Reflection:

TERM3: PATH PLANNING PROJECT

Self-Driving Car Engineer Nanodegree

Rafael Barreto

1. INTRODUCTION

In this project was implemented a path planning for an autonomous car be able to go through a highway with safety, within the speed limit, and in the way that it is comfortable for the passengers with maximum acceleration and jerk (rate of change of acceleration) not exceeded any time.

2. DEVELOPMENT

Some attempts were made trying to make the car go through the highway without any accidents or bad occurrences (for instance, overcome the speed limit) and it is able to change lanes if there is a slow vehicle in front of the autonomous vehicle. The best approach achieved was based on making the Finite State Machine simple as possible, regarding the number of states and the behavior of each state.

2.1 Sensor Fusion

For each iteration, the sensor fusion system must check what is going on in the environment around the car. First, the sensor fusion system check if there is a car close in front of the ego vehicle and after it checks if there are other lanes free for a future possibility of change lanes. The middle lane is preferred between the right or left lane in order to facilitate the car to flow better in traffic. Using Frenet Coordinates system, a lane on one side of the ego vehicle is considered free if:

- the sensor fusion system does not detect a vehicle in this lateral lane with “s” value bigger than the ego vehicle “s” value (car_s) and with a difference of 40 meters:

```
if ((check_car_s > car_s) && ((check_car_s - car_s) < 40))
```

- the sensor fusion system does not detect a vehicle in this lateral lane with “s” value less than the ego vehicle “s” value (car_s), with a difference of 6 meters and with speed (check_speed) bigger than the ego vehicle.

```
if ((check_car_s < car_s) && ((car_s - check_car_s) < 6) && (check_speed < car_speed))
```

Besides that, the fusion system checks how many cars exist in each possible side lane. This is useful when the car is in the middle lane and the system have to decide between two lane free what is the best option.

2.2 FSM

The Finite Sate Machine designed has 5 states shown below:

- KL: Keep Lane
- PLCL: Prepare Lane Change Left
- LCL: Lane Change Left
- PLCR: Prepare Lane Change Right
- LCR: Lane Change Right

The car will try to drive at the speed limit and in the middle lane. When the vehicle drives at the middle lane the probability of being stucked in traffic is lower than when the vehicle stays in one side lanes. Besides that the FSM stays more simple.

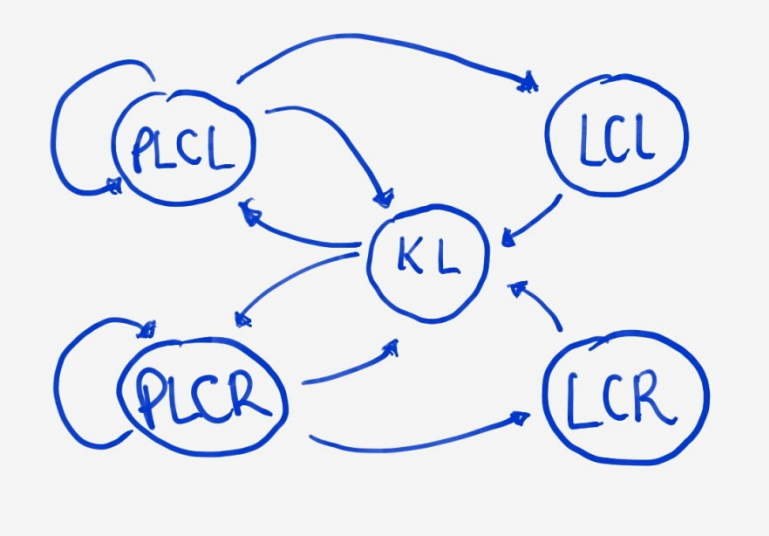

Figure 1: FSM with 5 states

If the sensor fusion system detects a vehicle in front of the ego vehicle with speed less than the speed limit of the road (50 mph) a Boolean variable is activate and if there is a side lane free the change the state from KL to PLCR (if this free lane is on the right side of the car for instance). In the is state PLCR (or PLCL) the sensor fusion system needs to check 50 times if this selected side lane is really free. If the check procedure fails due some car detected the count restart again and the system tries one last time to confirm if this side lane is free. If fail again, the state of the vehicle return no KL state. If the sensor fusion confirms the free side lane, the next state will be the LCR (or LCL) that is a very fast state that changes the variable lane and provoke the generation of the path to the new lane.

2.3 Path Planning: Point Paths and cubic interpolation

The point paths are made using a cubic spline interpolation function provided by the library “spline.h”. This function provides easily paths with smooth curves. The first points of the path are the last points of the previous path provided by the simulator.

