Capstone Project Report

Code:

https://github.com/russell13192/CarND-Capstone

Team Members

George Murray(team leader): gmurray@fetchrobotics.com

Ashish: bashish.iitg@gmail.com

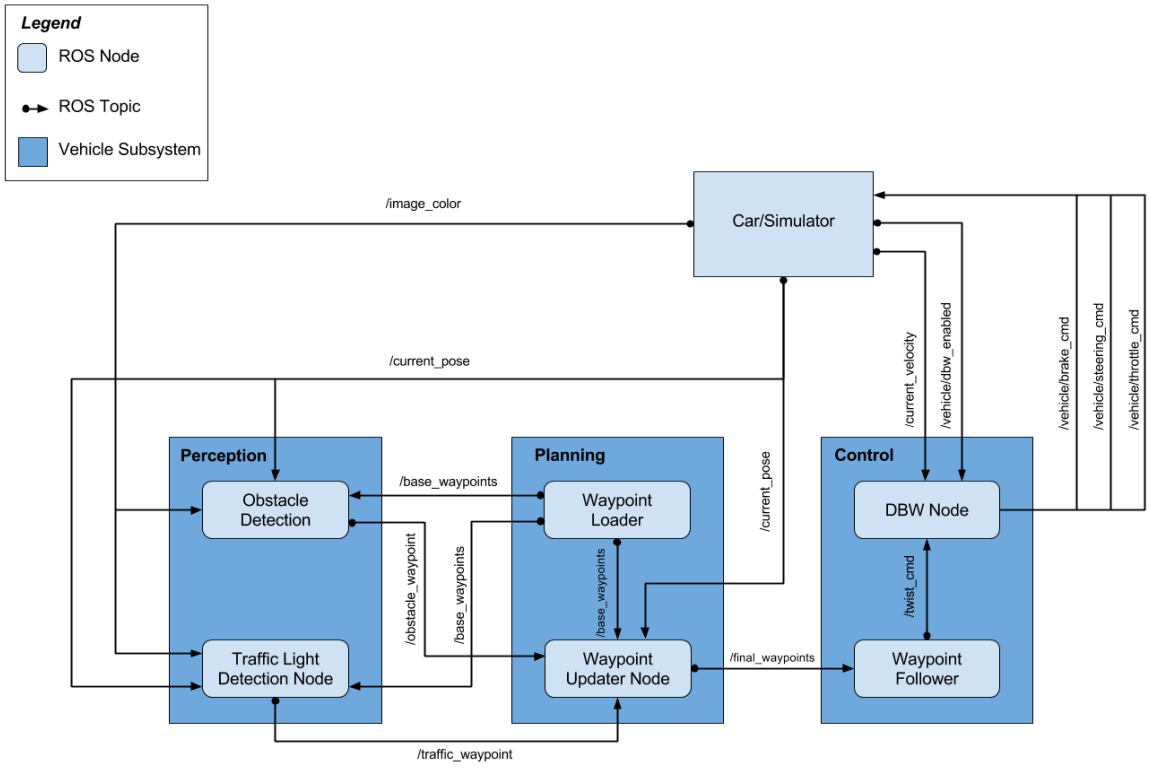
LI Hexuan: hexuan.li@hotmail.com

Keisuke Shima: komaki289@gmail.com

Darrick: dz.aiml2018@gmail.com

Introduction of Project

The following figure shows and autonomous vehicle system architecture diagram based on ROS, including ROS nodes and topics.



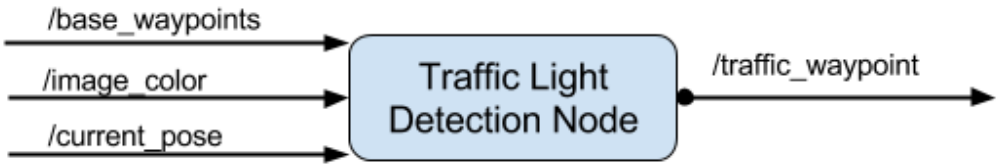
The final project is a group project, students are required to implement a few ROS nodes in the above system with other ROS nodes already given. Below is the brief overview of those nodes needed to be implemented, including traffic light detection, control and waypoint_updator.

Traffic light detection

The traffic light detection node is included in tl_detector.py. This node takes in data from the /image_color, /current_pose, and /base_waypoints topics and publishes the locations to stop for

red traffic lights to the /traffic_waypoint topic. The /current_pose topic provides the vehicle's current position, and /base_waypoints provides a complete list of waypoints the car will be following.

The /current_pose topic provides the vehicle's current position, and /base_waypoints provides a complete list of waypoints the car will be following.



My work is to build both a traffic light detection node and a traffic light classification node. Traffic light detection is take place within tl_detector.py, whereas traffic light classification is within ../tl_detector/light_classification_model/tl_classfier.py.

Traffic light classfication

In this project, the traffic light classification is based on SSD Inception V2 neural network. Thanks to the work done by Alex Lechner, in his post, he walked through all the necessary steps of how to do a transfer learning of SSD classification. The training data is from here. And the network is trained based on Tensorflow-gpu 1.14.0.

