Assignment – 8

Vinayak Sharma (ID:20585279)

Trajectory Generation

1) The value of `d` in (s,d) frenet frame system helps keep car in a particular lane. To force the car to follow the center lane we set the value of d as a midpoint between start and end of the lane in question.

2) We generate multiple waypoints (up to 30m) where x in (x,y) coordinate would be direction in which the car is originally heading . We obtain y position of end point corresponding to distant horizon using spline function. We compute the distance d between initial and final point using trigonometry and use that to calculate the Normal N using d, velocity and time for each point which in our case is 0.02 seconds. We divide the normal N (distance from starting point) in 30 equal parts and use each part to calculate the corresponding y values on the spline. Thus, we now have the new coordinates for the car to follow along the road.

3) To avoid the collision with the car in front we keep track of a bool variable which is set to true if the distance between the closest car in front and our car is less than 30 metres and we use this variable to decrease the ref_velocity of the car.

4) To avoid cold start, we generate 1 artificial waypoint behind the car based on the current location of the car and we set ref_velocity to zero and increment the ref_velocity for every iteration by 0.224 while speed of car is less than 49.5 miles/hour (to avoid exceeding the speed limit).

Behaviour Planning

There are 6 lanes and we care about lanes marked 0, 1, 2. Thus, while looping through the sensor fusion data, we care about other cars in lane 0, 1 and 2 and store the closest car in each lane. There are 3 bool variables which are used to identify if any car is within 30 metres ahead of the car and within 7 metres behind the car in each lane. This is done to make safe lane changing decisions and even at times reducing the rel_velocity of our car. There is an enum of lateral state change of the car called `LatStateChange` consisting of values STAY_INLANE for current lane and LEFT_LANE_SHIFT for changing from current lane to left lane and similarly for RIGHT_LANE_SHIFT. After looping through the sensor fusion values, the next way points are calculated based on the decision made for the position of the car based current lane number car is in , how close others cars are around the current car and use this to update the state of the car using enum values. Based on the updated state of the car the waypoints are calculated based on that. For example, if the car is updated to shift to the left lane then the next way points are calculated in the left lane. There are three major if else blocks for checking the current lane number the car is on and deciding the next steps for the car based on how close the other cars are in the other lanes. Rest of the logic for trajectory generation remains the same. The other minor change was the rate of change of the ref_velocity of the car based on the closeness of car ahead. This was done to smoothen the car while path while braking / reducing velocity based on the closensess.