# Trajectory Tracking For Car-like Robots Based On Nonlinear Model Predictive Control ECH 267

Yuankai Zhu
ykzhu@ucdavis.edu

March 14, 2022

Instructor: Professor Matthew Ellis

**Abstract**

In this report, a nonlinear model predictive controller(MPC) for car-like robots trajectory tracking with static obstacles is presented. A car-like robot, namely a non-holonomic vehicle, has the same model(bicycle model or Ackermann model) with most vehicles on the road. Following the trajectories generated by the motion planning module with minimum deviation play an important role, otherwise the vehicle could hit the obstacles. In this project, the nonlinear MPC method is implemented in two different scenarios, one is the pure trajectory tracking while the other is the trajectory tracking with obstacle avoidance. For the scenario with static obstacles, the potential field algorithm is implemented.

## 1 Introduction

High precision path tracking is highly demanded in robot applications, such as autonomous vehicles of following the traffic rules, autonomous tractors of precision farming. The expectation of the path tracking error could be around centimeters according to the requirement. Usually, the path would be defined by a number of waypoints which contains the position and orientation information. The goal of path tracking is to minimize the average and the maximum deviation between the desired path and the actual path. The path tracking error can be defined as the shortest distance between the actual points and the desired path. There are various approaches such as pure pursuit [1], nonlinear proportional control [8], and vector pursuit [6], aiming to solve path tracking problems.

If taking the time into consideration, the first order or second order derivatives of the position or orientation, which are velocity, acceleration, angular velocity and angular acceleration accordingly, it would be the trajectory tracking problem. In trajectory tracking problems, the desired trajectories must be provided to the controller at each time sample. Then, the tracking error is defined as

the difference between the planning and actual trajectory points. For trajectory tracking problems, the most popular control method is the PID(proportional, integral, derivative) controller which is well developed in industry. Other control techniques can be also implemented, such as LQR [2], and MPC.

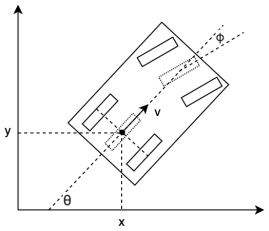
Furthermore, the uncertainty in the real world affects the behaviors of the robots. The upstream module of the controller collects the information of the real world and localization on map to generate trajectories. However, if there are unprecedented obstacles appearing while actuating controllers, the online modification for the pre-planned trajectories should be applied based on the sensor data. In this project, the real-time iterative MPC would be implemented. The MPC controller minimizes the total tracking errors along the entire receding horizon motion planning. The pure pursuit [1] and the path deformation [4] are special cases, and the pure pursuit will be a benchmark. The simulation is also implemented and tested in different scenarios, including those with obstacles.

The second section models car-like robots. Then, the third section interprets the MPC method. Next, the fourth section provides the simulation and the results. The final section discusses the results and potential future work.

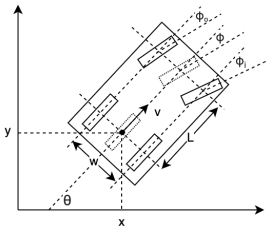
## 2 System Modeling

The car-like robots are usually non-holonomic. The definition of *holonomic* is related to the number of control inputs and the total degree of freedom(DOF): if the number of inputs are the same of the DOF, it's called *holonomic* system; if the number of inputs are fewer than the DOF, the system is *non-holonomic*. For car-like robots, the inputs are usually steering and the moving along the direction of the wheels, while the DOF is 3(longitudinal, lateral direction and orientation). Therefore, it is a non-holonomic system.

The typical models for car-like robots are the bicycle model and the Ackermann model. Both of the models take the four-wheel system equivalent to two virtual center wheels. The difference between these two models is how to model the steering wheels. For simplicity, take the center of the rear wheel base as the reference point. In the bicycle model, the four-wheel car is shrunk to two-wheel bicycle as the figure (1a) shows. However, due to the centers of the turning circle of the two steering wheels are different in the bicycle model, there will be sliding in reality. In this case, the Ackermann model is more popular in car-like robots. The idea of the Ackermann model is to ensure the center of the turning circle of two steering wheels are at the same position. To guarantee that, the steering angles of two steering wheels are different, that is, the inner wheel has the larger steering angle. The basic kinematic model of the car-like robots are as follows:




(a) Bicycle model. Shrink four wheels into two wheels.




