Model Predictive Control and Control Barrier Function for Motion Planning with Obstacle Avoidance

Lihan Lian*

Abstract—**Motion planning is one of the most common task for robotics system and safety is often the most important condition that needs to be satisfied. Model Predictive Control (MPC) is known for its capability of handling operational constraints, and it computes the optimal control input by solving constrained optimization problem in a receding horizon fashion. MPC has been widely used for robot motion planning problem since it can take account of the dynamical constraints, actuator limit, state bound and so forth. However, safety still remains a critical challenge since MPC can still suffer from problems such as high computational cost and loss of recursive feasibility in some situation. Control Barrier Function (CBF) is another technique that has been widely used for safety critical tasks. CBF ensures the safety by linking the control input and safety constraints through its Lie derivative. In this study, 2D double-integrator model is used as the robotic model and both MPC, CBF and their combination are studied in the context of motion planning.**

I. Introduction

A. Motivation

Safety-critical motion planning is one the most fundamental task in the field of robotics. From robot arm to legged robot, or from autonomous vehicle to unmanned aerial vehicle (UAV) and so forth, many of their tasks essentially boil down to planning and executing one or more safe and optimal paths from one place to another. Having robot violate the safety constraint (collision with obstacles or passengers) can be a huge loss both in terms of economy and human safety. Therefore, research on safety-critical motion planning algorithm is indeed critical.

With the improvement of hardware performance enabling more complex and intensive online computation, MPC has become more and more popular in the industry. Moreover, hardware advancement also accelerates the development of deep learning, which give birth to many powerful and popular products like chatGPT. Many researchers are now working on utilizing the power of generative model to bring MPC further to the next milestone. All these factors make MPC both a hot research area and practical tool in the industry, especially in the field of robotics.

*Department of Robotics, University of Michigan

Code is available at https://github.com/lihanlian/simple-mpc-cbf

CBF is another powerful techniques that utilize nice math properties to achieve safety guarantee in a brilliant way. For a control affine system, once an appropriate CBF is found, it can then be used for many safety-critical task with provably safety guarantee. CBF is often used together with Control Lyapunov Function (CLF), which is a technique to ensure system stability. Both CBF and CLF are becoming more popular in the field of robotics research for the past decade.

This project aims to study these important techniques and have a deeper understanding by solving problems through code implementation. All implementation is in python. CasADi is used as the optimization toolbox and matplotlib is the library used for plotting figures.

B. Related Work

1) *Model Predictive Control:* MPC has been widely used for many roboitcs application and motion planning is one of the majors task where MPC can play an important role for various kinds of robot. There is study on using MPC with Bezier Curves for autonomous driving [1]. Nonlinear MPC is used together with Augemented Lagrangian for solving real-time robot arm motion planning task as shown in [2]. For legged robot, resarcher also proposed an unified MPC framework for whole-body dynamics locomotion and manipulation [3]. However, most literature use Euclidean distance between the robot and obstacle to ensure safety constraints, which will be called MPC-DC in the later section of this report. This motivates the study on the using the CBF approach and its combination with MPC.

2) *Control Barrier Function:* CBF is becoming more popular and it tends to appear more frequently in the robotics research literature in recent years as well. The application of CBF-based quadratic program has been demonstrated in the domain of adaptive cruise control [4]. Combination of CBF and MPC has been systematically investigated for the discrete-time control system [5]. There is also study that uses CBF to ensure multi-layered safety for legged robot along with the use of

MPC [6]. The advancement of deep learning also enrich the study of CBF, [7] studied the use of CBF for end-to-end deep reinforcement learning to solving safety-critical continuous control tasks. Data-driven, machine learning based approach to reduce model uncertainty and improve safe behavior of system utilizing CBF is also investigated [8]. Due to the time constraint, only model-based approach will be discussed and learning-based approach is beyond the scope of this project.

C. Paper Structure

The report is organized as follows: in section II, I will introduce the specific problem that will be solved and provide a brief introduction to both MPC and CBFs. In section III, formulation of different types of algorithms for the problem that utilize either MPC or CBF will be given. Section IV provides figures and discussion of each algorithms. Finally, section V concludes the report and share final thought on further investigations and improvement can be done.

II. Problem Statemetn

Robot Dynamics and Task

In this project, a 2D double-integrator model is used for modeling robot dynamics, since double-integrator is commonly used for modeling wheeled robot and this grounded robot is moving in a 2D plane. The state space contains fours state variables:

$$x = [x, y, \dot{x}, \dot{y}]^T \tag{1}$$

where x, y are the position at x and y axis, $\dot{x}$ and $\dot{y}$ are the velocity in the x and y direction. For a double-integrator model, the control input is acceleration, and we are only able to observe the robot positional information. Thus, the state space equations are shown as follows:

$$x_{k+1} = A_d x_k + B_d u_k, \tag{2}$$

$$y_k = C x_k \tag{3}$$

where A_d and B_d are the matrices in discrete time that corresponding to the matrices A_c and B_c of a 2D double-integrator model in continuous time. In this project, the sampling time is chosen to be 0.1 second.