The traffic light image in the simulator is quite different than the one from Udacity test lot, A single SSD model is tried and trained on images from both simulator and test lot, giving very poor accuracy. I ended up to train two separate models, one model called frozen_inference_graph.pb based on simulator image, and the other model called frozen_inference_graph_real.pb is based on Udacity real test lot image. The different model is automatically chosen by launching different '.lauch' files:

The following command is to launch model for simulator environment

```
> roslaunch launch/styx.launch
```

And the following command is to launch model to test training bag data

```
roslaunch launch/site.launch
```

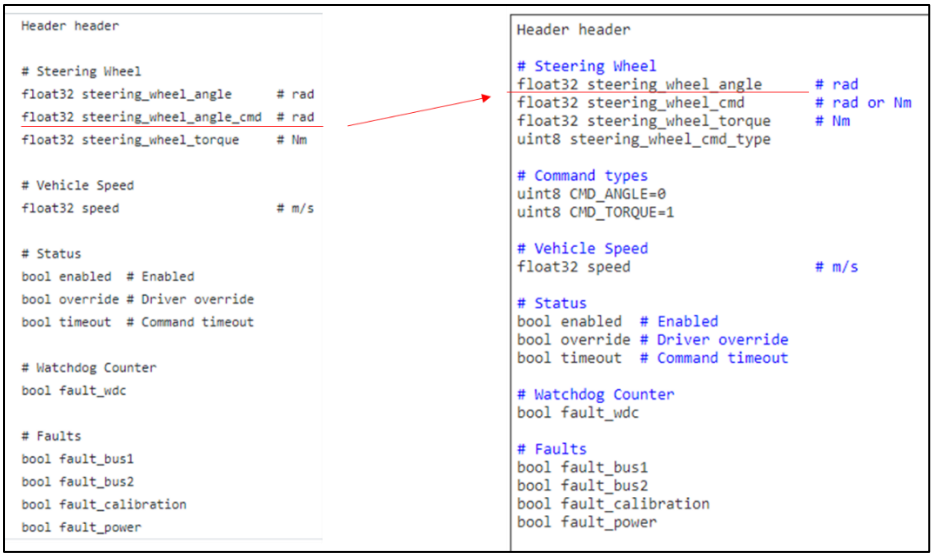
styx.launch and site.launch point to different config files, these config files has a bool field "is_site", if is_site=True, it means the test will done for real image, and tl_classification.py picks frozen_inference_graph_real.pb accordingly.

Issues and fixes

I have my own native installation of ROS environment, and all those dependency packages are installed with latest versions, but Udacity repo dependencies have not been updated, which caused version mismatches and a lot of debug time.

AttributeError: 'SteeringReport' object has no attribute 'steering_wheel_angle_cmd'

This error is due to the fact that dbw_mkz_msgs version in Udacity repo is not updated, while in the latest version of dbw_mkz_msgs, the original name 'steering_wheel_angle_cmd' is changed to 'steering_wheel_angle'



```
Header header

# Steering Wheel
float32 steering_wheel_angle      # rad
float32 steering_wheel_angle_cmd  # rad
float32 steering_wheel_torque     # Nm

# Vehicle Speed
float32 speed                     # m/s

# Status
bool enabled  # Enabled
bool override # Driver override
bool timeout  # Command timeout

# Watchdog Counter
bool fault_wdc

# Faults
bool fault_bus1
bool fault_bus2
bool fault_calibration
bool fault_power
```

```
Header header

# Steering Wheel
float32 steering_wheel_angle      # rad
float32 steering_wheel_cmd        # rad or Nm
float32 steering_wheel_torque     # Nm
uint8 steering_wheel_cmd_type

# Command types
uint8 CMD_ANGLE=0
uint8 CMD_TORQUE=1

# Vehicle Speed
float32 speed                     # m/s

# Status
bool enabled  # Enabled
bool override # Driver override
bool timeout  # Command timeout

# Watchdog Counter
bool fault_wdc

# Faults
bool fault_bus1
bool fault_bus2
bool fault_calibration
bool fault_power
```

There are two potential ways to solve this issue: one is to downgrade dbw_mkz_msgs version, which I tried and system didn't allow me to downgrade somehow; the second solution is to copy old version of dbw_mkz_msgs into ros/src directory, this method worked for me.

in import_graph_def raise ValueError('No op named %s in defined operations.' % node.op) ValueError: No op named NonMaxSuppressionV3 in defined operations.

I got the above error when I use trained SSD neural network at traffic light classifier in the simulator. This error is due to Tensorflow version mismatch between Udacity repo and my local environment. The default tensorflow version is 1.4.0 in Udacity repo, and I trained my neural network based on latest Tensorflow 1.14.0. I tried to downgrade Tensorflow in my local environment to 1.4.0 for training and wasn't successful. So what I did is to upgrade Tensorflow in Udacity repo from 1.4.0 to 1.14.0 for capstone project.

UnboundLocalError: Local variable 'car_wp_idx' referenced before assignment

When I run the whole pipeline on training bag data, I got the above error, the work around is to define car_wp_idx=-1 as initialization