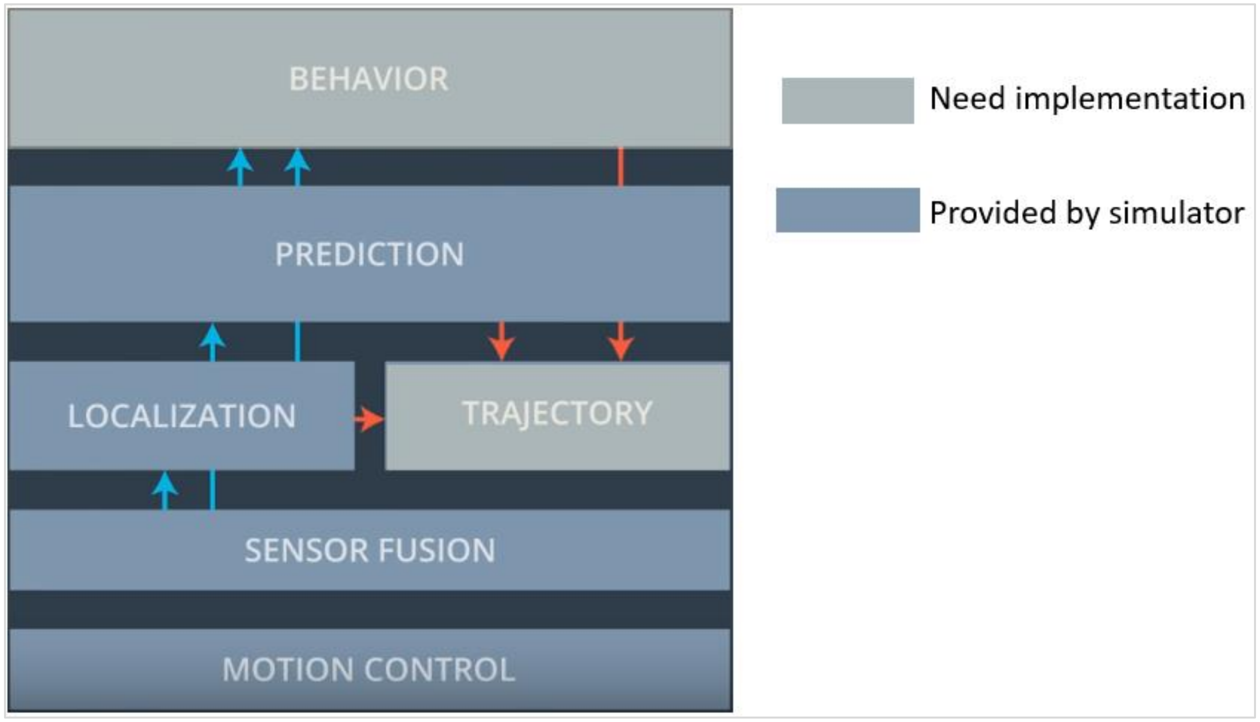
The Goal of the Project

In this project, the goal is to design a path planner that is able to create smooth, safe paths for the car to follow along a 3-lane highway with traffic. A successful path planner will be able to keep inside its lane, avoid hitting other cars, and pass slower moving traffic all by using localization, sensor fusion, and map data.

The Overflow of Data in Self-driving Car

The following figure shows the overflow of data in a self-driving car system. The data for the part **SENSOR FUSION**, **LOCALIZATION** and **PREDICTION** are already provided by simulator as following:



Localization data:

```
double car_x = j[1]["x"];
double car_y = j[1]["y"];
double car_s = j[1]["s"];
double s_of_car = j[1]["s"];
double car_d = j[1]["d"];
double car_yaw = j[1]["yaw"];
double car_speed = j[1]["speed"];
```

Predication data:

```
auto sensor_fusion = j[1]["sensor_fusion"];
```

So, the remaining task needed to be finished is to implement a **path planning scheme** and **generate a safe trajectory** for car to follow based on the ready **localization data**, which tells where the main car is, and the **predication data** indicating the status of other cars driving on the same side of road.

