Path Planning Project

By: Tim Tobey

Term 3 August 19, 2017

The code of this model follow this general pipeline.

If traffic (other vehicles on the road) is not too close to the ego (car being modeled) , the ego continues to accelerate until it reaches the speed limit. The speed limit for the car is set on line 328 at a value of 49.4. It is set at this value, not the 50 MPH maximum to allow some variance in the model an movement between the points.

The car determines if traffic is too close by on line 282 where it calls the TooClose function. This function calculates the traffic ahead and the ego and determines if there it too small of gap between the ego and the traffic. If the gap is too small it sets and returns the too_close flag to true (line 36 my_fucntions.cpp). The minimum distance allowed in this model is 30m which set in line 28.

If the traffic is too close the model begins to determine if changing lanes is an appropriate reaction. It does this by first determining what lane are available to move into on 286 in the MakeLaneList function. This function simply returns a vector of available lanes depending on the ego’s current lane.

Next, the model calculates the lane traffic on line 291 with the CalcLaneTraffic function. This function creates a vector that represents a lane in the model (line 86 my_functions.cpp). The vector created is a vector of ones that is the length of the course. Traffic is evaluated to see if if it is in the same lane as the lane being modeled. If so, the vector is changed to zeros in the range from where the car is to where it will be in 30 moves (line 92). If a car is behind the ego and slower than the ego then 10 meters are zeroed out for safety's sake. As theoretically, if ego is going faster than traffic behind it, the two could not meet(line 96).

The function returns a vector to represent the traffic in the lane.

Next the model, calculates the size of the corridor that contains not traffic in the lane. This is done on line 292 with CalcSafeTurn function. The function receives the lane vector that was just previously generated.

The function looks at 5 meters behind the ego and 25 meters from that spot ahead. It does this by taking the ego s value and on line 123 and 124. It then adds up ones that are in sequence on 137. If the model finds a zero the model stops counting. The sum of the sequence of ones provides the amount of free space ahead of the ego. If the free space is less than the safety corridor then the free space returned is -1, meaning this choice is not safe (148). If the amount is greater than the minimum safety size then that amount is returned.

The model loops through the lanes until all the free space is calculated. Beginning on line 297 the model chooses the lane that has the highest free space available. If the model returns a available lane is is set on line 302.

Next the model reviews its other speed choices. If traffic is too close it will continue to slow down until there is a safe distance between the ego and traffic. If the car is going to change lanes is slows some to 40MPH (line 314). This is done in order to let faster cars go by and try to open some free lane space to change lanes. The car does not slow down too much as this can create problems with traffic in the rear of the ego.

The next part of the model is the process of setting waypoints for the car to drive along. The code is annotated and it is straight forward so I will not detail all of it here. A key setting is on line 390 and 399. As the model uses a spline, the anchor points are calculated by using 45 meters apart. As this was found to remove jerk and velocity issues with lane changes.

Lane changes are achieved by moving the anchor waypoints over by .33 for each waypoint for 3 times. This allows the vehicle to soothly move from one lane to another. A lane factor is used on line 400. This tell the model if the .33 factor should be positive or negative given the lane change direction. For instance, a lane change from lane 0 to lane 1 would be positive and a change from lane 2 to lane 1 would be negative.

On line 410 a change lane flag is reset and the lane change in process flag is reset on line 411.

After shifting the car’s view point (417), the model takes the anchor points on line 428 and splines the x and y values created earlier. On line 422 the calculation to break up the spline into points is begun.

On line 450, new travel points are added to any existing points that may still exist. My model used 45 points as this seems to work. After filling out the points, they are returned to the simulator for action.

Final notes: I wish I had more time to spend on this project even though I spent greater than 80 hours on it. The most significant improvement was supplied by the class helper video to reset the car heading to zero. This solved many problems for me and helped me move forward.

If I had the time, I would like to try to machine learning using the traffic feedback to see if I could better the time of the model.