Nonlinear Model Predictive Control for Mobile Robots' Path Optimization to Hybrid-A* Solution in Unstructured Environment

Group 34
Ma, Runyu Cao, Yongxi Chen, Jiaxuan Ren, Lambert

***Abstract*— In this project we present a solution to constrained optimization of a mobile robot with nonlinear Model Predictive Control (MPC) in an unstructured environment. A nonlinear kinematics bicycle model is deployed. The MPC controller optimizes over a kinematics feasible path we pre-calculated with Hybrid-A*. A simple strategy is used to describe dynamic obstacles as constraints in MPC. We demonstrate that the nonlinear MPC can both optimize regarding control effort and path length over the pre-obtained path, and avoid dynamic obstacles which are not considered in the Hybrid A* phase.**

I. INTRODUCTION

The task of parking cars in unstructured environments is practical and challenging. Parking can be formulated into a very standard path planning problem with dynamic obstacles. This problem is often resolved in two steps, a) kinematic feasible path generation, and b) constrained optimization for optimal path [1],

a) Kinematic feasible path generation: A global feasible path that takes non-holonomic property of a mobile robot into consideration is generated for reference;
b) Constrained optimization: a local planner optimizes the obtained path regarding some given objective and constraints.

Considering the relatively low dimension of configuration space and potential of multi-querying, we use a motion primitive method in step a. More specifically, hybrid A* is selected. This step is necessary for two reasons, a) although the path generated is sub-optimal, it is always around the global optimality, and b) constrained optimization alone can be time consuming when faces with many obstacles, and even fail in many cases. For step b, we choose the a generalised constrained optimization method, Model Predictive Control (MPC). The aim of using MPC are to minimize some cost (control effort, time, path length, etc.), as well as to formulate dynamic obstacles as constraints and perform avoidance. The focus of this work is on the MPC phase.

There are two main challenges underlying the method. Firstly, unlike scenarios such as lane keeping, the parking lot is an unstructured environment. This implies large state space and introduces problems in describing constraints, especially dynamic ones. Moreover, in order to generate more appropriate trajectory and tackle dynamic obstacles, the vehicle model should at least consist of two control inputs, velocity and steering angle. This leads to the nonliearity of the bicycle model required [2]. The formulated optimization problem also becomes nonconvex thereby.

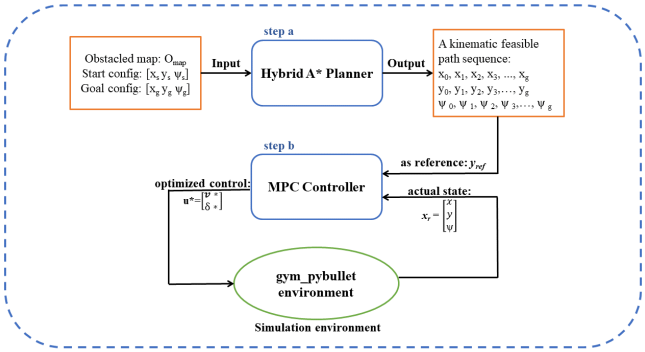

Fig. 1. Overall pipline of the project

To adapt to the complex unstructured environment, we describe all static obstacles in the Hybrid A* phase, and propose a simple strategy to formulate dynamic obstacles as constraints in the MPC phase. To take nonlinearity into consideration, a nonlinear kinematic bicycle model is deployed, and a nonlinear optimization toolbox, `ACADOS` [3], is used for implementations.

The overall pipeline of our work is summarized as in 1, and contributions of this work are listed below:

- the implementation of a nonlinear MPC of a kinematic bicycle model that is capable of path optimization and dynamic obstacle avoidance,
- adaptation of a Hybrid A* method to generate global kinematical feasible path for parking,
- construction of a decent scale parking-lot scenario in `pybullet` to test the implemented method

II. ROBOT MODEL

A nonlinear kinematic bicycle model is utilized in MPC. Although a simpler linear model with only steering angle as control input is often favorable, it cannot control the longitudinal velocity. This makes the resulting trajectory less natural at varying radius arcs, and it cannot resolve dynamic obstacles in many cases. Therefore, we choose a model that 2-dimensional control space, to control both front wheel steering angle and driving speed.

