Path Planning Project

Introduction

This project aims to design and implement a path planner that is able to create smooth, safe paths for the car to follow along a 3-lane highway with traffic.

Rubric Points

Here, I describe each rubric point and how I have attempted to address it.

The code compiles correctly.

The code indeed compiles without any errors.

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

In order to have the car drive around the loop without any incident, I focused on addressing each of the following rubric points.

The car drives according to the speed limit.

It was easy to have the car follow the speed limit. The entire track has a single speed limit of 50 mph. The speed can be easily controlled by varying the distance between the path points provided by the path planner to the simulator.

- The reference velocity is declared in line 209 of main.cpp. This is the velocity that the path planner tries to target.
- The velocity is decreased whenever the vehicle is too close to another vehicle in front of it. This is done by decrementing the reference velocity, as seen in line 347 of main.cpp.
- The velocity is increased whenever it is less than 49.5 mph. The increment per time step is a constant 0.223 mph. As a result, the velocity can never exceed 49.723 mph, which is slightly below the speed limit. This can be seen from line 368 of main.cpp.

Max Acceleration and Jerk are not Exceeded.

I used the same technique as described in the project walkthrough to ensure that the maximum acceleration and jerk were not exceeded:

- The velocity is never increased or decreased by more than 0.223 mph in a single time step. This corresponds to a maximum change in velocity of less than 0.1 m/s in 0.02 s, i.e. a maximum acceleration or deceleration of less than 5 m/s^2. This is half the limit of 10 m/s^2.
- During lane changes, the sideways component of acceleration becomes non-zero. As suggested in the project walkthrough, I am using a spline to guarantee smoothness of the path points; this reduces the likelihood of high values of sideways acceleration during lane changes.

- Since the maximum and minimum acceleration are 5 m/s^2 and -5 m/s^2 respectively, and the jerk is calculated over 1 second intervals, the jerk can never have a magnitude higher than 10 m/s^3.
- Since there is guaranteed continuity between different pieces of the spline generated, the sideways jerk experienced during lane changes is limited as well.

Car does not have collisions.

In order to ensure that the car does not have collisions, the following steps have been taken:

- The path planner reduces the velocity as the vehicle begins approaching a slower moving vehicle in front of it.
- In order to do this, I calculate the safe distance towards the front using the distance that the vehicle can cover in 2 seconds at its current speed. This is done in line 270 of main.cpp.
- The velocity reduction code kicks in as soon as there is another vehicle within the safe distance.
- The magnitude of the reduction is inversely proportional to the room in the lane ahead; this ensures that as we get closer to the vehicle in front of us, we start reducing the speed more aggressively. See line 347 of main.cpp for details.
- The path planner also uses velocity information about vehicles behind to judge the safety of lane changes such that collisions are avoided. For this, it computes a safe distance to the rear for each vehicle detected by the sensor fusion module. If a vehicle in a lane next to us is coming towards us at high speed, a lane change would be possible only if it is behind us by a high distance. This is similar to checking a rear-view mirror and blind spots before changing lanes.

The car stays in its lane, except for the time between changing lanes.

The car stays in its lane at all times, except when it is changing lanes. This is accomplished using a lane variable declared in the beginning; see line 201 of main.cpp, and then using this lane variable to generate new waypoints to be interpolated using the spline.

The car is able to change lanes.

The lane changing algorithm is the highlight of my work in this project. I believe that I have created an interesting algorithm for changing lanes in order to allow the car to make maximum forward progress without compromising safety.

- Lane changing is accomplished by changing the lane variable to the desired lanes; the waypoints are then generated in the new lane, and the spline through the existing waypoints in the current lane and the new waypoints with high spacing (30 m) in the target lane results in a smooth trajectory during a lane change.
- A lane change is performed only when it is safe to do so. This involves making sure that no vehicle is expected to be within the front safe distance and the rear safe distance of

the car between the start time and end time of the lane change. This is ensured using the following conditions:
    - If there is another car in the unsafe zone in the target lane right now, it is unsafe to start a lane change right now. See line 321 of main.cpp for an example.
    - If there is going to be another car in the unsafe zone of the target lane at the time the lane change is expected to end, it is unsafe to start a lane change right now. See line 320 of main.cpp for an example.
    - If a car in the target lane that is behind us right now outside the unsafe zone but will be ahead of us outside the unsafe zone at the time the lane change is expected to end, it is unsafe to start the lane change right now, since the car is travelling much faster than we are and will cross us during the time we are moving into its lane. See line 322 of main.cpp for an example.
    - The previous point also applies in the opposite case, i.e., where we are going to pass a slow-moving car while moving into its lane. See line 323 of main.cpp for an example.
- While safety is a necessary condition for lane change, it is not a sufficient condition. A lane change is performed only when there is a reason to do so, i.e. when performing the lane change would help in the forward progress of the car:
    - Lane changes are considered when the car is too close to a vehicle right in front of it. This indicates that the vehicle ahead is moving slower than the speed limit, and if it is safe to shift to an adjacent lane, the car should do so since it would improve its forward progress. See lines 349-364 of main.cpp for the code.
    - Lane changes are also considered when the car is not immediately blocked by traffic. This is done on the basis of the amount of forward room that is available in each lane and the speed of the car directly in front of our vehicle in each lane. Changing the lane pre-emptively when the car is not immediately blocked or slowed down by traffic ahead makes for even better forward progress because if the lane change works out, the car does not even have to slow down due to traffic it passes.
        - The cost of a lane is calculated based on how much forward progress the car can make in that lane during the next 8 seconds. This depends on the current speed of the vehicle, the speed of the vehicles in front of the vehicle in each lane, and the distance to the next vehicle ahead in each lane, i.e. the room available in the lane. See line 341 of main.cpp for the code where this cost is calculated.
        - See lines 372-386 of main.cpp for the code for deciding whether or not to change the lane based on the lane costs.

There is a reflection on how to generate paths.

This document is a description of how my path planner generates paths for the vehicle. In addition, it is also a reflection about the general topic of path planning. Here, I will describe a couple of limitations of this path planner:

- This path planner looks only at the immediate future. It only generates trajectories up to 1 second long involving single lane changes. It does not generate complex trajectories over longer periods involving multiple lane changes. If we extend the horizon and also perform planning at that level, this can greatly improve the forward progress of the vehicle in tricky scenarios, such as when the vehicle is stuck in a corner lane due to traffic in its lane and the adjacent lane, while the other corner lane is completely free.
- This path planner is not perfect. Even though it is able to drive long distances without incidents, sometimes incidents do happen. Some of the ways in which incidents can happen are described below:
    - If a vehicle ahead of us moving slower than us quickly moves into our lane, our car would not have enough time to stop. Being able to stop in such situations would require us to also track what each vehicle is doing and detect situations like this.
    - Using splines to generate trajectories leads to more sideways movement within the lane than is perhaps desirable. This is especially true on curved portions of the highway. As a result, the car sometimes gets dangerously close to vehicles in the adjacent lane. Sometimes, it stays slightly outside its lane for more than 3 seconds, resulting in the corresponding incident being logged.
    - Using splines to generate lane change trajectories sometimes leads to a high sideways jerk during the lane change.