Optimization-based locally dynamic obstacle avoidance for autonomous fleet

Jinyun Zhou, Jun Zeng, Taozheng Yang, Wei Ren

***Abstract*—Avoiding obstacles becomes a real concern for autonomous driving. In this project, our goal is to utilize MPC to simulate a fleet management scenario, where a leading car follows an a predefined trajectory and avoid obstacles dynamically by minimizing steering angle, and a series of following cars following the leading car with minimal cost and state error as a whole.**

The problem can be briefly summarized as follows: Firstly, a trajectory of the leading car's is generated by MPC, while some obstacles are placed in the way and need to be taken into consideration during the path planning. To make this problem become solvable, a convexification of vehicle's obstacle constraint and minimizing the steering angle through SCP(sequential convex programming) [1] in MPC are both applied. Secondly, we simulate following cars to keep reasonable distance between itself and the leading one, while avoiding obstacles needs to be taken in account. In this part, some scenarios will be tested, such as information's time delay from the leading car. If all works as expected, we will observe the well-organized following by the second car and our results can be generalized to a fleet of cars.

Codes and videos can be found in our homepage http://www.oparp.com/car.

I. Scenario

Many previous work have been done for the avoidance of obstacles, most of them have tried to track the predefined reference with fixed obstacles. This effectiveness of this method is fragile for the sudden appearance of obstacles. In our case, we designed a local dynamic obstacle avoidance MPC. This MPC has three features, including obstacle detection, avoidance path generation and path following, respectively.

In our scenario, the truck is running along the predefined straight line. When the sudden obstacles appear, the truck will turn left or right to avoid the obstacles with steering angles as small as possible. an efficient method based on Sequential Convex programming is applied in our implementation.

To simplify our problem and decrease the computing time, We adopted the steering angles as the only input and acceleration is considered as constant in this case. Our goal is to minimize the steering angles. Since the circular obstacles are not convex, a convexification is done with linearization around the operation point. A slack variable is also introduced to allow the violation of obstacle constraint before the convergence.

The following of a series of vehicles is worth studying. As we notice the relatively high computation costs incurred during the path planning of the leading car stage, and the increasing complexity for the followers to both avoiding the static obstacles on the path as well as the moving leading vehicle, it is not practical for the following cars to carry out similar computations as the leading car. Therefore, a model predictive control algorithm was adopted to track the leading car's actual position.

II. Modeling

A. Vehicle's Model

The first step of our project is to use a simplified model to solve our problem, by considering the simplified kinematic bicycle model below, where we take coordinates of vehicle (x, y), velocity v and angle of the current velocity with respect to the longitudinal axis of the car ϕ. For the notation, we write $z = (x; y; v; \phi)$ The inputs of systems are the acceleration and the steering angle of car, noted as $u = (a; \psi)$ respectively. the turning angle is also used in the dynamical system, written as β, with the relation $\beta = arctan(\frac{l_r}{l_f + l_r} tan(\psi))$.

$$\dot{x} = v\cos(\phi + \beta), \qquad \dot{y} = v\sin(\phi + \beta),$$
$$\dot{\phi} = \frac{v}{l_r}\sin(\beta), \qquad \dot{v} = a,$$

During the implementation, the evolution of dynamical system is discretized with a sampling time Δt,

$$x_{i+1} = x_i + \Delta t \cdot v\cos(\phi + \beta), \quad y_{i+1} = y_i + \Delta t \cdot v\sin(\phi + \beta),$$
$$\phi_{i+1} = \phi_i + \Delta t \cdot \frac{v}{l_r}\sin(\beta), \quad v_{i+1} = v_i + \Delta t \cdot a,$$

and we assign it as $z(i + 1) = f(z(i), u(i), \Delta t)$ for the following description.