There are two scenarios been studied. In both cases, robot starts from the origion (0,0) with zero initial velocity, and the goal state are both (10,10) with zero velocity. For the first case, there is only one static obstacle, which is circuluar and centered at (5,5) with radius of 2m. In the second case, there are two obstacles. First obstacle is static and it is centered at (2,4) with radius of 2m. Second obstacle is moving horizontally at $y = 6m$ and has radius of 1m. It starts from (7,6) and moving toward point (9,6), it will go back to (7,6) once reach (9,6) and moves repeatedly in the same pattern with a speed of 1m/s. The robot is also circular and has radius of 0.5m.

Model Predictive Control

MPC is essentially a control algorithm that solves optimal control input by solving an optimization problem at each time step in a receding horizon fashion. The prediction horizon N is often set in advance, and the cost function J is a function of state variable x and control input u at time step $k = 0 \dots N$ as shown follows:

$$\begin{aligned} u^* = \min_{k_{0:N-1}, u_{0:N-1}} \quad & J(x_k, u_k) \\ \text{s.t.} \quad & x_{k+1} = A_d x_k + B_d u_k, \\ & u_{min} \leq u_k \leq u_{max}, \end{aligned}$$

where $x_{k+1} = A_d x_k + B_d u_k$ stands for the constraints of system dynamics and $u_{min} \leq u_k \leq u_{max}$ stands for the constraints on actuator limits. Normally there will also be constraints such as $g(x)$ that imposes constraints to ensure conditions like safety, and both constraints for initial and goal state will also need to be satisfied. Once the control input u is solved, it is applied at time step $k = 0$, and then a new optimization problem will be solved from horizon $k = 1 \dots N+1$ to provide control input at time step $k = 1$. In this way, the constrained optimization problem is solved in a receding horizon fashion and the output (u*) is only been applied at the starting time step.

Control Barrier Function

CBF is a technique that guarantee safety through the use of set invariance, i.e., not leaving a *safe set*. Let's consider a safe set C defined as the *superlevel set* of a continuously differentiable function $h : D \subseteq \mathbb{R}^n \rightarrow \mathbb{R}$, yielding:

$$C = \{x \in D \subseteq \mathbb{R}^n : h(x) \geq 0\}, \tag{4}$$

$$\partial C = \{x \in D \subseteq \mathbb{R}^n : h(x) = 0\}, \tag{5}$$

$$\text{Int}(C) = \{x \in D \subseteq \mathbb{R}^n : h(x) > 0\}. \tag{6}$$

For a control affine system define as below:

$$\dot{x} = f(x) + g(x)u \tag{7}$$

with f and g are both locally Lipschitz. To ensure safety (state x stays within safe set C), the following condition needs to be satisfied based on [9]:

$$\sup_{u \in U} [L_f h(x) + L_g h(x) u] \geq -\alpha(h(x)). \tag{8}$$

for all $x \in D$ where α is an extended class $\mathcal{K}_\infty$ function. In this project the α is chosen to be a scalar with in the range of $[0, 1]$, same with the paper [5].

III. Methods

Two different algorithms are used for robot motion planning task in this study. There are two cases, first case contains only static obstacle and second case has both static and moving obstacle. In both cases, the purpose of the robot is to navigate to the goal state while successfully avoid collision with obstacle.

MPC-DC

MPC-DC uses the euclidean distance directly, and impose it as the safety constraint for the optimization problem. The formulation is shown as follows:

$$
\begin{aligned}
J_k^*(\mathbf{x}_k) = \min_{\mathbf{u}_{k:k+N-1}} & \sum_{k=0}^{N-1} \big((\mathbf{x}_k - \mathbf{x}_{goal})^T Q(\mathbf{x}_k - \mathbf{x}_{goal}) \\
& + \mathbf{u}_k^T R \mathbf{u}_k\big) + (\mathbf{x}_N - \mathbf{x}_{goal})^T H(\mathbf{x}_N - \mathbf{x}_{goal}) \\
\text{s.t.} \quad & \mathbf{x}_{k+1} = A_d \mathbf{x}_k + B_d \mathbf{u}_k \qquad k = 0, \dots, N-1 \\
& g(\mathbf{x}_k) \geq 0, \qquad k = 0, \dots, N-1 \\
& \mathbf{u}_{min} \leq \mathbf{u}_k \leq \mathbf{u}_{max} \qquad k = 0, \dots, N-1 \\
& \mathbf{x}_0 = \mathbf{x}_{start}
\end{aligned}
$$

where $g(x_k)$ is the function that describe the euclidean distance between the robot and the obstacle. Q and R are the constant matrics that assign the weight of penalty to the difference between the current state and goal state and the control input. H is the matrix used to assign terminal cost.