The state space model of kinematic bicycle model for steering of the vehicle is briefly summarized as follows,

Assumptions and limitations:

- no longitudinal / lateral load transfer
- linear range tires
- no roll, pitch or vertical motion

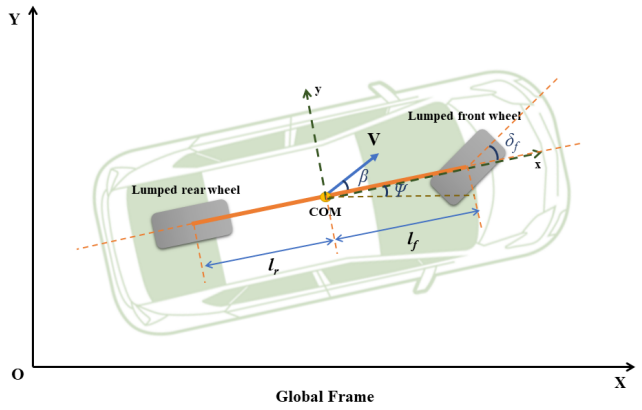

Fig. 2. Kinematic Bicycle model

- "ideal" steering dynamics
- no suspension
- no compliance effect

We use an extended state of 5 dimensions, representing the pose of the vehicle in the global frame $[X, Y, \theta]$, vehicle velocity V and steering angle δ,

$$\boldsymbol{x} = \begin{bmatrix} X \\ Y \\ \theta \\ V \\ \delta \end{bmatrix} \tag{1}$$

The control input is defined as the overall acceleration of the vehicle a and the changing rate of the steering angle $d\delta$.

$$\boldsymbol{u} = \begin{bmatrix} a \\ d\delta \end{bmatrix} \tag{2}$$

The slip angle at the center of mass (COM) is calculated as,

$$\beta(\delta) = arctan(tan(\delta)\frac{l_r}{l_r + l_f}) \tag{3}$$

The transition model is then written as,

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\theta} \\ \dot{V} \\ \dot{\delta} \end{bmatrix} = \begin{bmatrix} V cos(\theta + \beta(\delta)) \\ V sin(\theta + \beta(\delta)) \\ \frac{V}{l_r} sin(\beta(\delta)) \\ a \\ d\delta \end{bmatrix} \tag{4}$$

The formula (4) only involves terms in **x** and **u** and thus is a complete transition model. We define it as $\dot{x} = f(x, u)$ thereafter. One should notice that the nonlinearity is introduced in the transition of $\dot{X}$ and $\dot{Y}$, where V is multiplied with triangular functions.

III. Motion Planning

Model predictive control combined with Hybrid-state A* search is used to fulfill the goal of solving a problem of obstacle avoidance and trajectory optimization in semi-structured environment.

Inspired by *Dolgov, D. et al*[4], the motion planner is divided in 2 parts. The first part is a **motion primitives** method that generate kinematically feasible path in open space. The second part is an optimization-based method, more specifically, model predictive control (**MPC**), to improve the quality of the previous path and calculate the velocity of the

A. Path Planner

The hybrid-state A* algorithm is a modification of the traditional A* algorithm. It divides the search space into a grid and creates a graph using the centers of each cell as neighbors. However, it also associates each grid cell with a continuous 3D state of a vehicle.[4]

