ROB 535 Control Project Team26 Documentation

Jiachen Jiang
jiachenj@umich.edu

1 Task Definition

The control project of controlling a bicycle model mainly contains two parts as follows.

- **Task 1** We are required to design a controller for the system to get from the beginning to the end of a pre-defined track as rapidly as possible;
- **Task 2** We need to develop a control design algorithm (which may or may not modify the controller constructed in the first task) to avoid obstacles that are known only at run-time.

2 My Contribution

I mainly focus on task2. There are three parts of my work, including Avoid obstacles module, generate reference speed and angle module and PI controller module.

2.1 Avoid Obstacles

The function of this module is getting target path according to obstacles and test track. Specifically, the target path is a series of positions and angles, e.g. (x, y, θ), which can avoid all obstacles seen within 150m. I choose the center lane as the target path initially. If I detect an obstacle on my target path, I would calculate the minimum distance between the center of the obstacle and the left/right lane to determine whether the obstacle on the left or right. Then we would choose the left lane or right lane as our target path according to position of the obstacle.

To detect the obstacle, I draw a circle around the polygon of the obstacle with a little bit large radius. Then I calculate the distance between the centre of circle and our target line and compare it with the radius. If the distance is smaller than the radius of obstacle, then the obstacle is detected.

To avoid touching the boundaries of left/right lane, I set the target path only at 2/3 between center lane and left/right lane.

2.2 Generate Reference

The function of this module is generating reference velocity(u) and angle(ϕ) according to the first module's target path. Since the points in original path is sparse, I interpolated it for 20 times. Then for each simulation time step, e.g. 0.01, I get the reference velocity(u) and angle(ϕ).

For velocity(u) in time t,

$$u(t) = \frac{\sqrt{(x_1(t) - x_2(t))^2 + (y_1(t) - y_2(t))^2}}{t_{perpoint}} \tag{1}$$

where $x_1(t)$ and $y_1(t)$ are positions of start point in target path in time t, $x_2(t)$ and $y_2(t)$ are positions of end point in target path in time t. And $t_{perpoint}$ is constant time used from start point to end point.

For angle(ϕ) in time t, I use the angle(θ) in start point in target path in time t directly.

2.3 PID Controller

The PID controller would output control $[\delta, F_x]$ to meet reference in the second module. It only needs to simulate 50 points to control the car for 0.5s. I use Proportional controller for F_x according to reference velocity and Proportional+Integral controller for δ according to reference angle.

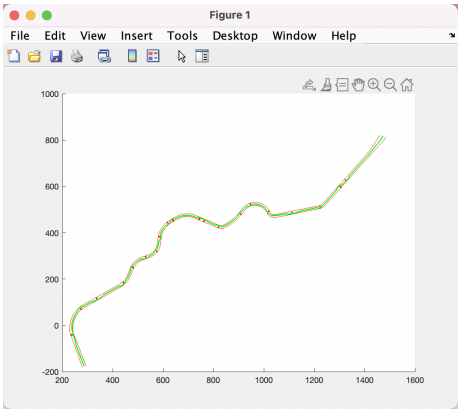
To generate appropriate $F_x(t)$,

$$F_x(t) = Kp_u * (u_{ref}(t) - u(t)) \tag{2}$$

where Kp_u is velocity gain of proportional part and $u_{ref}(t)$ is reference velocity and $u(t)$ is our current velocity.
To generate appropriate $\delta(t)$,

$$\begin{aligned} \delta(t) &= Kp_\phi * (\phi_{ref}(t) - \phi(t)) + Ki_\phi * I_{part}(t) \\ I_{part}(t) &= \sum_{i=1}^{t*0.01} 0.01 * Sign(l/r) * d_{ref}[i] \end{aligned} \tag{3}$$

Where Kp_ϕ and Ki_ϕ are velocity gain of proportional part and integral part. $\phi_{ref}(t)$ is reference angle and $\phi(t)$ is our current angle. $I_{part}(t)$ is the integral part. The $sign(l/r)$ indicates that whether the car is on the left or right of reference path. The $d_{ref}[i]$ is the distance from current point to reference point.



(a) Trajectory of avoiding obstacles in real time.

```
struct with fields:

                                Y: [29901×6 double]
                                U: [29901×2 double]
                       t_finished: 298.7500
                            t_end: 299
                left_track_position: []
                    left_track_time: []
   left_percent_of_track_completed: []
                    crash_position: []
                        crash_time: []
  crash_percent_of_track_completed: []
                    input_exceeded: [0 0]
         percent_of_track_completed: 1
                          t_score: 298.7500
```

(b) Information of avoiding obstacles in real time.

Figure 1: Task 2 Result

3 Teammates' Work

- **Zimin Lu** Control Task: mainly focus on task1. Generate the control input matrix using PID methods successfully.
- **Chengtian Zhang** Control Task: Explore MPC method for task1 and calculate the Jacobin matrix by hand. Generate the reference lane for task2.
- **Shuoqi Wang** Control Task: Calculate Jacobin matrix using matlab jacobian() function. Test MPC method for task2.
- **Lingfei Huo** Perception Task.