Parallel Parking Implementation Using Constrained Finite Time Optimal control

A specific case solving parking problem

Author: Anke Zhang, Di Wu, Ziang Wang

***Abstract*—We are a subteam under Localization and Positioning project. Our mini project for E296MA is on optimal control for parallel parking problem in finite horizon. The team tries to minimize a cost function which indicates the length of the parking trajectory, under constraints on center of mass (COM), velocity of COM, acceleration of COM and steering angle. By minimizing the cost function, the team is able to find the optimal trajectory to park the car under the constraints. The sole purpose of this project is to get the team familiar with the dynamic model of vehicle and perform optimal control over the model. The team solves this problem with MATLAB, YALMIP[2] toolbox and IPOPT[3] solver. This mini project will be a great kickstart to the capstone project in next semester.**

I. Introduction

The vehicle model for the parking problem can be simplified into a Kinematic Bicycle Model[1]:

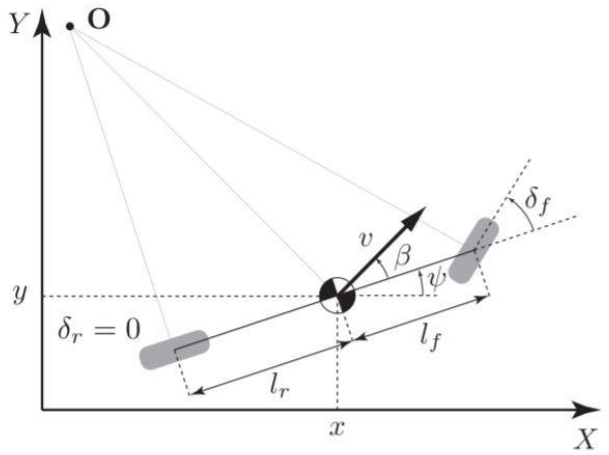

Fig 1: Kinematic Bicycle Model[1]

Where x and y are COM's position coordinates. v and a are the velocity and the acceleration of the COM. β is the angle of the velocity of the COM. ψ is the global heading angle relative to x-axis. δ_f is the steering angle of the front wheel. And the wheel l_f is the distance between the front axle and COM. l_r is the distance between the rear axle and COM. The followings are the nonlinear continuous time equations that describe the kinematic model[1]:

$$\begin{aligned}
\dot{x} &= v\cos(\psi+\beta)\\
\dot{y} &= v\sin(\psi+\beta)\\
\dot{\psi} &= \frac{v}{l_r}\sin(\beta)\\
\dot{v} &= a\\
\beta &= \tan^{-1}\left(\frac{l_r}{l_f+l_r}\tan(\delta_f)\right)
\end{aligned}$$

The parking problem is solved based on the kinematic bicycle model.

$$min \sum_{k=0}^{N} ||x_{k+1} - x_{k+1}|| + ||y_{k+1} - y_{k+1}||$$

$z_{k+1}=z_k+f*\Delta t$

$z_{min} \leq z_k \leq z_{max}$

$u_{min} \leq u_k \leq u_{max}$

$|a_{k+1} - a_k| \leq a_b$

$|\delta_{fk} - \delta_{fk+1}| \leq \delta_b$

$z_0=\underline{z_0}$

$z_N=\underline{z_N}$

Where $z=[x,y,v,\psi]^T$. u is the input, which includes a and δ_f. f is the kinematic model, which specified in the function bikeFE.m and the equation above. In this case, we didn't include β in our dynamic model for simplification. We assume the angle β as 0 degree all the time. In fact, β actually remains a small angle since our car length is low. If the simulated car length is big, β need to be take into account. Then, the cost function indicates the length of the parking trajectory. By minimizing the cost function, one could find the shortest trajectory to park the vehicle.

YALMIP [2] toolbox is used to solve the optimization problem. This toolbox makes solving optimization more generalized and allows the user to set up the problem description more easily. YALMIP is capable of calling external solvers and it

will transform the constraints to the needed format of the solver. User can define what solvers to use. The kinematic model being used in this report is nonlinear. Thus, the team defines IPOPT (Interior Point OPTimizer) [3] as the solver in YALMIP to solve the nonlinear optimization problem.

II. Problem Setup

First, we imposed our Problem Model to be:

Distance from center of mass of car to Front wheel: 1.738m
Distance from center of mass of car to Rear wheel: 1.738m
Max Steering Angle: +/- 0.6 rad
Max Acceleration: +/- 1.5m/s^2
Max Velocity: +/- 10m/s^2
Max Change of Acceleration (Jerk): +/- 0.2m/s^3
Max Change of steering: +/- 0.06 rad/s
Car Moving Range in X direction: +/- 20m
Car Moving Range in Y direction: +/- 10m

We also imposed the constraints on Jerk and angular rate of steering. Although these properties are not specified in the Model. We think Imposing these characteristics is critical for a more realistic simulation.