III. Model-Predictive Controller Formulation

A. Path planning for the leading vehicle

1) Assumptions and Formulations: We assume that N, M are the length of prediction horizon and simulation time steps. When the obstacles are not detectable in the predictable horizon, we adopt the following the MPC formulation to calculate the predictive steps.

$$obj = J_1 + J_2 + z(1)^T Q z(1) + u(1)^T R u(1) \tag{1}$$

subject to

$$J_1 = \sum_{i=1}^{N} (x^P(i+1) - x(i+1))^T Q_x (x^P(i+1) - x(i+1)),$$
$$J_2 = \sum_{i=1}^{N} z(i+1)^T Q z(i+1) + u(i+1)^T R u(i+1),$$
$$x^P(i+1) = x(i) + v(i)\cos(\phi(i))\Delta t,$$

where J_1, J_2 indicates the reference tracking to the horizon and the part of standard LQR, respectively. During the iteration, the prediction horizon reference in next step x^P is calculated by the position and velocity in the previous steps.

However when the obstacles are detectable in the prediction horizon, a slack variable ρ is introduced to make the vehicle outside the dangerous areas and here is the MPC formulation in this case.

$$obj = J_1 + J_2 + J_3 + z(1)^T Q z(1) + u(1)^T R u(1) \tag{2}$$

Advisor: Francesco Borrelli

subject to

$$J_1 = \sum_{i=1}^{N}(x^P(i+1) - x(i+1))^T Q_x (x^P(i+1) - x(i+1)),$$

$$J_2 = \sum_{i=1}^{N} z(i+1)^T Q z(i+1) + u(i+1)^T R u(i+1),$$

$$J_3 = \sum_{i=1}^{N+1} \rho_0 \rho(i), \quad \rho(i) > 0,$$

$$x^P(i+1) = x(i) + v(i)\cos(\phi(i))\Delta t,$$

$$(x^{ite} - x^{obs})(x(i) - x^{obs}) + (y^{ite} - y^{obs})(y(i) - y^{obs}) \geq (r^2 + (x^{ite} - x^{obs})^2 + (y^{ite} - y^{obs})^2)/2,$$

where r, (x^{obs}, y^{obs}) are the radius and the position of obstacles. In our problem, we assume the shape of obstacle is circular. The last constraint is local linearization of the constraint [1] and (x^{ite}, y^{ite}) is the operation point calculated in the previous step.

For all two MPC formulations, we should obey the boundary restrictions and the evolution rule of dynamical model,

$$z_{min} \leq z_i \leq z_{max}, \quad u_{min} \leq u_i \leq u_{max},$$
$$z(i+1) = f(z(i), u(i)),$$
$$\psi^{var}_{min}\Delta t \leq \psi(i+1) - \psi(i) < \psi^{var}_{max}\Delta t,$$

where ψ^{var} represents the acceleration of steering angle.

2) Algorithm and Implementation: Our pseudo code for leading car's obstacle avoidance algorithm is shown in algorithm 1. The reference tracking to the original track is done by predicting its destination at each prediction horizon and minimizing the prediction and the actual position.

Algorithm 1 Leading car's obstacle avoidance

```
1: Compute the initialization solution $z_c$ without considering any
   obstacle constraints with objective value of $J_c$.
2: if The obstacle is detected in the prediction horizon then
3:     $c = 1$
4:     Form the convexified obstacle constraints by $z_c$
5:     Compute the objective value $J_{c+1}$ with the solution $z_{c+1}$
6:     if $J_{c+1} - J_c \leq \epsilon$ then
7:         Return $z_{c+1}$
8:         Continue to next simulation time
9:     else $c = c + 1$
10:     Return to line 4
11: else Continue to next simulation time
```

B. Path planning for the following vehicles

1) Assumptions and Formulations: There are a few assumptions have been made for the MPC following part. Firstly, it is assumed that the follower is aware of the current and all historic position of its leader, the car which precedes it, which is a reasonable assumption within a relatively practical scope. Secondly, the model developed in the Modeling section is also applied to the following cars, where the acceleration and the steering angle of the cars are allowed to change abruptly. Thirdly, it is assumed that there is no moving obstacles on the path except for the preceding leading cars. Fourthly, the tracking reference of the followers are not exactly their respective leading car's position, but a modified reference position including a safety buffer that would change its relative direction to the leading car's position when the leading car is steering to avoid the obstacles. Some additional assumptions can be adjusted accordingly to different control requirements, such as whether to penalize more on the followers' tracking deviation compared from its leading car or on the followers' control inputs. Based on the assumptions made above, the general formulation of the following scenario is shown as below:

Table I: Parameters

l_f	3m	l_r	3m
a_{min}	$-1.5m/s^2$	a_{max}	$4m/s^2$
ψ_{min}	-0.6rad	ψ_{max}	0.6rad
ψ^{var}_{min}	-0.05 rad/s	ψ^{var}_{max}	0.05 rad/s
x_{min}	0m	x_{max}	100m
y_{min}	-4.5m	y_{max}	4.5m
v_{min}	$0m/s$	v_{max}	$40m/s$

$$obj = \sum_{i=1}^{N+1}(z(i) - z_{ref}(i-\delta) - z_{safe}(i))^T Q$$
$$(z(i) - z_{ref}(i-\delta) - z_{safe}(i)) + \sum_{i=1}^{N+1} u(i)^T R u(i)$$

constraints:

$$z_{min} \leq z_i \leq z_{max} \quad u_{min} \leq u_i \leq u_{max}$$
$$\beta_{i+1} - \beta i z(i+1) = f(z(i), u(i), \Delta t)$$

δ is the number of time step delay between the adjacent two cars:

$$u_i = \begin{pmatrix} a_i \\ \beta_i \end{pmatrix}$$

When $\delta > i$, we force the inputs of following vehicles to become all zeros, to ensure the their stationarity.

2) Algorithm and Implementation: This subsection illustrates the algorithm of the MPC following. Two followers are selected to illustrate the concept. The Algorithm is shown in algorithm 2.

Algorithm 2 MPC car following

```
1: procedure                                          ▷
2:     for $i \leftarrow 1$ to $N$ do       ▷ N is the total simulation length
3:                                          ▷ do the carB, the first follower
4:         for $j \leftarrow 1$ to $P$ do          ▷ P is the planning horizon
5:             $obj \leftarrow obj_i + obj$
6:         for $j \leftarrow 1$ to $P$ do
7:             $constraint \leftarrow constraint_i + constraint$
8:         solve the convex optimization problem
9:         store the first input and state
10:                                         ▷ do the carC, the second follower
11:        for $j \leftarrow 1$ to $P$ do          ▷ P is the planning horizon
12:            $obj \leftarrow obj_i + obj$
13:        for $j \leftarrow 1$ to $P$ do
14:            $constraint \leftarrow constraint_i + constraint$
15:        solve the convex optimization problem
16:        store the first input and state
```

IV. MPC Results and performance evaluation

A. Leading vehicle obstacle avoidance

During our implementation, our parameters of model comes from the Fedex truck frequently seen everyday on the road, and $L_1 = 6m$, $L_2 = 2m$, $L_3 = 2m$ are defined for the length, width, height of the truck respectively. Other choices of parameters are shown in the table I. For the configuration of MPC, Q is tuned to be lower than Q_x to have the truck not coming back to original track too quickly after

Figure 1: Complete trajectory and N-step prediction at each time

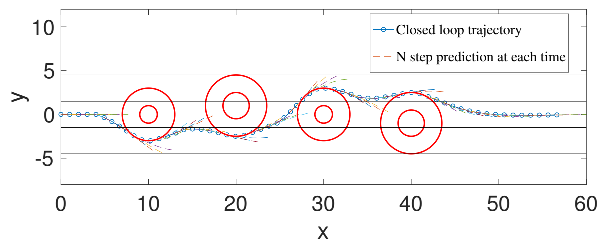
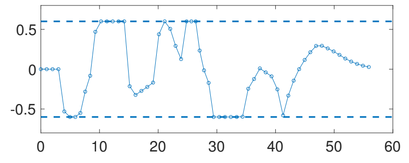


Figure 2: Steering angles for simulation



avoid an obstacle and encounters another one soon afterwards. Also higher Q_X helps avoid the obstacle better. ρ_0 and epsilon is tuned to have SCP iteration converge properly. The derivative of steering angle is limited to avoid oscillation which occasionally happens when the truck encounters an obstacle which center is on the original track.

Prediction horizon N is a key parameter in our design. If the N is too large, the computation time is too high, If the N is too small, the truck's reaction to the obstacle is too late and would cause the truck unable to avoid the obstacle clearly.

