**Reflection**

This project’s aim is to develop a path planner that incorporates collision avoidance and lane changes, all while attempting to maintain a constant speed below a given speed limit. The first major step in the code is to identify whether it is safe to initiate a lane change, and whether the vehicle is close enough to a vehicle in its front to warrant a lane change.

**Lines 249 to 305:** This is where sensor fusion data for every vehicle is analyzed. Initially, it is assumed that both a right lane change and a left lane change is safe, and that the ego vehicle is not too close to another vehicle in the front, in the same lane. The code here checks to see whether vehicles in adjacent lanes will be within some weighted factor of 30m of the ego vehicle within 0.02 seconds. If such a vehicle in the left lane is detected, the flag lane\_change\_left\_safe is set to false. Similarly, if such a vehicle in the right lane is detected, the flag lange\_change\_right\_safe is set to false. Finally, if a vehicle in the ego vehicle’s lane is detected to be within 30m after 0.02 seconds, the flag too\_close is set to true. When this flag is set to true, the following operations are attempted in this order of preference:

* Left lane change
* Right lane change
* Deceleration

Finally, if the too\_close flag is set to false, acceleration is applied at a favorable rate until a target speed of 49.5 mph is reached. The inputs to the code that sets up the vehicle path are ref\_vel and lane.

**Lines 317 to 415:** This section is where the trajectory generation takes place. Much of the code for this section was adapted from the walkthrough. The approach takes the following steps:

* For each cycle, the previous path’s end point is used as the reference, unless the previous path is almost empty, in which case the position of the car is used as the reference
* From this reference point, the lane value obtained from the previous step is used to compute the next 3 waypoints
* A spline is fit to these points. A local transformation is done to make the reference angle zero
* The next\_x\_vals and next\_y\_vals vectors are then populated with the leftover previous\_path\_x and previous\_path\_y.
* These vectors are then augmented with the newly computed points which are spaced out such that the car’s speed ends up being close to ref\_vel, after the previously done local transformation is reversed.