Then, we imposed the cost function the same as the one in Introduction. By the IPOPT Method of optimization, the car will find its optimal path with the shortest trajectory.
In our case, we want to solve the parallel parking problem for the car from various initial position. Thus, we initialized two initial example conditions. One is near to the desired final state. The other initial condition is far from the desired position:

State Variable: $[x, y, v, \psi]$
Initial State 1: $[8, 8, 0, 0]^T$
Initial State 1: $[3, 3, 0, 0]^T$
Final State: $[0, 0, 0, 0]^T$

The state variables are X Position, Y Position, velocity, and heading angle of the car. Because we are doing parallel parking, the heading of the initial state and the final state should point in the same direction.

III. Simulation Result

A. Result of Far initial condition:

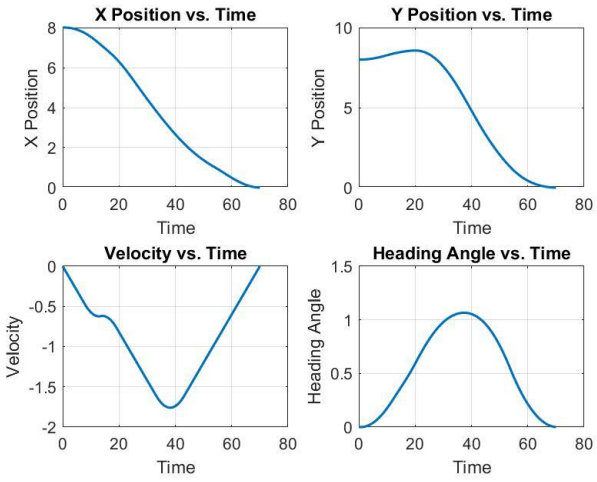

Fig 2: State Result

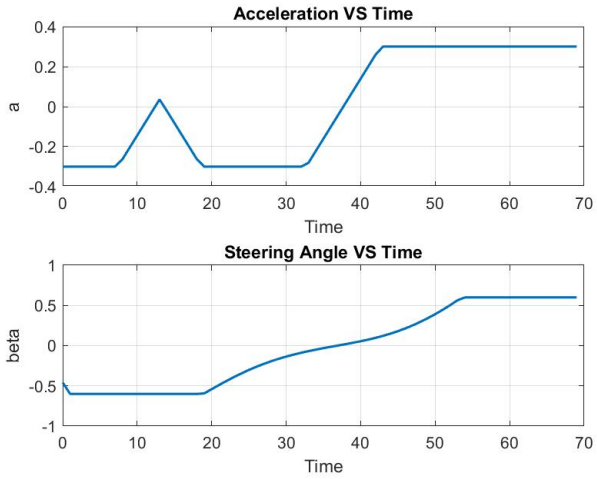