The ego vehicle should drive in the reference velocity and using the spline function the points are separated using the follow script explained in the Project Walkthrough Video:

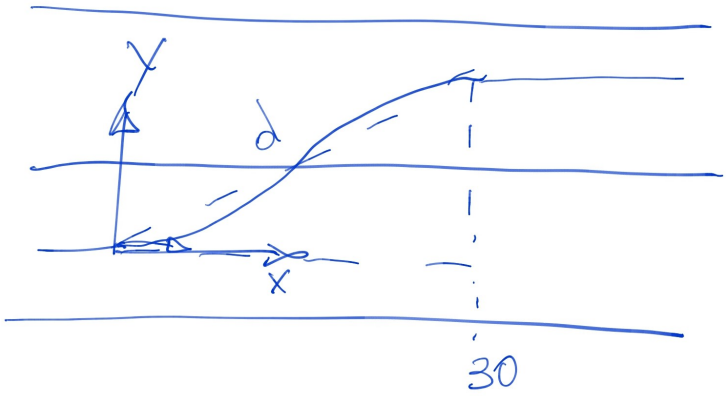

Figure 2: The distance of each path point will be (target_x/N) meters.

```
// Calculate how to break up spline points so that we travel at our derired reference velocity
double target_x = 30.0;
double target_y = s(target_x);
double target_dist = sqrt((target_x)*(target_x) + (target_y)*(target_y));

double x_add_on = 0;

cout << prev_size << endl;

for(int i = 1; i <= (POINTS - previous_path_x.size()); i++) // previous_path_x.size() is (POINTS - 1)
{
    double N = (target_dist/(0.02*ref_vel/2.24));
    double x_point = x_add_on + (target_x)/N;
    double y_point = s(x_point);

    //set a new checkpoint
    x_add_on = x_point;

    double x_ref = x_point;
    double y_ref = y_point;

    // go back to global coordinates .. shift and then rotation
    x_point = (x_ref*cos(ref_yaw) - y_ref*sin(ref_yaw));
    y_point = (x_ref*sin(ref_yaw) + y_ref*cos(ref_yaw));

    x_point += ref_x;
    y_point += ref_y;

    next_x_vals.push_back(x_point);
    next_y_vals.push_back(y_point);

}
```

Figure 3: Making path points with spline cubic interpolation

2.4 Safe Distance

The safe distance between the ego vehicle and one other vehicle in front of it depends on the velocity of the ego vehicle. The safe distance is interpreted as a distance long enough to allow the complete stop of the ego vehicle in case of abrupt braking of the front car that is being followed. Therefore, the safe distance can be calculated as:

$$safe\_distance = 2 + car\_speed/2.24 \text{ (in meters)}$$

2.5 Reference Velocity

When the ego vehicle has no traffic ahead the reference velocity gently increases until achieving 49.5 mph. When existing one slow car in front of the ego vehicle, as shown at Figure 5, the reference velocity becomes the estimated velocity of the car in front of it. This information is provided by the Sensor Fusion System. This way the ego vehicle can follow the car in front it until an opportunity of change lane with safety.

3. RESULTS

With this project the ego vehicle was able to drive more than 30 min without any accidents. The car drives according to the speed limit and it is able to change lanes safely when there is a car in front of it. The curves and the moments of change lanes are made with smooth paths generated by the spline cubic function.

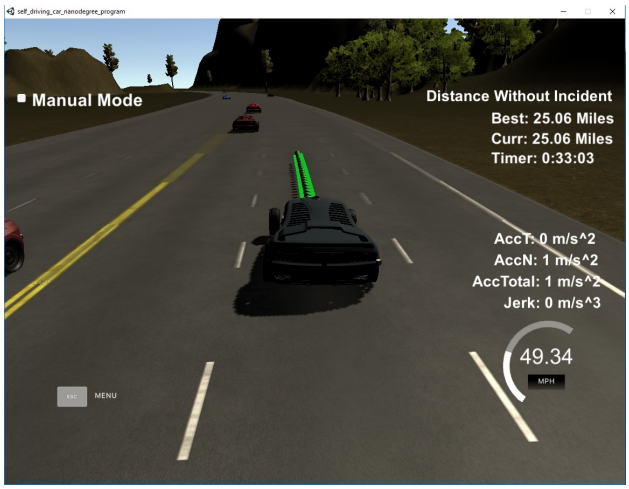

Figure 4: The ego vehicle free to travel at the speed limit.

Figure 5: The ego stuck in traffic with reference velocity of the car in front of it.

Figure 6: The ego vehicle make a change lane to get a free lane where it can drive at the speed limit