waypoint_Updater (initial)	Yu	11/1	Verified 10/31	Learning curve of ROS
Traffic Light Detection & Classification	Reddy & Sahil	11/15	Start 11/6	
DBW to issue control	Eric? & ?	11/5		
Waypoint_Updater (deceleration/acceleration	Patrick & Yu	11/20		
Integration	Yu & ?	11/25		
Verification and tuning	? & all	11/30		

This is a rather rough schedule. It shows that it's already very tight for 11/30 to complete.

For Waypoint_Updater, the remaining work is to subscribe to traffic light waypoints, and decide appropriate deceleration at the red/yellow light and subsequent acceleration after getting green light.

The traffic light detection should pubilsh message of traffic light (being red, greee, maybe yellow).

For DBW_node, the key remaining piece is TwistController, producing the appropriate throttle, steering, brake values based on the current speed, the desired speed, and dbw_enabled status. For the TwistController, there is a PID implementation, but the parameters of P, I, D needs to be determined.

3 Questions/Problems

Outdated.

1. In Waypoint_Updater, for output final_points, how to set the speed? Should we set both the linear and angular speed? It seems that only need to set the linear.x, as it's in the my_car's local coordinates, there is only movement on the x-axis. There is no movement on the angular z, not even possible on x, and y. It seems that we should also set the angular velocity on z-axis? but how?
2. How to tune the parameters of PID?
3. Where to use the lowpass filter, for the inputs to DBW_node or its outputs, or both? Maybe, just the current velocity, as it's the only value from the physical world that might have noise.
4. What's the physics formula to express the required brake force considering the mass of the vehicle, its current velocities?

4 Major nodes or modules

The following are the major nodes or modules clicking on the links can visit the corresponding design and construction document.

4.1 `./waypoint_tracker.org`

This is the supper class abstracting the common treatments to waypoints among waypoint_updater and tl_dectector.

It resides in a module called waypoint_lib

To import: ==:

```
from waypoint_lib.waypoint_tracker import WaypointTracker
```

4.2 Waypoint_Updater

4.3 DBW_Node

4.4 `./tl_detector.org`

4.5 `./tl-classification.org`

This is the wrapper to the deep learning classifer.

It expects to access the file of the trained module at path: ./ros/src/trained_model/frozen_inference_graph.pb assuming the current directory is the project directory of the Capstone project.

5 Overall Architecture

The maximum loop frequencies

6 TODO TODO

1.