Fig 3: Input Result

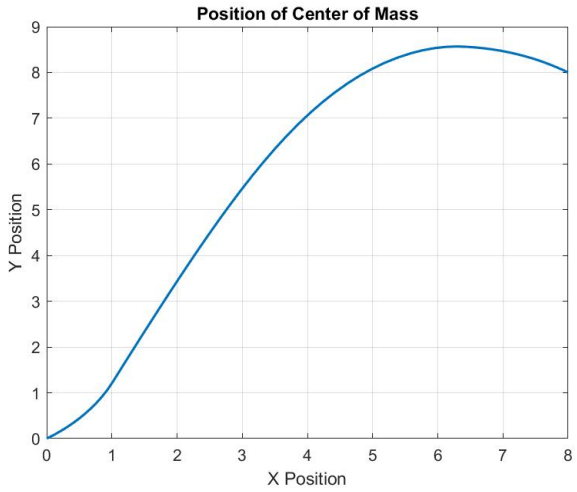

Fig 4: Car XY Position for Center of Mass

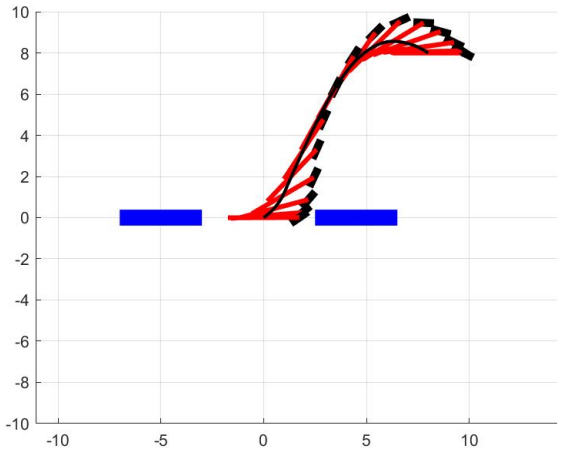

Fig 5: Car Simulation

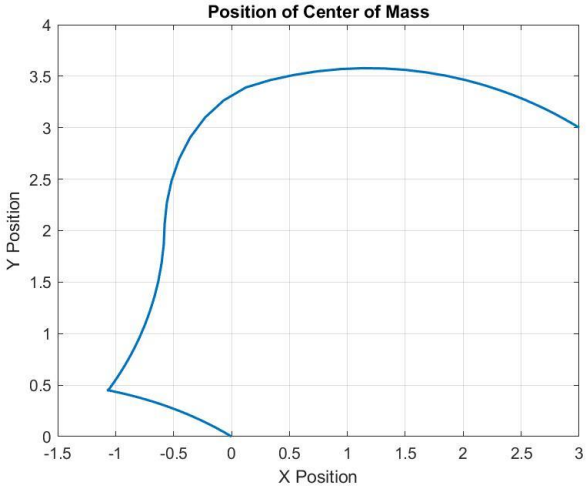

Fig 8: Car XY Position for Center of Mass

Result of Near initial condition:

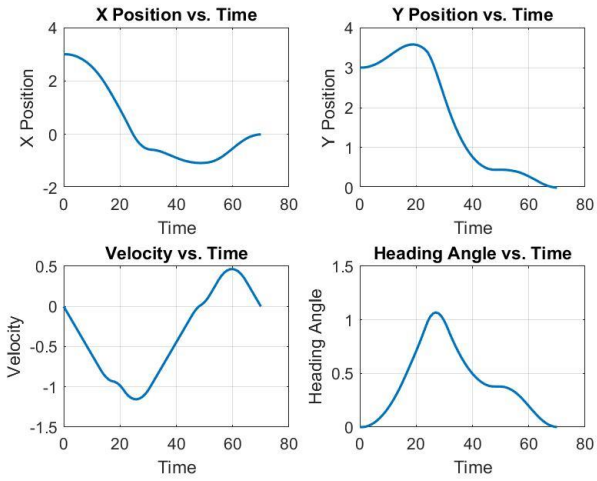

Fig 6: State Result

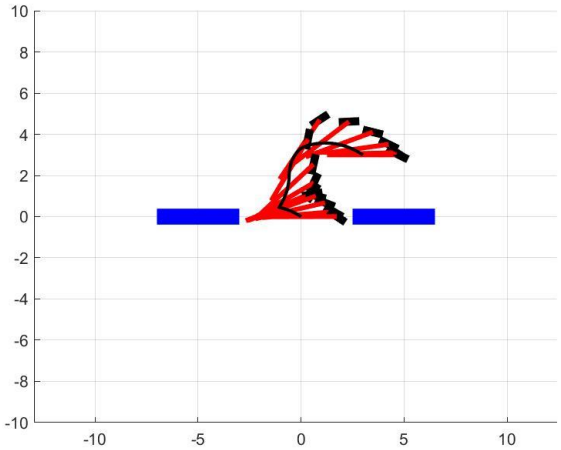

Fig 9: Car Simulation

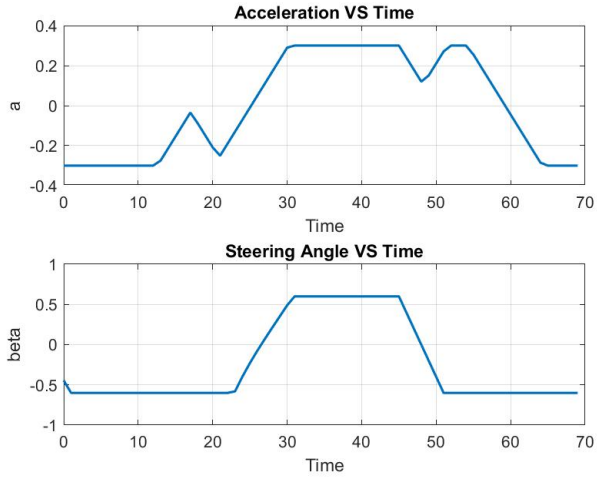

Fig 7: Input Result

Analysis on Initial Conditions:

In these two cases, the two 'blue car' is not actually imposed as a constraint. It is only shown for demonstration purposes. When the car starts relatively far away from the destination (z0=(8,8,0,0)), it will go all the way to the destination. And when the car starts relatively near the destination (z0=(3,3,0,0)), the car goes back first and then go forward to adjust position.
In both case, it is very similar to our normal parallel parking.