Algorithm 1 Hybrid A* Search

```
function RoundState(x)
    x.Pos_X = max{m ∈ ℤ | m ≤ x.Pos_X}
    x.Pos_Y = max{m ∈ ℤ | m ≤ x.Pos_Y}
    x.Ang_θ = max{m ∈ ℤ | m ≤ x.Ang_θ}
    return x
end function

function EXISTS(x_succ, 𝒞)
    if {x ∈ 𝒞 | roundState(x) = roundState(x_succ)} ≠ ∅ then
        return true
    else
        return false
    end if
end function

Require: x_s ∩ x_g ∈ X
O = ∅
C = ∅
Pred(x_s) ← null
O.push(x_s)
while O ≠ ∅ do
    x ← O.popMin()
    C.push(x)
    if roundState(x) = roundState(x_g) then
        return x
    else
        for u ∈ U(x) do
            x_succ ← f(x, u)
            if ¬exists(x_succ, C) then
                g ← g(x) + l(x, u)
                if ¬exists(x_succ, O) or g < g(x_succ) then
                    Pred(x_succ) ← x
                    g(x_succ) ← g
                    h(x_succ) ← Heuristic(x_succ, x_g)
                    if ¬exists(x_succ, O) then
                        O.push(x_succ)
                    else
                        O.decreaseKey(x_succ)
                    end if
                end if
            end if
        end for
    end if
end while
return null
```

Fig. 3. pseudocode of Hybrid A* Search Algorithm

The search begins by linking the current state of the vehicle to the initial search node. When a node is removed from the open list, it is expanded by applying different steering actions to the continuous state associated with the node, and new children states are created using a kinematic model of the vehicle. For each of these continuous children

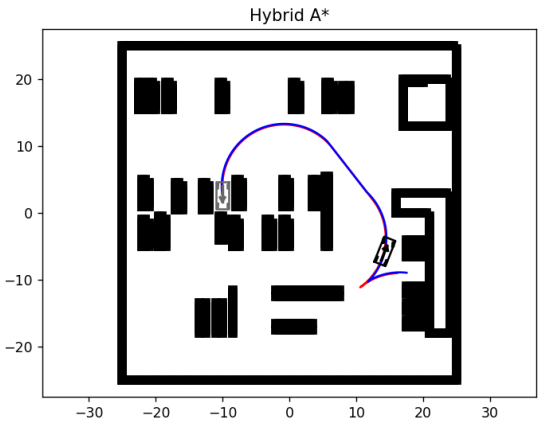

Fig. 4. Hybrid A* Search applied in parking scene example.

states, the algorithm calculates the corresponding grid cell, and if a node with the same grid cell is already present on the open list and the new cost is lower, the continuous state of the node is updated and the node is re-prioritized on the open list.

The pseudocode of Hybrid A* is shown in figure 3. Figure 3 is a demonstration of applying the Hybrid A* algorithm in a parking lot scene example, where a path is successfully constructed without hitting any of the obstacles.

1) Heuristics: Hybrid A* uses two heuristics to make path planning. The first heuristic, which we call non-holonomic-without obstacles, ignores obstacles but takes into account the non-holonomic nature of the car. The second, holonomic-with-obstacles heuristic is a dual of the first in that it ignores the non-holonomic nature of the car, but uses the obstacle map to compute the shortest distance to the goal by performing dynamic programming in 2D.

2) Analytic Expansions: The Reed–Shepp path is used to improve search speed and precision. In forward search, a node in the tree is expanded by simulating a kinematic model of the car (a particular control action) for a short duration of time (resolution of the grid map). The Reed–Shepp then evaluated for any collisions against the current map of obstacles, and it is only added to the tree if the path does not have any collisions.

B. Model Predictive Control

Model predictive control (MPC) is an advanced method of process control that is used to control a process while satisfying a set of constraints, and it has already been widely used in the control of self-driving cars [5]. MPC models are usually used to predict the change in the dependent variables of the modeled system that will be caused by changes in the independent variables. In this project, the MPC is used to guide the robot to follow the path generated by the hybrid A* algorithm. Moreover, it is also used to guide the robot to avoid the unexpected obstacle during the self-driving process.