(b) Ackermann model. Front wheels steer in different angles.

Figure 1: Bicycle model and Ackermann model

$$\dot{x} = v\cos\theta \tag{1}$$

$$\dot{y} = v\sin\theta \tag{2}$$

$$\dot{\theta} = v\frac{\tan\phi}{L} \tag{3}$$

The $x$ and $y$ coordinates is the position of the car's rear wheel base midpoint $P = [xy]^T$. The unit vector $v$ starts at $P$ and lies along the direction of the motion. The $\theta$ reflects the orientation and the $\phi$ is the steering angle.

In the Ackermann model, which has the similar kinematic equation as (1), (2) and (3). The difference is that, it defines a $r$ which is the turning radius of the virtual center wheel. Then, the relationship between three steering wheels(inner one, virtual center one, and the outer one) are following:

$$\tan\phi = \frac{L}{r}, \tan\phi_i = \frac{L}{r - w/2}, \tan\phi_o = \frac{L}{r + w/2}$$

Due to the fact that, if the steering angle of the virtual center wheel is determined, the angle of the inner and outer steering wheel can be calculated by the previous equations, for simplicity, only take the steering angle of the virtual center wheel into consideration as the state of the system.

At the low level of the controller system, the steering angle and the speed would decrease to zero. Such that, it can be abstracted as a first order model with the following equations with time lags constant [7]:

$$\dot{\phi} = -\frac{1}{\tau_\phi}\phi + \frac{1}{\tau_\phi}u_\phi \tag{4}$$

$$\dot{v} = -\frac{1}{\tau_v}v + \frac{1}{\tau_v}u_v \tag{5}$$

Now the car-like robot system can be described by (1) to (5), and the system states are $\mathbf{x} = [x, y, \theta, \phi, v]^T$, and the control inputs are $\mathbf{u} = [u_\phi, u_v]^T$. From the

state space model (1) to (5), if the inputs are all zeros, the equilibrium points of the system is $[x, y, 0, 0, 0]^T$. Obviously, the system is not I.S.S, because with a small input $u_v$ and zero $u_\phi$, the state $x$ and $y$ can reach infinity. However, the inputs are bounded with:

$$-2\pi/3 \leq u_\phi \leq 2\pi/3 \tag{6}$$

$$0 \leq u_v \leq 1 \tag{7}$$

# 3 Optimal Control And MPC

Compared to study of other nonlinear MPC problems, trajectory tracking has astonishing sparse results. Since the trajectory tracking usually should be implemented in real time, one of the reasons for sparse results is that the MPC calculation takes too much time compared to other light methods like PID with pure pursuit, even though other methods lose quite a bit of optimality. However, for autonomous vehicles on the road or tractor for precise farming, the optimality plays a more important role, such that the computing time can be reduced by increasing the computing power. In this case, with the rise of the autonomous vehicle industry, MPC in trajectory tracking would be increasingly taken into consideration.

In the trajectory tracking problem, rather than states in system (1) to (5), the error between the reference trajectory waypoints and the actual trajectory would be taken into consideration as the cost which is expected to be minimized. In general case, the optimization cost function at time step $k$ is:

$$\min J = \mathbf{e}_{k+M+1}^T Q_e \mathbf{e}_{k+M+1} + \sum_{i=k}^{\min(N,k+M)} \mathbf{e}_i^T Q_e \mathbf{e}_i = \sum_{i=k}^{\min(N,k+M+1)} \mathbf{e}_i^T Q_e \mathbf{e}_i \tag{8}$$

where $N$ is the total number of waypoints in the trajectory, $M$ is the length of the horizon receding, and the $k$ is the current simulation time step,

$$\mathbf{e}_i = \min(\mathbf{x}_i - \mathbf{x}_{ref})$$

$\mathbf{x}_{ref}$ is the planning trajectory, which is defined by waypoints. The minimum here means the shortest deviation from the reference trajectory. For example, given a path, which only contains the position information, the shortest deviation is the shortest distance from the current position to the reference path. Such that, it is a time-variant state. That's another reason why the trajectory tracking or even the path tracking problem is difficult to solve [3]. For simplicity, a straight line with infinite length is given as the reference path:

$$x - y + 1 = 0$$

such that, a set of errors $\mathbf{e} = [e_1, e_2, e_3, e_4]$ can be given:

$$e_1 = |x - y + 1|/\sqrt{2} \tag{9}$$
$$e_2 = \theta - \pi/4 \tag{10}$$
$$e_3 = \phi \tag{11}$$
$$e_4 = v - 0.5 \tag{12}$$

where $e_1$ reflects the distance from the current position $(x, y)$ to the line $x - y + 1 = 0$; $e_2$ reflects the heading error between the current heading $\theta$ and the desired heading $\pi/4$ which is the heading of the line $x - y + 1 = 0$; $e_3$ reflects the error between the current steering angle $\phi$ and the desired steering angle which is 0; $e_4$ reflects the speed error between the current speed $v$ and the desired speed $0.5m/s$.

The pure trajectory tracking problem can be defined as the cost function (8) with constraints (1) to (7) and (9) to (12). The implementation in Python can be found in the attachment.

In this project, the potential field algorithm [5] is implemented to practice obstacle avoidance. By achieving this, a cost related to the obstacle information would be added into the cost function (8):

$$\left(\frac{C}{D(x,y) + \epsilon}\right)^{\rho}$$

where $C > 0$, $\epsilon$ is small and $> 0$, and $\rho > 1$, $D(x, y)$ is the distance between the current position $(x, y)$ and the obstacles. For simplicity, in this project, the obstacle is considered as a point with no area at $(1, 2)$. Such that, with the potential field to the obstacle, the optimization problem can be defined as the cost function:

$$\min J = \sum_{i=k}^{\min(N,k+M+1)} \left(\mathbf{e}_i^T Q_e \mathbf{e}_i + \left(\frac{C}{\sqrt{(x_i - 1)^2 + (y_i - 2)^2} + \epsilon}\right)^{\rho}\right) \tag{13}$$

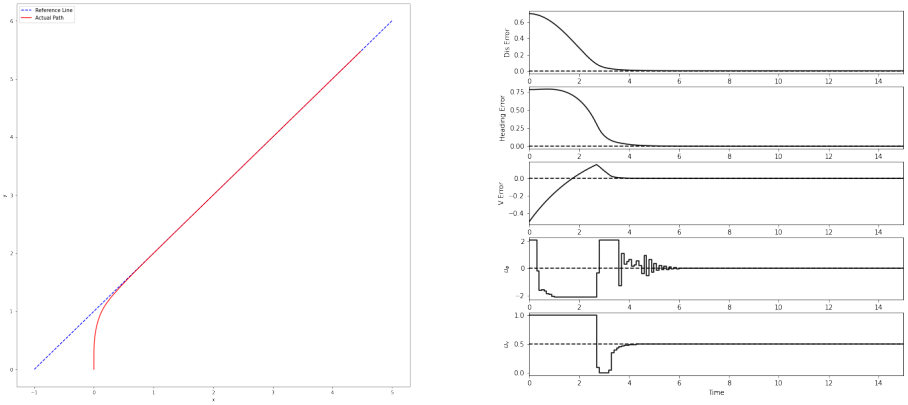
with constraints (1) to (7) and (9) to (12).

# 4 Simulation Result And Discussion

For simulation, some values should be defined: the MPC sampling period $dt = 0.1s$, total number of waypoints $N = 300$, the receding horizon length $M = 25$, $C = 1$, $\epsilon = 0.01$ and $\rho = 2$. In trajectory tracking problem, the significance of the path error, heading error and the velocity error decreases respectively and the steering error does not matter. Such that, the penalty in the cost function can be defined as

$$Q_e = \begin{bmatrix} 1000 & & & \\ & 100 & & \\ & & 0 & \\ & & & 50 \end{bmatrix}$$