Path-Planning

The path-planning scheme will follow the way of how a real driver (actually myself) would take if being put under the same condition, that is:

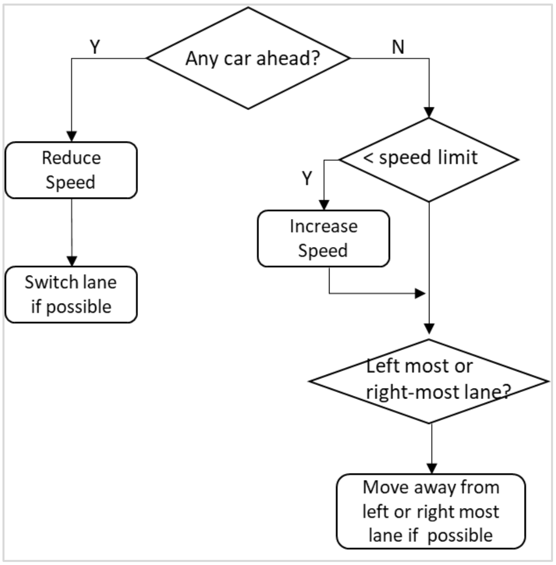
1. If there is a car less than 30 meters ahead, the first thing I would like to do is to reduce the speed and try to stay away from the front car.

   After that I would try to switch to the next lane with less traffic if possible:

   - If the car is at the left-most lane, it can only switch to the second lane from the left if available;
   - If the car is at the right-most lane, it can only switch to the second lane from the right if available;
   - If car is neither at the leftmost or rightmost lane, both left and right lanes are checked and then switch to the one with less traffic;
   - If no lane is available, the car will just stay at the current lane with reducing speed.
2. If there is no car ahead, but the speed is less than the speed limit, then speed will be increased.

   From safety point of view, I won't drive at either the left-most or the right-most lane for too long, I would change to any middle lane if it is available.

The following is the flow chart summarize the above description:

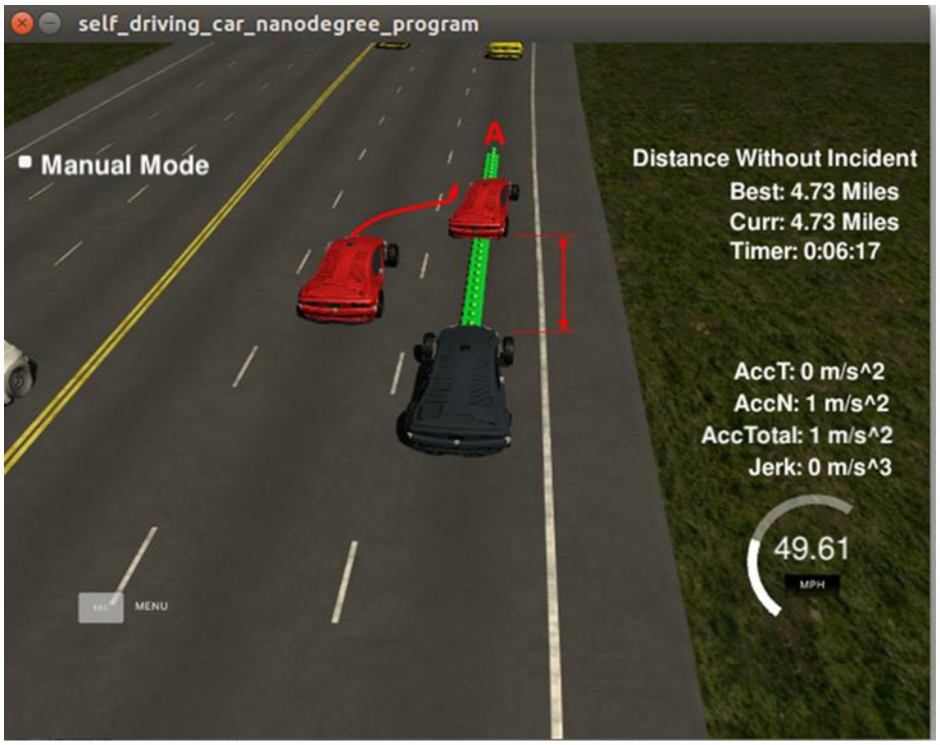

Any car ahead?
Y
N
Reduce Speed
Switch lane if possible
< speed limit
Y
Increase Speed
Left most or right-most lane?
Move away from left or right most lane if possible

Path Planning Implementation details:

```
if (any_car_ahead[lane])   // if there is a car within 30 meters ahead
{           //first thing is to reduce speed
            ref_vel -=std::max(0.224*std::exp(3.0/distance_ahead[lane]),0.02);
            if (lane==0 && any_lanes_available[1]){
              lane++;  //change to the second lane from the left if available
            }
            if((lane+1)==total_lane && any_lanes_available[lane-1]){
              lane--;  //change to the second lane from the right if available
            }
            //if neither left-most nor right-most lane, check lanes on both left or right side and pick the one
with less traffic
            if ((lane>0 &&any_lanes_available[lane-1]) ||
                        ((lane+1)<total_lane&&any_lanes_available[lane+1])){
                        if(any_lanes_available[lane-1] && any_lanes_available[lane+1]){
                          lane += distance_ahead[lane-1]>distance_ahead[lane+1]?-1:1;
              }else{
                                    lane += any_lanes_available[lane-1] ?-1:1;
                        }
            }
}
else // if there is no car within 30meters ahead
{
            if (ref_vel<speed_limit)
              ref_vel +=  0.224; //increase speed if the speed is less than speed limit
            if(lane==0 && any_lanes_available[1]){// switch away from the left-most lane if possible
                        lane += distance_ahead[0]>distance_ahead[1]?0:1;
            }
            if((lane+1)==total_lane && any_lanes_available[lane-1]){ // switch away from the right-most
lane if possible
              lane += distance_ahead[lane-1]>distance_ahead[lane]?-1:0;
            }
```

Special case:

It is observed that a couple of times incident happened when there is a car swerved and cut into the green predicted lane as show in the following figure. Even though the black car keeps >30meters away from the front yellow car with a speed determined by green curve, the red car cut in within the predicted waypoints, resulted into black car rear-ended into red car.



There are two possible solutions to avoid this collision:

> one is to monitor red car's heading direction based on the sensor fusion data, if it is the same as that of black car, meaning there is no lane changing. If red car heading direction changes toward the current lane, like a human sees the red car's right turn light flashing, the black car should start to reduce speed;
>
> the second solution is to monitor the current distance to any car in the front, besides the predicted distance previous_size * 0.02S later. If the current distance is less than 30 meters, the black car should starts reducing speed.

In this project, the second solution is implemented for simplicity, even though the first solution has some advantage, like it gives more time for black car to reduce speed.

For the above special case, the car speed is reduced based on distance, that is:

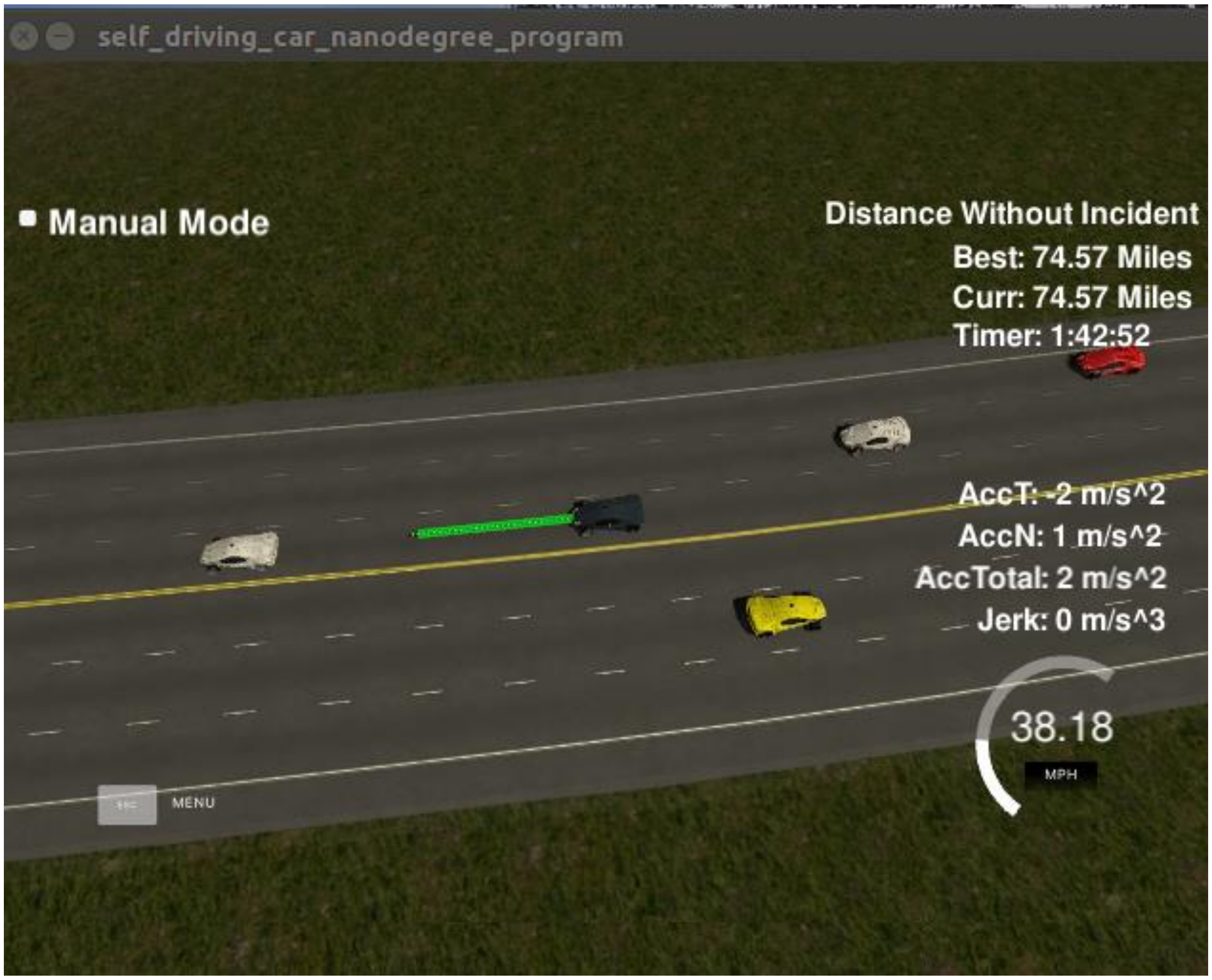
$$ref_{vel} = ref_{vel} - 0.224 * \ e^{3.0/distance}$$

ref_vel -=std::max(0.224*std::exp(3.0/distance_ahead[lane]),0.02);

If the distance is too small, like the case the red car suddenly cut in, the speed could be reduced way faster than 0.224. This scheme is exactly the same as what a real driver could do. He will step on brake so hard trying to stop the car, even though it will give uncomfortable jerk.

Simulation

The following figure shows the longest distance the car can go without any incident, it can go as far as 74 miles.


self_driving_car_nanodegree_program
Manual Mode
Distance Without Incident
Best: 74.57 Miles
Curr: 74.57 Miles
Timer: 1:42:52
AccT: -2 m/s^2
AccN: 1 m/s^2
AccTotal: 2 m/s^2
Jerk: 0 m/s^3
38.18
MPH
MENU