Model Documentation

Project 7: Highway Driving/Path planning

PROJECT SPECIFICATION

This project submission includes the following files:

1. The Workspace with the compiling code
2. And a model documentation explaining the code model for generating paths in detail : “Model Documentation.pdf”

CODE DESCRIPTION

Firstly the code is divided in to 3 parts; 1. Prediction, 2. Behaviour planning and finally 3. Trajectory planning. In the following points, I explain what each section does in detail.

1. **Prediction:**

    a. This part of the code is primarily responsible for analysing the sensor fusion data which gives us information gathered by the car sensors, about the neighbouring vehicles on the road. Their position, and their velocity. We use these variables of the cars to estimate where the cars would be in the future, with respect to our vehicle, so that we can accurately plan the path and avoid collision.

    b. Firstly, we define 3 boolean variables, ‘car_ahead’, ‘car_right’, ‘car_left’, and set as ‘false’ the usage of these variables is explained later. A for loop is run through all the vehicles’ sensor fusion data and the value of their ‘d’ coordinate (Frenet) is obtained. Using the value of ‘d’ it is possible to calculate the lane in which the car is present. Here the lanes are subsequently numbered as 0, 1 and 2 as left, middle and right.

    c. The value of the ‘s’ coordinate, the velocity of the vehicle are subsequently obtained from the sensor fusion data as well. Here its important to note that the velocity is obtained in Vx and Vy coordinates, so the absolute is to be calculated.

    d. With the help of this speed, its possible to now predict the distance the car is going to travel in the next 0.02 seconds (distance = speed x time).

e. Using the above lane information and the ‘s’ coordinate, with a simple set of ‘if’ condition statements firstly its possible to find out if the car is in our lane. If yes, then the distance from us to the car is calculated. Thus giving us the information if the car is ahead of us and within a distance of 30m. If its true, then the Boolean variable car_ahead is set to true. Similarly, the lane and the distance is further checked if the car is in the right or the left lane and with in the danger zone of 30m, and the corresponding Boolean variables are set to true.

2. Behaviour Planning

a. In this section of the code, the values of the Boolean variables from the prediction section are used to tell the vehicle what to do, in each case.

b. Here we define the maximum velocity which the car is allowed to travel with (49.5 mph) so that its under the speed limit. We also define the maximum acceleration that the car is allowed to have so that there is no jerk when the car accelerates or decelerates.

c. Now with the help of ‘if’ statements, we check in the first case, if there is a vehicle ahead of us (car_ahead = true), if yes and also if we are not in the left lane, the lane of the car is decreased by 1, meaning that the car shifts to the lane left to it.

d. Similarly, if there is a car ahead of us, and if we are not in the right lane, and there is no car in the right lane, the car is prompted to shift into the right lane. If either left or right lane shift is not possible, the car is prompted to reduce its velocity with the decrement operator of the maximum acceleration which is defined as 0.224.

e. Finally, for the case in which we are either in the left or the right lane and there is no car ahead of us, and there is no car present on the middle lane, our car is prompted to change back its lane to the center lane. Also, if we notice that the velocity of our vehicle is less than the allowed velocity, its incremented by the max acceleration value.

3. Trajectory Planning

a. Finally after the prediction and the behaviour planning steps, its time to tell the car how to plan the trajectory that it should be following. This section of the code, consists of the concepts from the lesson 6.Project Q&A where the developer has explained how to use the spline library effectively to plan the trajectory.

b. Firstly we define two variables 'ptsx' and 'ptsy' to hold the points' x and y coordinates that we intend to create.

c. Using the information from the simulator, about the previous path that has been travelled by the car, we firstly check if there are enough points on the previous path, if its less than 2 points, we start the trajectory planning by using the car coordinates as the reference. We take the coordinates of the car and additionally we calculate the coordinates of a point previous to it, which lies on the tangent to the current direction of travel of the car.

d. Alternatively, if the previous path has more than 2 points, we choose the last and the second last points as reference points or anchor points to calculate the trajectory.

e. We now additionally calculate the x and y coordinates, of the points where the car would be after travelling 30 meters in the positive 's' direction, we do it by using the relative velocity value of the car and the lane value of the car and the getXY helper function. Similarly we calculate the coordinates of points 60m and 90m ahead of the car. These 3 points and the two ancor points from the above calculation are added to the variables 'ptsx' and 'ptsy', lets call these 5 points as anchor points.

f. We now set these 5 points on to the spline, and define two more variables 'next_x_vals' and 'next_y_vals' which would be holding the points that we would actually be travelling with our trajectory.

g. We add to the 'next_x_vals' and 'next_y_vals' the points in the previous path of the vehicle, which the vehicle has not already covered, so that we minimize the effort of creating more new points and use the previous path.

h. To calculate the spacing between these points, we use the reference velocity of the car and the target distance, which in this case is 30 meters. Here it should also be noted that the increment or decrement of the reference velocity value which was calculated in the behaviour planning segment, is done in this segment.

i. After calculating the spacing, these points are then added to the two variables and passed on to the path planner.