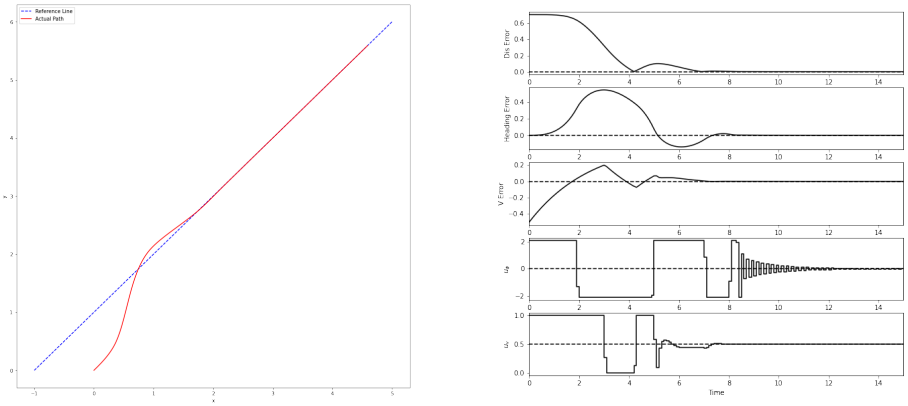
In two different scenarios, one is obstacle free and the other one is with obstacle, the robot all start from the origin $(0, 0)$ with zero steering angle and the speed. The only difference between the initial conditions are the heading angles. The values of the heading angle are $\pi/2$, $\pi/4$ and 0 respectively. The path tracking results and the errors versus optimal inputs are in following figures:




(a) Heading north

(b) Heading north inputs

Figure 2: Start from heading north without obstacles.



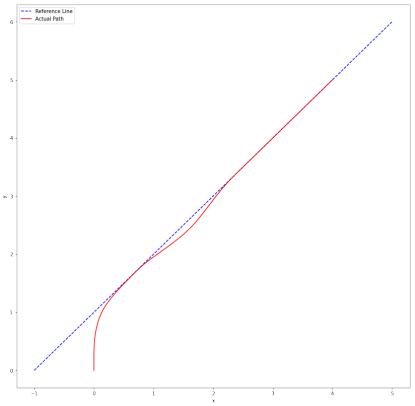

(a) Heading northeast.

(b) Heading northeast

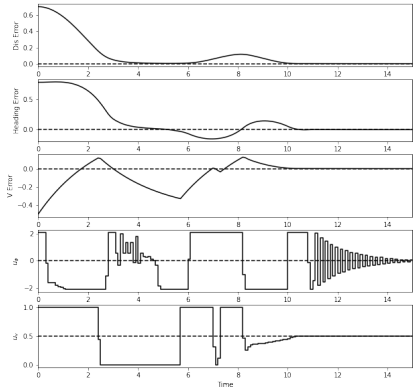
Figure 3: Start from heading northeast without obstacles.

In the pure trajectory tracking scenario, the robot returns to the reference line quickly by increasing the speed and turning to the reference line. While approaching to the reference line, the robot reduces the speed and turning to the desired heading direction.

How about the behaviors with obstacles? The results are as follows:

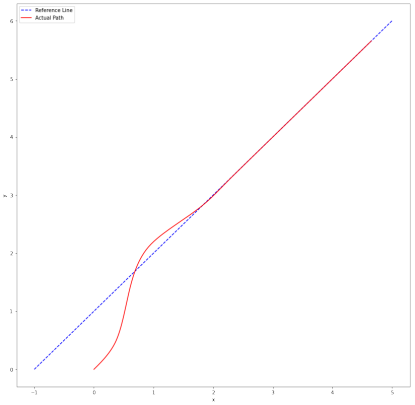



(a) Heading north.

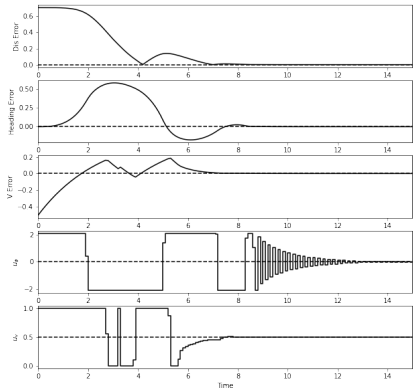



(b) Heading north inputs.

Figure 4: Start from heading north with obstacles.




(a) Heading northeast.




(b) Heading northeast

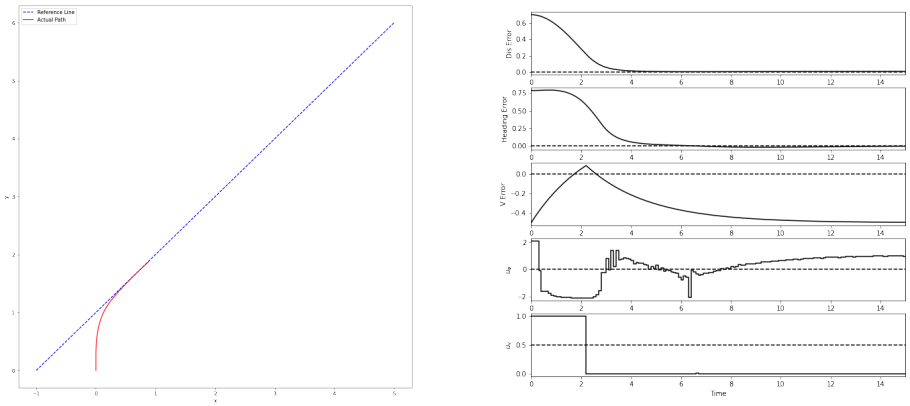
Figure 5: Start from heading northeast with obstacles.