B. Reachability:

We also explored the reachability of the system by brute force when various speed constraint is imposed. In this case, we imposed 2 velocity constraints:

$$v \leq \pm 10 m/s$$
$$v \leq \pm 2 m/s$$

If the maximum speed of the car is low, the car cannot reach the final point in finite amount of steps. Thus, the car is expected to have a smaller reachability set if the speed limit of car is lower.

Result:

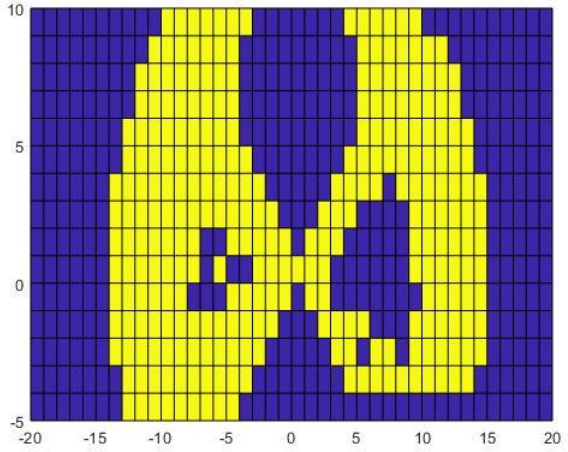

Fig 10: Reachability Vmax = 10

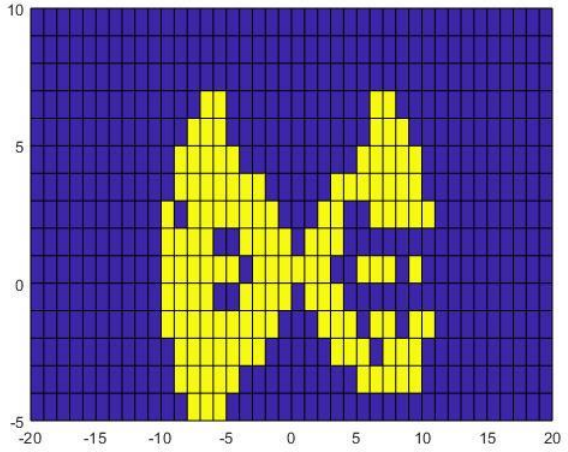

Fig 11: Reachability Vmax = 2

Reachability Analysis:

Run Time

Because this problem is not a linear problem, we cannot use analytical method to obtain the reachability set. Thus, we used a brute force method to solve this problem by repetitively solving a 41 x 16 grid across all the x, y space limit. As we expected, the run time of this problem is very high. Thus, we need to use parallel computing Toolbox in Matlab, which enable us to use multiple workers in matlab to solve this problem. In our case, we initialized 16 workers in a Xeon system. However, it still take around 1 hour for Fig 10 and 3 hour for Fig 11. We think the reason that Fig 11 takes much longer is caused by a larger infeasible set. When, the solver is solving an infeasible case, it needs to go through all the possible cases to conclude the infeasibility.

Feasible Region:

The Fig 10 and Fig 11 shows the Feasibility of the problem for all range of initial Position to reach the final point of (0, 0) mimicking parallel parking. In these cases, the car is parallel to the x axis the plot, which varies [-20, 20], and the car head is pointing to the positive direction of x axis(pointing to right). The yellow point means the problem is feasible with this initial position and max velocity, and the blue point shows the problem is not feasible. The result is behave as we expected, the range of the reachable set decreases as the max velocity of the car decreases. It means if the car start from a place that is too far, the car cannot reach the final goal in finite amount of step if its maximum velocity is too low. Also, we noticed that the car is very likely to be infeasible if the initial position contains a very small x difference between the initial position and final goal position. This is also consistent with our life experience: it is very hard to do a parallel parking if we start right beside the parking spot, which we need to drive the car forward some distance and back the car to the parking. In optimal control, the action that drive the car forward and back the car down makes the problem highly nonlinear, and making the solver very hard to find a feasible solution. Thus, most of these points are appears to be infeasible.

In fig 9 and fig 10, there seems to be places that a large infeasible region is enclosed by a ring of feasible region. We think this is caused by numerical errors, where we saw the solver complains that it almost can make this problem feasible within a difference of 10^{-6}in accuracy. We think most of the infeasible point enclosed in the yellow region is caused by such problem.

REFERENCES

[1] J. Kong, M. Pfeiffer, G. Schildbach and F. Borrelli, “Kinematic and Dynamic Vehicle Models for Autonomous Driving Control Design”

[2] J. Lofberg, “YALMIP: A toolbox for modeling and optimization in MATLAB”

[3] Ipopt home page: https://projects.coin-or.org/Ipopt