The nonlinear model predictive control is selected for this project as the iterative solutions of optimal control problems

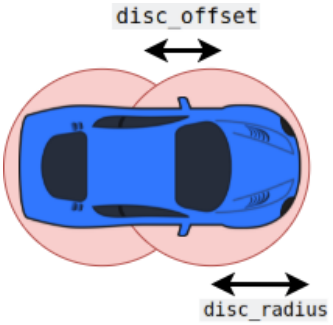

Fig. 5. Vehicle collision model.

are not necessarily convex anymore [6]. For this reason, this control scheme will work better in the current playground as a reversing process seems like a better solution intuitively.

1) Nonlinear MPC formalization: The model predictive control problem is formalized as a Lagrange cost term, which has the form

$$l(\mathbf{x}, u, z) = \frac{1}{2}\|V_x\mathbf{x} + V_u u + V_z z - y_{ref}\|_W^2 \tag{5}$$

where matrices $V_x \in \Re^{n_y \times n_x}$, $V_y \in \Re^{n_y \times n_u}$ and $V_z \in \Re^{n_y \times n_z}$ map x, u and z onto y, respectively. And $W \in \Re^{n_y \times n_y}$ is the weighing matrix. The vector $y_{ref} \in \Re^{n_y}$ is the reference [3].

Since the playground here is in 2D, the problem is finally formalized as follow:

$$\begin{aligned} \min_{x,u} \sum_{k=0}^{N} l_k(\mathbf{x}, u) = \sum_{k=0}^{N} \frac{1}{2}\|V_x\mathbf{x}_k + V_u u_k - y_{ref,k}\|_W^2 \\ s.t. \dot{\mathbf{x}} = f(\mathbf{x}, u) \\ A\mathbf{x} \leq b \\ -5 \leq v \leq 10 \\ -0.5 \leq \delta \leq 0.5 \\ -0.6 \leq \dot{\delta} \leq 0.6 \\ -1 \leq \dot{v} \leq 1 \end{aligned} \tag{6}$$

where V_x and V_u is identity matrix, and δ is the steering angle. f is the ode function of kinematic bicycle model. A and b describes the hyperplane of the constraints of dynamic obstacles.

2) dynamic obstacle avoidance: The collision model for the vehicle is shown in figure 5. It is assumed that the vehicle is consist of 2 discs with radius r and the distance between the COM of the car and discs is d.

As solving hybrid A* algorithm is quite time-consuming , it is unable to re-plan the path globally when a moving obstacle is observed. Obstacle constraints must be formalized in the local planner. To formulate a convex constraint in MPC, which is relatively easy to solve, obstacle is linearized locally and hyperplane is used to segment the space. This hyperplane in R^2 is calculated based on the state of robot, parameter of vehicle collision model and the radius of obstacles, which is shown in figure 7. This line is perpendicular to the line from