With the obstacle on the path, the robot can turn and avoid the obstacle at position $(1, 2)$ as the figures show. Comparing figure 3(a) and figure 5(a), the path deviates from the reference line more in figure 5(a) due to the impact of the obstacle. Also note the stabilization of the steering angle after passing the obstacle. It shows that the inputs could be stabilized.

Why is there no result for heading east? It failed to calculate the results. The possible reason is that, the searching area of heading east is much larger than previous initial conditions (heading north and heading northeast). Due to the capacity of the solver, the result of starting from heading east cannot be obtained in this project.

An interesting case as shown in the following figure is that, if increasing

the penalty of the cost of the obstacle, no matter increasing the value of $C$ or increasing the horizon receding $M$, the robot would stop in front of the obstacle with the zero speed. Increasing the penalty of the speed error does not push the robot to move. A possible reason is that, the obstacle cost increases the non-convexity of the optimization problem. Such that, the algorithm got stuck in the local minimum or it would take more computing power to escape the local minimum.




(a) Stop in front of the obstacle.

(b) Inputs.

Figure 6: Get stuck in the local minimum.

# 5 Conclusion And Future Work

In this report, in order to solve the optimization problem in a limited time, the problem is simplified in many parts. Here are the potential improvement for this project:

a. The nonholonomic car-like robot model is simplified as the kinematic model. For accuracy, the dynamic model should be taken into consideration.

b. The reference path is simplified as a straight line. In practice, the planning trajectories could be any feasible smooth curve with various linear and angular speed, acceleration, or even jerk. However, increasing the complexity of the reference trajectory would increase the difficulty of the calculation of errors.

c. The obstacle cost is simplified by ignoring the shape and area of the obstacle. However, changing the expression of the potential field could cause the algorithm to be stuck or even failure to compute a result. In this case, in addition to the result that the robot stops in front of the obstacle,

nonlinear MPC is less capable to solve the obstacle avoidance problem. In fact, the obstacle avoidance is usually implemented in the upstream module which is the planning module. The MPC is utilized to follow the trajectory precisely.

d. This project is based on CasAdi and python3 in Jupyter Notebook. CasAdi is a software that could solve optimization problems but is very slow. For trajectory tracking problems, it is easy to fail by changing the parameters or the initial condition. For future work, better computing power and better software could be utilized to solve such problems.

e. As mentioned above, the pure pursuit + PID method is popular in industry. In future work, this method could be implemented as the benchmark.

# References


[1] O. Amidi. Integrated mobile robot control. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST, 1990.

[2] A. W. Divelbiss and J. T. Wen. Trajectory tracking control of a car-trailer system. *IEEE Transactions on Control systems technology*, 5(3):269–278, 1997.

[3] T. Faulwasser. *Optimization-based solutions to constrained trajectory-tracking and path-following problems*. PhD thesis, Magdeburg, Universität, Diss., 2012, 2013.

[4] F. Lamiraux, D. Bonnafous, and O. Lefebvre. Reactive path deformation for nonholonomic mobile robots. *IEEE transactions on robotics*, 20(6):967–977, 2004.

[5] S. G. Vougioukas. Reactive trajectory tracking for mobile robots based on non linear model predictive control. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3074–3079. IEEE, 2007.

[6] J. Wit, C. D. Crane III, and D. Armstrong. Autonomous ground vehicle path tracking. *Journal of Robotic Systems*, 21(8):439–449, 2004.

[7] D. Wu, Q. Zhang, J. Reid, H. Qiu, and E. Benson. Model recognition and simulation of an e/h steering controller on off-road equipment. *Fluid Power Systems and Technology*, 1:55–60, 1998.

[8] Y. Zhang, S. Velinsky, and X. Feng. On the tracking control of differentially steered wheeled mobile robots. 1997.