Team of One

prasoonkgupta@gmail.com

I was late to join any team which really pushed my further down when i attempted to do this project myself. Finally, I am able to complete it. I understand if I don't get any feedback from running this on actual car, as it will not be practically possible to do it for every student.

Project Description

Publish waypoints for some distance ahead of the vehicle. First start by subscribing to topics that publish base waypoints (latched subscriber so we do it just once) and current pose of the car. After reviewing the message descriptions of type Lane, I was able to read information from these two topics and publish look ahead final waypoints.
Twist commands are linear and angular accelerations for the car. The waypoint follower node takes the final waypoints and publishes twist commands to dbw node. The DBW (drive by wire) is the bridge between hardware and software. It helps ROS actuate controls of the car.
The dbw node takes the twist commands and publishes commands to throttle, steering and brake. For the autonomous system to work, it needs dbw to publish commands at least 50 hz to be safe or else the system shuts down. We do this by setting this in loop method.

Now that the car is able to drive, it needs to be able to stop at traffic lights when it is red and start back again when it turns green.
We do this by the traffic light detection node. It subscribes to the camera image and feeds it to a traffic light classifier and based on the color of the light, it publishes the next waypoint with red light. Once the light detection node is complete, the waypoints updater node is where I make updates to the waypoints to adjust for velocity for stopping at upcoming red light if needed or starting again when the light turns green while respecting the maximum velocity and acceleration limits.

The walkthroughs really helped a lot with ros portion of the project. The part where I got terribly stuck was building the classifier. I attempted numerous approaches and attempt. I first started with a very simple image classifier, but soon found the size of the lights is much small compared to the whole image and was not getting desired accuracy. So I tried the object detection api, but could not get it to work with tensorflow 1.3. Then I started writing my own code for faster rcnn which looked promising for the part I was able to do, but could not complete it in time. So went back to resnetv1_101 with pretrained weights as a classifier along with some data augmentation and preprocessing with normalizing with mean and cropping the image by 10% top and bottom. This classifier works mostly but has some failures with red, but I think it is due to the lag as the test accuracy for 0.97.

So a very interesting project, that I took forever to complete is ready for submission.