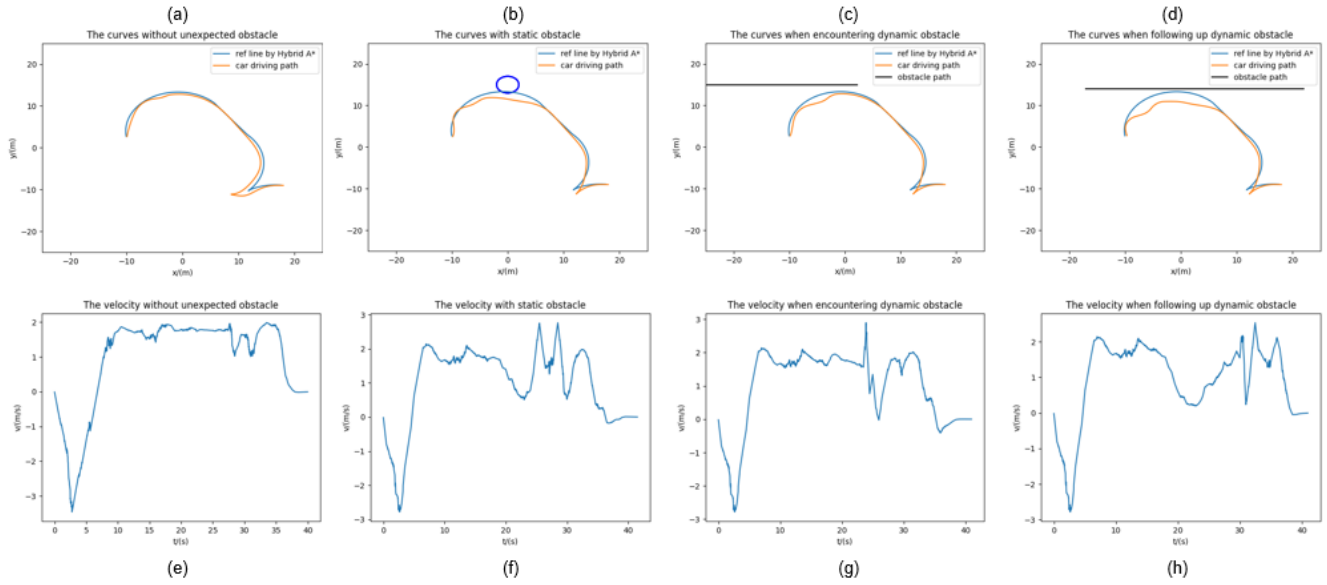

Fig. 6. (a) The driving path of the car comparing with the reference line generated by Hybrid A* when there is no unexpected obstacle on the route. (b) The driving path of the car when there is a static obstacle at the top of the curve. (c) The driving path of the car when encountering a dynamic obstacle. (d) The driving path of the car when following up a dynamic obstacle. (e) The car's velocity when there is no unexpected obstacle. (f) The car's velocity velocity when there is a static obstacle. (g) The car's velocity when encountering a dynamic obstacle. (h) The car's velocity when following up a dynamic obstacle.

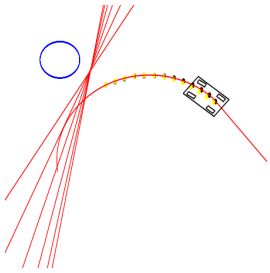

Fig. 7. Hyperplane in R^2.

obstacle center position to the reference robot position. The distance from the hyperplane to obstacle center is the sum of disc radius and obstacle radius. Thus the inequality can be formulated as :

$$k_x x_{disc} + k_y y_{disc} + k_c <= 0 \tag{7}$$

To avoid collisions, position of centers of 2 discs should meet the requirements above.

$$\begin{aligned} x_{disc,0} &= x + dcos(\theta) \\ y_{disc,0} &= y + dsin(\theta) \\ x_{disc,1} &= x + dcos(\theta) \\ y_{disc,1} &= y + dsin(\theta) \end{aligned} \tag{8}$$

IV. Results

We have tested our code for both the path planner and the model predictive control in the playground shown as figure 8. The lower right part where a car is parking is the initial place of the car, and the upper left part , the other end of the green line is the destination of our car. The other rectangular

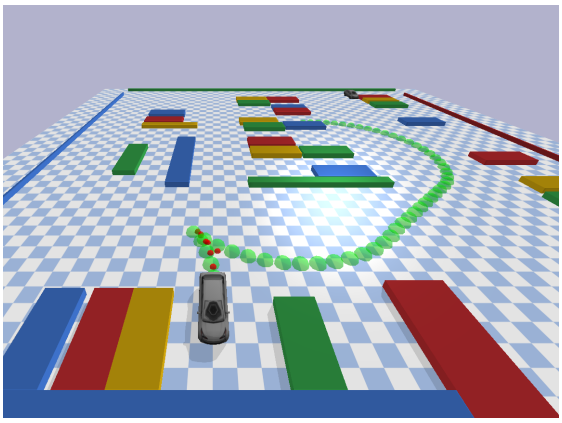

Fig. 8. Playground in gym and pybullet framework simulation environment.

blocks with various colors are obstacles we would like to avoid.

