Programming a Real Self-Driving Car

This is the project write for the final project of the Udacity Self-Driving Car Nanodegree: Programming a Real Self-Driving Car from team "Apollo11" For more information about the project, see the project introduction here.

In this project we wrote ROS nodes to implement core functionality of the autonomous vehicle system, including traffic light detection, control, and waypoint following.

Team:

Name	Email
Yuvaram Singh	yuvaramsingh94@gmail.com
Patrick Wu	sd4174263@163.com
Saravanan Moorthyrajan	sarav_m@hotmail.com
Manasa Raghavan	mraghavan@gmail.com

Notes to Reviewer

In this submission we have implemented all the required nodes to run successfully in the simulator. However due to our environment setup issues, we haven't been able to run successfully on the simulator with real time traffic light classification. When real-time traffic light classification is enabled, there is a huge lag in the system and it breaks the processing chain. For the purpose of successfully completing the track on the simulator, we have resorted to use the traffic_light_status available to avoid running the classifier. We have verified that the traffic light detection works successfully by running inference on test images. Due to lack of time to further investigate it and an upcoming deadline for submission for one of our team members, we decided to make this submission and continue to work on a fully working solution. The real-time classification can be enabled by setting simulation to 'False' in tl_detector.py.

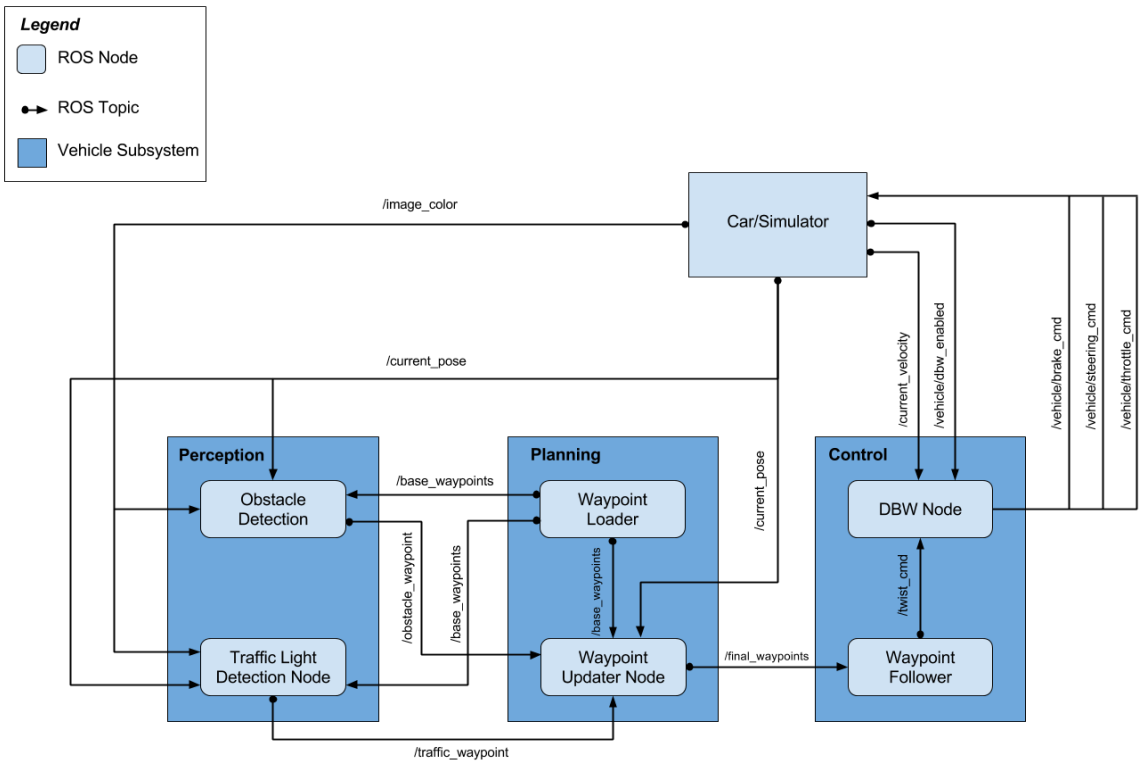
Project Objective

The objectives of the System Integration project are:

- Smoothly follow waypoints in the simulator.
- Respect the target top speed set for the waypoints'
- Stop at traffic lights when needed.
- Stop and restart PID controllers depending on the state of /vehicle/dbw_enabled.
- Publish throttle, steering, and brake commands at 50hz.

