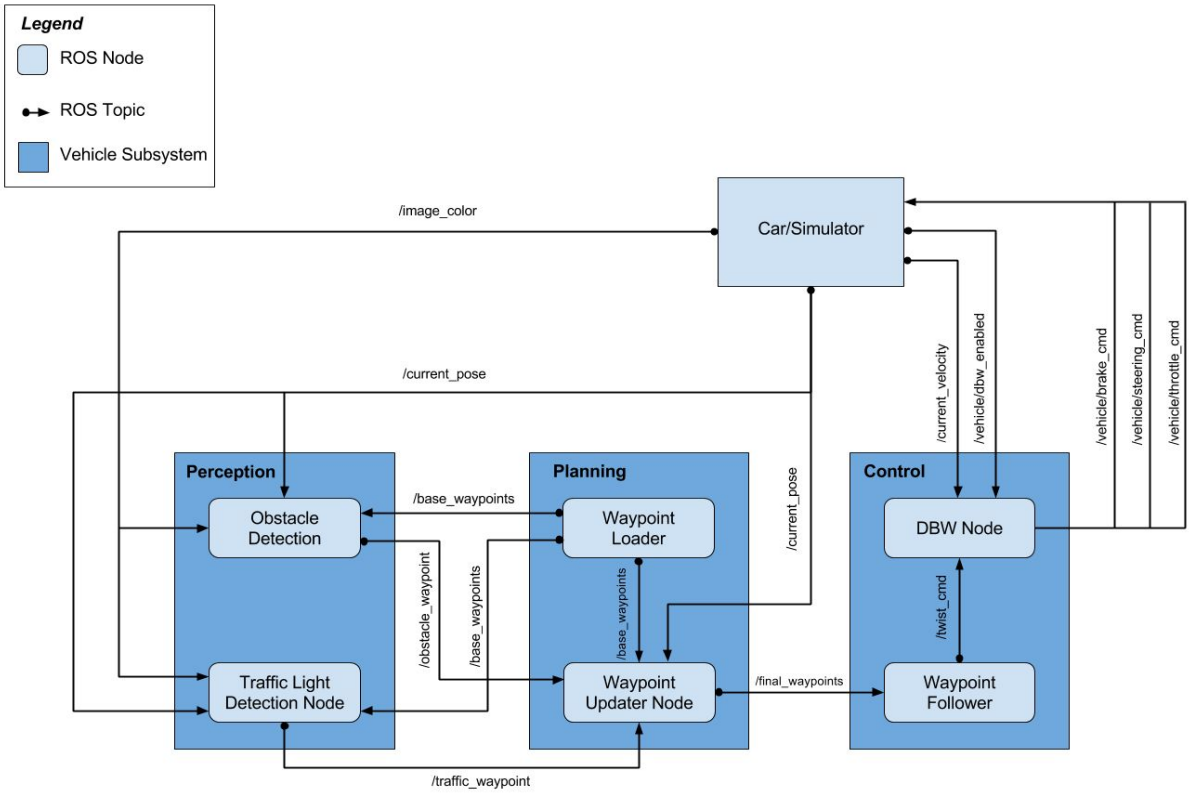
Udacity Self Driving Car - Term 3 - Capstone

Capstone Project

Derek Wu, 2018



Introduction

The purpose of this project is to create a ROS program that can that navigates the car smoothly while detecting traffic lights. The car is programmed to follow the provided waypoints and stopping at traffic lights. In order for the car to perform the requirements, the project is split into the following ROS nodes:

1) **Waypoint Updater** - Provides the car with the waypoint at specified intervals

2) **Traffic Light Classifier** - Use the simulated camera to determine the traffic light status (Red, Yellow, or Green)
3) **Traffic Light Detector** - Provide the car with the appropriate commands according to the traffic light status
4) **DBW Node** - Contains the physics and physical properties of the car. Controls the car when the Waypoint Updater provides the next step.

Credit

Portion of the code is taken from the Q&A section from the project (Capstone Project: Card 6, 8, 10 and 12).

This code uses darkflow by thtreiu which is a tensorflow port of the famous darknet by PJreddie which is the neural network toolbox that powers YOLO. The traffic light detection uses transfer learning and references the pretrained YOLOv2 weights trained on VOC dataset.

During the transfer learning process, the simulated dataset from Udacity's Bosch Traffic Light challenge is used. A custom script was written by me that converts the YAML file provided to the Pascal VOC format that darkflow accepts. I have attached my converter code to this project zip file so that others can use it if they need so in the future.

Requirements to Run

In order to run my Capstone Project correctly, the darkflow library will need to be installed. To facilitate this, follow these steps:

1) Call into the darkflow root directory
2) Make the file **create.sh** an executable script (chmod +x)
3) Run **./create.sh**

Waypoint Updater

This node subscribes to three publishers which includes the current position, the waypoints, and the traffic lights waypoints. The main function of this node is to publish the

waypoints that the car needs to reach within a certain look forward period. In my case, this lookahead period is 200 waypoints.

Traffic Light Classifier

This node subscribes to the simulated camera on the car. The node takes the camera and sends an image to the function TLClassifer which has the underlying machine learning algorithm that determines the status of the traffic. For my project, I chose to use YOLOv2. However, instead of using the original darknet I used the tensorflow port darkflow. I took the simulated data from Udacity's Bosch Traffic Light challenge. I took this data used transfer learning on the original YOLOv2 VOC weights. Originally, I tried using the Tiny YOLOv2 VOC version since this has good performance while utilizing much less resources. However, I was unable to train until the results were satisfactory. Using the tiny version had too many false positives (detects traffic lights when none exist in the view). The final version had an IUC below 0.8. In order to make it work 100% of the time throughout the track, I took advantage of the fact that there is 3 traffic light shown. It takes the highest probability and frequency of the class detected.

Traffic Light Detector

The main function of the node is take the data from the traffic light classifier and act accordingly depending on the traffic light status. If a red light is detected, this node publishes information to the waypoint updater to stop at the appropriate waypoint.

DBW Node

This drive by wire node contains all the physical properties and physics of the car which includes the PID parameters. When the waypoint uploader feeds a waypoint to the system, the car needs to move in way without hurting the occupants or beyond the range that the car can handle or react. This DBW node ties all the fed data into the control system which acts on the simulated actuator and steering that is required.