The green dotted line in figure 8 is the ideal path planned by the path planner hybrid A* algorithm. As shown in figure 6(a), the nonlinear MPC is able to roughly follow the ideal path and get to the destination, including both the reversing and parking process. Then we add a static car as an obstacle to our car model, which is unexpected for our path planner, and as shown in figure 6(b) and (f), our car is able to turn left and decelerate timely to avoid running into the obstacle.

To further validate the feasibility and robustness of our model predictive control module, we would also like to test our model in another two situations where there is a dynamic unexpected obstacle occurred in the playground, respectively encountering another car and following up another car. For both of the dynamic situations, the unexpected car will be at

the top of the curve shown as figure 8 almost synchronously as our car model, the path of which shown in black line in figure 6(c) and (d). And the velocity of this dynamic obstacle is set to $1m/s$. For the encountering situation, the car will sharply turn left and decelerate before running into the obstacle as shown in figure 6(c) and (g). For the following up situation, the dynamic obstacle's place is set to a much lower place which is exactly the same height as the top of the reference line generated by hybrid A*. And as shown in figure 6(d), our car is able to turn left much more obviously comparing with the previous two situations. And the car is also able to decelerate nearly to 0 to wait for the obstacle's going first as shown in figure 6(h). This means that our nonlinear MPC model is able to assist our car to implement some simple obstacle avoidance functions.

V. DISCUSSION

A. Method chosen for the hyperplane

As the previous section shows, we select the hyperplane based on the perpendicular line of the line from obstacle center position and the reference position. However, there are some problems with this method.

For the last situation discussed in the result section, which is the situation when our car is following up a dynamic obstacle, there is some particular time that the reference position of our robot is far beyond the place of itself. Therefore, the reference position locates to the left of the dynamic obstacle. And this will result in classifying our car model and the dynamic obstacle to the same side of the hyperplane, which is not we are expecting. Maybe there is a better way to decide the hyperplane to classify the obstacle and the safe destination for the robot.

B. The horizon chosen for the nonlinear MPC

When tuning the parameters for the nonlinear MPC, horizon is a rather important one having influence on the performance of our self-driving car. When setting the parameter too small, the car will not timely react to the obstacle especially when it is running at a much higher speed since the car cannot forecast its future position for a much longer time. Considering that a car generally has a rather high inertia, the car probably will run into the obstacles. When setting the horizon too high, there will be some problems combined with the problem discussed in the previous section as a long horizon may result in a defective hyperplane. Moreover, during the experiment, it is obvious for us that a higher horizon will result in a slower running of the code. Therefore, maybe there is a still better horizon for our nonlinear MPC model.

C. Some problem with the scene without obstacle

As shown in figure 6(a)-(d), there is some obvious difference between them during the reversing process. For the scene that there is no obstacle, the car goes further and the speed is much higher than the other 3 plots, and it nearly runs into the block. This is the problem we are extremely confused about, since that the obstacle seems to have no influence on the control part of the reversing process, and there is no parameter difference between the two situations. Moreover, it proposes us a further question, that is whether it is necessary for us to take this into consideration during the hybrid A* process. It seems that if the hybrid A* is further optimized to leave a more space for the car reversing process or the nonlinear MPC takes this specific block into consideration, there will be a safer path for our self-driving car.

REFERENCES

[1] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Practical search techniques in path planning for autonomous driving," *Ann Arbor*, vol. 1001, no. 48105, pp. 18–80, 2008.

[2] P. Polack, F. Altché, B. d'Andréa Novel, and A. de La Fortelle, "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?" in *2017 IEEE intelligent vehicles symposium (IV)*. IEEE, 2017, pp. 812–818.

[3] FreyJo. acados. (2021, Feb 22). [Online]. Available: https://github.com/acados/acados/blob/master/docs

[4] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *The international journal of robotics research*, vol. 29, no. 5, pp. 485–501, 2010.

[5] J. B. Rawlings, "Tutorial overview of model predictive control," *IEEE control systems magazine*, vol. 20, no. 3, pp. 38–52, 2000.

[6] F. Allgöwer and A. Zheng, *Nonlinear model predictive control*. Birkhäuser, 2012, vol. 26.