Writeup

Path Planning Project

My 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 is able to keep inside its lane, avoid hitting other cars, and pass slower moving traffic all by using localization, sensor fusion, and map data.

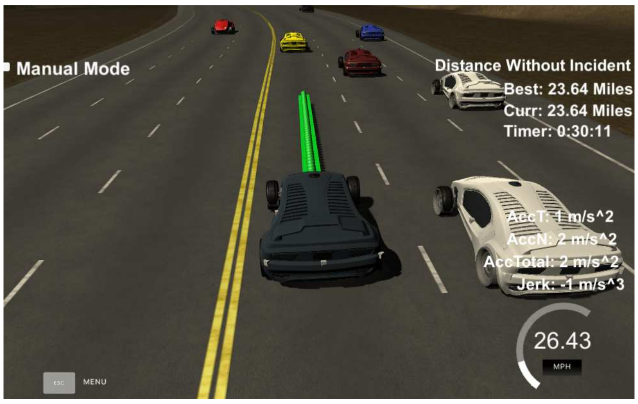
Rubric Points **The code compiles correctly.**

Code must compile without errors with `cmake` *and make. Given that we've made* `CMakeLists.txt` *as general as possible, it's recommend that you do not change it unless you can guarantee that your changes will still compile on any platform.*

Yes, Code is compileable and no errors encountered.

Rubric Points **The car is able to drive at least 4.32 miles without incident..**

The top right screen of the simulator shows the current/best miles driven without incident. Incidents include exceeding acceleration/jerk/speed, collision, and driving outside of the lanes. Each incident case is also listed below in more detail.



Yes, 23.64 Miles achieved.

Rubric Points **The car drives according to the speed limit.**

The car doesn't drive faster than the speed limit. Also the car isn't driving much slower than speed limit unless obstructed by traffic.

Yes, the upper limit is constrained at 49.5 mph, and the car will slow down only if there is a traffic jam in all its adjacent lanes.

Rubric Points Max Acceleration and jerk are not Exceeded.

The car does not exceed a total acceleration of 10 m/s^2 and a jerk of 10m/s^3.

Maximum acceleration and deceleration is constrained by a reasonable numerical value as introduced in the class. And double lane changing is prohibited so the lateral jerk is also under control. No warning message during the whole trip of 30 min.

Rubric Points Car dose not have collisions

The car must not come into contact with any of the other cars on the road.

Yes, confirmed.

Rubric Points The car stays in its lane for the time between changing lanes.

The car doesn't spend more than a 3 second length out side the lane lines during changing lanes, and every other time the car stays inside one of the 3 lanes on the right hand side of the road.

Yes, the lane changing manoeuvre is clear-cut. and is not slalom-like while driving straight.

Rubric Points The car is able to change lanes

The car is able to smoothly change lanes when it makes sense to do so, such as when behind a slower moving car and an adjacent lane is clear of other traffic.

the lane changing decision is just in time when there is open road available at a nearby lane. in this case no deceleration is expected.

Rubric Points There is a reflection on how to generate paths.

the code model for generating paths is described in detail, This can be part of the README or a separate doc labelled "Model Documentation"

My implementation is based on the walkthrough video presented in the class.

Collision identifier was introduced in my code block. sensor fusion data was injected into the algorithm, inside which we calculate other vehicle's speed, s coordinate from Frenet and lane number. and using them for the collision detection.

we split the collision detection into 2 categories, if there is a car drive in front of us less than a safety distance, then we activate the `too_close` flag, set it to 'true'.

```
if((check_car_lane!=cur_lane) && (check_car_s-car_s)<10.0 && (check_car_s-car_s)<30.0){
    too_close = true;
}
```

if a car drives in adjacent lane, either drives in front of me up to 30 meters or later behind less than 10 meters, will trigger a specific flag labeling the path free of impact, then preparing for lane change.

```
if((check_car_lane==cur_lane) && (car_s-check_car_s)>0.0 && (check_car_s-car_s)<30.0){
    ...
}
```

a Boolean vector is used to labeling/activate the available adjacent lane/lanes.

```
vector<bool> laneFlag = {true,true,true};

laneFlag[cur_lane] = false;
```

in order to labeling dis-qualified lanes:

```
if((check_car_lane==cur_lane) && (car_s-check_car_s)>0.0 && (check_car_s-car_s)<30.0){
    laneFlag[check_car_lane] = false;
}
```

Freedom of Lane changing maneuver is constrained so double lane slalom is prohibited.

```
bool oneLaneChange(int cur_lane, int target_lane) {

    if (cur_lane == 0 && target_lane == 1) {
        return true;
    }
    if (cur_lane == 1 && (target_lane == 0 || target_lane == 2)) {
        return true;
    }
    if (cur_lane == 2 && target_lane == 1) {
        return true;
    }
    return false;
}
```

to trigger lane shifting. if `too_close`, meanwhile exists a qualified candidate lane labeled ‘ture’, then activate lane shifting.

```
if (too_close) {
     for (int i = 0; i < 3; i++) {
         if (laneFlag[i] && oneLaneChange(lane, i) {
             cur_lane = i;
         }
     }
}
```

To ensure the car drives according to the speed limit at 50 mph, the car lunched at 0 mph with step size of velocity coefficient at 0.224, which results a reasonable acceleration without introducing any uncomfortable jerk. the maximum velocity is also under monitoring, if we go beyond 49.5mph, no more speed up.
`too_close` label is activated to reduce the speed by the same rate to avoid jerk feeling while decelerating.

```
if (too_close) {
    ref_vel -= 0.224;
} else if (ref_vel < 49.5) {
    ref_vel += 0.224;
}
```

as taught in the class, trajectory generation module yield a smooth cubic spine. this track interpolation method provided an outperformed result. the spline library header file is quoted, and we feed 5 points into the pipeline: 2 points from the trajectory generated in the last time step; 3 more waypoints at 30/60/90 meters ahead in a given lane.

```
vector<double> next_wp0 = getXY(car_s + 30/60/90,
                          (2 + 4 * cur_lane), map_waypoints_s, map_waypoints_x, map_waypoints_y);
```

Finally there is a counter introduced. the purpose is to moderate the lane changing response, to discourage an aggressive driving behavior.

```
num_units_in_current_lane += 1;
...
if ((laneFlag[i] && oneLaneChange(lane, i) && (num_units_in_current_lane > 10)){
...
num_units_in_current_lane = 0;
}
```