ECE 495 A7 Written Questions

Trajectory Generation

1) How is lane following achieved?
   Lane following is achieved using the car's current Frenet coordinates (s, d) from localization and stepping forward the s value by some increment each time step, while the d value is set based on the lane we want to drive in. These Frenet coordinates are then converted to Cartesian (x, y) coordinates for the waypoint path.

2) How to use spline to generate a smooth trajectory?
   We use 5 anchor points (the 2 latest points from the car's previous path, and 3 points ahead of the car whose Frenet s coordinates are sparsely spaced 30 m apart) to generate a smooth spline function for our path (using the spline library). We then pick more closely spaced points inside the spline between our reference point and target point to construct our actual path of waypoints to travel to next. We compute d (the Euclidean distance between our reference point and our target point) then compute N = d/(0.02 * vel), where N is the number of waypoints along the spline between our reference and target. We then divide the x axis segment from our reference x coordinate to the target x coordinate into N evenly spaced x values, then pass these x values to our spline function to get the corresponding y values of the N waypoints. These waypoints will be added to our list of next points to travel to.

3) How to avoid collision with the car in front?
   We iterate through the sensor fusion data (s, d, x, y, vx, vy) reported by the simulator for all other cars on the road. We then use the d value of each car to determine if it is in the ego car's lane, compute its speed, project its future s value using its speed, and lower the ego car's speed incrementally if the other car in front of us is too close (within 30m).

4) How to avoid cold start?
   Instead of setting our initial reference velocity to 49.5 mph and changing to that velocity instantly, we start with it being 0. While the ref_vel is < 49.5, we keep incrementing it by .224 each step to avoid the cold start acceleration violation.

Writeup

Briefly explain your approach for behaviour planning and any modifications to the provided trajectory generation code (7-10 sentences)

We compute the list of adjacent lanes to the ego lane (e.g. if ego is in lane 1, the adjacent lanes would be {0, 2}), and we call this list `availLanes`. Then we iterate through the sensor fusion data list - if the check_car is in one of the `availLanes`, we compute the s value difference between the check_car and the ego car, and we do this regardless of whether the check_car is ahead of or behind the ego car. This way, when deciding whether the ego car should move into an adjacent lane, we can compute the distance to the closest cars both in front and behind (in that adjacent lane) and determine if there is a safe gap to merge into. If the ego car is less than 30m behind the closest car in its lane, we use the same approach as in the starter code and set the `too_close` flag to true, which triggers the ego car to decrement its speed. In addition, if we are less than 30m behind the car in front, we also check whether we should change lanes. We do so by computing a "score" called `bestLaneGap` for each of the `availLanes` based on the size of the "gap" we can merge into; this gap is computed using the distance to the closest car in front and behind that we computed earlier. We choose the lane with the optimal score, or we choose to remain in the current lane if certain thresholds are not met (so it's not worth changing lanes).

Note that before checking whether we should change lanes, we also check whether the ego car is in the middle of the lane (with some small tolerance), which prevents the ego car from attempting a lane change while a previous lane change is still in progress (i.e. it hasn't reached the middle of the lane yet).