In our simulation, 60 time step with sampling time 0.2s is applied [2]. The IPOPT is used as the solver. One simulation result is shown in figure 1. In this case, $R = 1$ (reduced to single dimension since the acceleration is fixed at 0), $Q_x = 1$, $Q = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$, $\epsilon = 1$, $\rho_0 = 1000$.

When N has the value 5, 8, 11, the computational times are 1.9s, 4.2s, 9.3s for each iteration when encountering the obstacle. Computation time needs to be decreased dramatically in the future work. Some improvements are described in the later parts.

B. *Following vehicle obstacle avoidance*

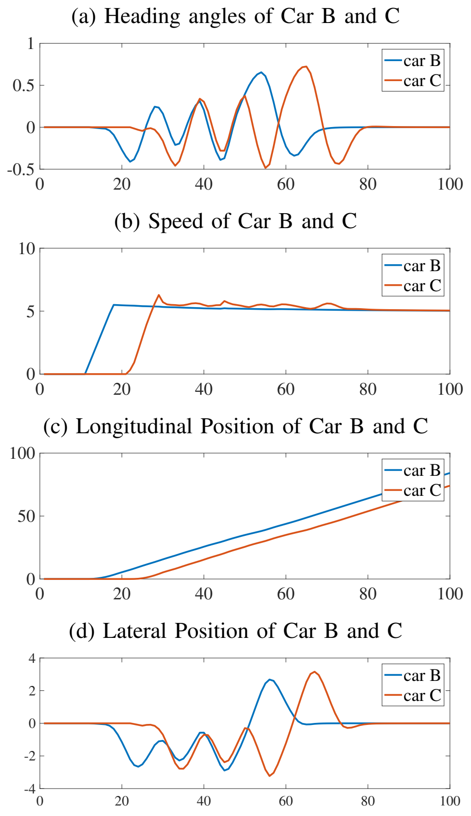
As we could see from the path planned by following car B and car C, their precise behaviors are greatly influenced by the selection of some critical parameters such as the ratio of Q and R, the magnitude and direction of safety factor, the delay time between adjacent cars. Generally speaking, the follower cars' behaviors reached our expectations, with relatively precise reference tracking as well as modest input commands (acceleration and steering angle). For the following part, we have explored the roles played by different parameters: the longer the MPC horizon, the better the performance. The bigger the penalty on the state tracking, the better the tracking preciseness. One simulation result is shown in figure 2 and time delay features can be seen through these figures.

C. *Visualization*

The MPC system has many parameters during the optimization. When designing the MPC system, it will be helpful to visualize the parameters in a real time 3D animation. This project designed a MPC system about autonomous driving. We built a 3D visualization application based on WebGL to research the results of MPC simulation.

htp[!]

Figure 3: Vehicle Following Simulation



A truck 3D model was introduced in this application. A scene with obstacle were introduced in the application. The application used data from status variables, input variables and output variables of MPC system to drive the WebGL animation loop to visualize the real-time positions, theta, and so on. The application also set the parameters of the WebGL camera to follow the car in real time. Our visualization is seen online .

V. Future Work

Due to the limited time, we still have some improvements to do in the future work. For example, the state performance of leading car MPC is good in terms of obstacle avoidance and reference tracking. But the computation time is big fallback. We were trying to implement semidefinite programming relaxation in which the quadratic term of the objective terms is substituted with a new state matrix $Z \geq z^T z$. We were also trying to directly apply the solver rather than through Yalmip.

If the computation time could be dramatically reduced, then the geometry of truck by MTP toolbox and acceleration as another input could be added to the controller.

The following performance could be improved further by modelling the car utilizing the true dimension instead of using the point as a model for the analysis.

References

[1] Bassam Alrifaee, Janis Maczijewski, Dirk Abel. Sequential Convex Programming MPC for Dynamic Vehicle Collision Avoidance, 2017 IEEE Conference on CCTA

[2] Jason Kong, Mark Pfeiffer, Georg Schildbach, Francesco Borrelli. Kinematic and Dynamic Vehicle Models for Autonomous Driving Control Design, Intelligent Vehicles Symposium (IV), 2015 IEEE