System Architecture

The ROS nodes implementing the Perception, Planning and Control sub-systems are shown in the figure below.



Waypoint Updater Node

The Waypoint update node updates the waypoints ahead by calculating the distance and angle to the next waypoint ahead. In case a Red traffic light is detected, it calculates the waypoints up to the stop line.

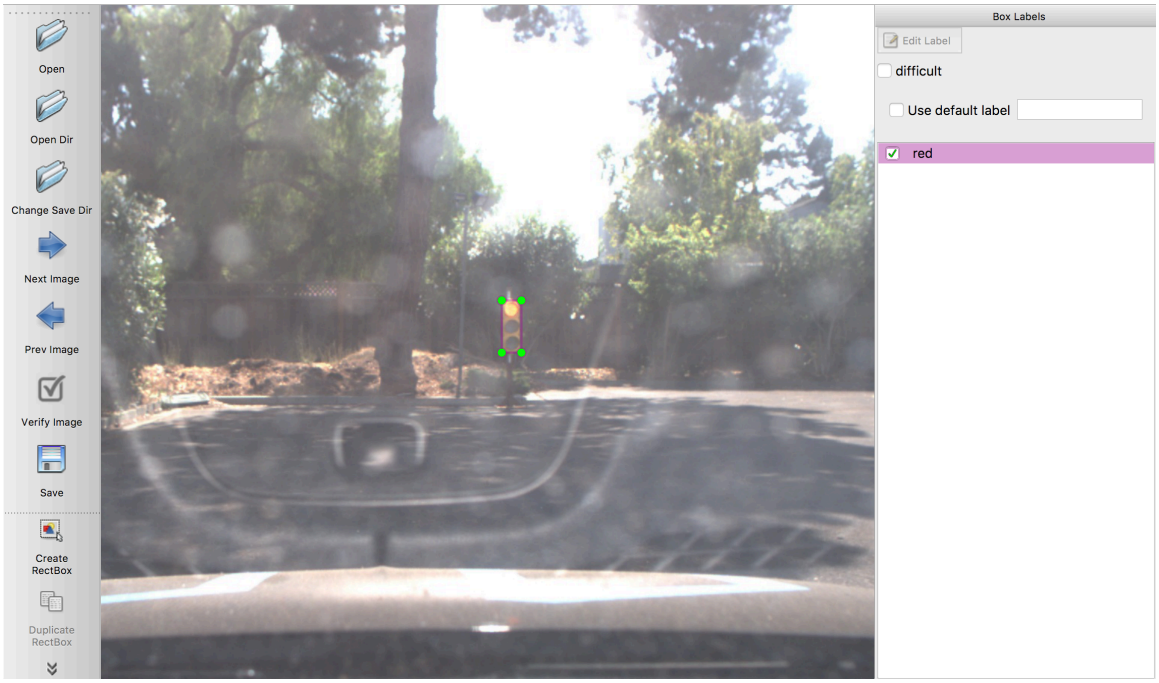
Controller Node

A PID controller is implemented in the DBW node to control the speed, throttle and brake commands out to the car control.

Traffic Light Detector Node

The traffic light detector node detects the traffic light ahead based on the camera image received. The traffic light classifier was developed using TensorFlow Object detection API and following the guidelines provided in this post

The first step was to create a training dataset of labeled images to use for the training the network. We used the training bag provided by Udacity , captured images from the simulator and the Bosch Small traffic lights dataset to train our model. We used the LabelImg tool to annotate the images.



We leveraged the existing deep learning models available for object detection and used transfer learning to train our model for Traffic light classification. We used the Region-Based Fully Convolutional Networks (R-FCN) with Resnet 101

The results for classification:

Traffic light detection for simulator images:


Red: 99%
Red: 99%
Red: 99%
0
100
200
300
400
500
0
100
200
300
400
500
600
700


Yellow: 99%
Yellow: 99%
Yellow: 99%
0
100
200
300
400
500
0
100
200
300
400
500
600
700

Traffic light detection on test site images:


0
200
400
600
800
1000
0
200
400
600
800
1000
1200


0
200
400
600
800
1000
0
200
400
600
800
1000
1200


0
200
400
600
800
1000
0
200
400
600
800
1000
1200