Highway Driving
Project 2

Sarbjeet Singh
June 24, 2019

Trajectory Generation and Lane Changing

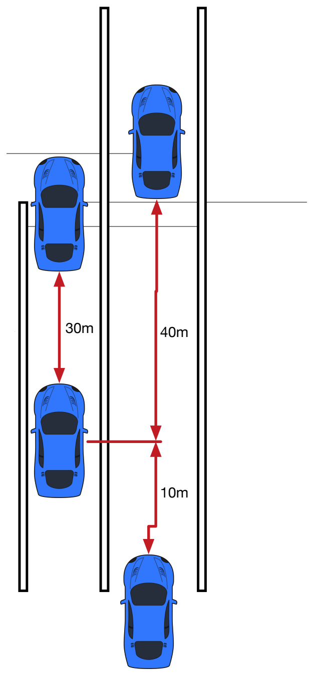
To achieve a vehicle that can successfully navigate the simulated highway, the following trajectory generation and lane changing methods were implemented.

For trajectory generation, the methods outlined in the Project Q&A were utilized. This involved using the spline library to build a spline made up of equally spaced points that the car could follow. The steps to generate this spline are:

1. Create a vector to store the points, which will make up the spline. Add 2 previous points to start off the spline. The previous points help the car transition from lane to lane and remove jitter between new spline calculations. If previous points do not exist, use current position and calculate one point back.
2. Add points for the car at 30 meter increments to generate a large spline
3. The spline points must be rotated to an axis at 0 degrees to avoid issues with a single Y coordinate have multiple X coordinates.
4. Break up the first 30m of the spline into points separated by the distance required for the car to travel at the set speed. Once calculated, rotate these points back to the correct angle of the car and pass the vector to the simulator.

Once this trajectory generation method was implemented, I investigated multiple ways to decide how to perform lane changes. A finite state machine is required to distinguish when certain actions must occur. I broke this down to 4 states:

- Keep lane: the car maintains it lane and increases or decreases speed as needed. If a car is detected in the same lane and the ego car slows down to 80% of the speed limit, the car will switch to the “Prepare Lane Change” state.
- Prepare Lane Change: This states contains the checks that the car must perform to approve a lane change. Conditions for both a left and right lane change are checked and any violations are giving a cost. The primary check is if there cars are in the lane to the left or right. If there are cars, they must be outside of a buffer zone, as shown in the image below. The buffer zone is 40 meter ahead of the car and 10 meters behind the car. This reason for these values is that if a car in front of the ego car is too close, the car in the lane that



could be switched into should be slightly ahead, otherwise a lane change does not make sense. Since a lane change occurs at a slower speed, a car behind our vehicle in another lane should be far enough behind for it not to collide. During the keep lane speed, the car aims to never drop below 30MPH, or about 40% of the speed limit, which is also slowest speed at which a lane change can be made.

Each car in violation will result in an increase of the cost. Lane changes can only occur if there is zero cost to the change. If a lane change is possible, the state changes to the Left Change Lane or Right Change Lane states. If no change is possible, the car returns to the Keep Lane state.

- Lane Change Right or Left: The car begins changing states and remains in this state until it is with 1m of the center of the new lane, at which points it returns to the keep lane state.

Overall, the trajectory is generated based on the spline and simply changing the lane value will create the appropriate trajectory. A finite state machine of 4 states is used to perform lane changes depending on the correct conditions being met.

For future implementations, a more robust finite machine can be used. This finite state machine can contains more states and conditions that allow the car to perform more complex maneuvers, such as squeezing between cars to pass or picking the fastest lane always. Additionally for the trajectory generation, a more complex cost function that weights multiple trajectories instead of a conditional statement for a single trajectory, can be used for more complex and accurate behavior.