MPC-CBF

MPC-CBF also uses the euclidean distance, but it uses the euclidean distance to first formulate a CBF, then uses the CBF to impose the safety constraint for the optimization problem. The formulation of MPC-CBF is shown below:

$$
\begin{aligned}
J_k^*(\mathbf{x}_k) = \min_{\mathbf{u}_{k:k+N-1}} & \sum_{k=0}^{N-1} \big((\mathbf{x}_k - \mathbf{x}_{goal})^T Q(\mathbf{x}_k - \mathbf{x}_{goal}) \\
& + \mathbf{u}_k^T R \mathbf{u}_k\big) + (\mathbf{x}_N - \mathbf{x}_{goal})^T H(\mathbf{x}_N - \mathbf{x}_{goal}) \\
\text{s.t.} \quad & \mathbf{x}_{k+1} = A_d \mathbf{x}_k + B_d \mathbf{u}_k \qquad k = 0, \dots, N-1 \\
& L_f h(x_k) + L_g h(x_k) u \geq -\gamma(h(x)) \qquad k = 0, \dots, N-1 \\
& \mathbf{u}_{min} \leq \mathbf{u}_k \leq \mathbf{u}_{max} \qquad k = 0, \dots, N-1 \\
& \mathbf{x}_0 = \mathbf{x}_{start}
\end{aligned}
$$

where $h(x_k)$ is the control barrier function which is the same as the euclidean distance function defined in MPC-DC algorithm. $L_f h(x_k)$ is the Lie derivative of function $h(x)$ along the vector field f and $L_g h(x_k)$ is the Lie derivative of function $h(x)$ along the vector field g. The function γ is an extended class $\mathcal{K}_\infty$ function and scalar values are used for code implementation ($\gamma = 0.2, 0.5, 0.7, etc.$). Q, R and H are the same weight matrices as defined in the MPC-DC algorithm.

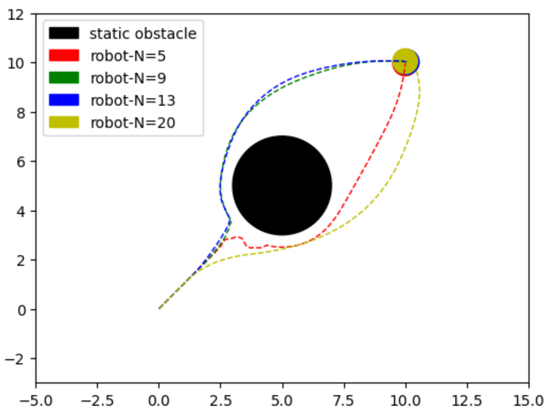

Fig. 1: Planning results using MPC-DC in Case I.

IV. Discussion

Case I. One Static Obstacle

In the first case, the only obstacle is static and it is centered at (5,5) with radius of 2m. For MPC-DC algorithm, from table I and Fig 1 we can see, at all cases of N, the robot will not be able to fully avoid collision with obstacle (glancing collision with obstacle edge). In addition, as the prediction horizon increasing, the total time for finishing traveling is decreasing.

All MPC-CBF algorithms are tested with the prediction horizon $N = 10$. It is clear that it is more good at avoiding obstacle as shown in Figure 2. It is able to avoid the collision earlier and the smaller the value of γ, the further it will be from the obstacle.

Horizon	Time	Total Cost	Collision Avoidance
N = 5	10.8	780260	*Barely*
N = 9	8.6	901565	*Barely*
N = 13	7.8	908592	*Barely*
N = 20	7.2	650552	*Barely*

TABLE I Comparison on the performance of different prediction horizon of MPC-DC algorithm in case I.

γ	Time	Total Cost	Collision Avoidance
γ = 0.2	7.2	962287	✓
γ = 0.5	7.4	946826	✓
γ = 0.7	8.0	1104570	✓
γ = 0.9	7.7	988671	✓

TABLE II Comparison on the performance of different hypermarater γ of MPC-CBF algorithm in case I.

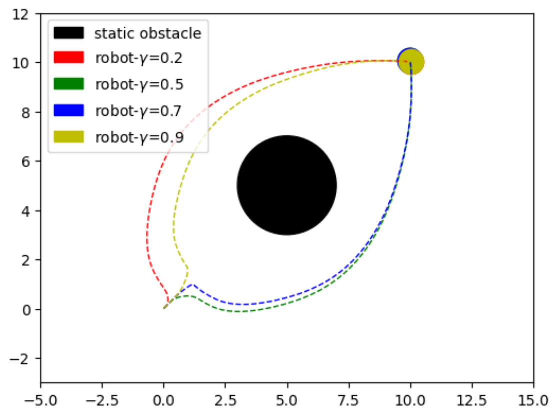

Fig. 2: Planning results using MPC-CBF in Case I.

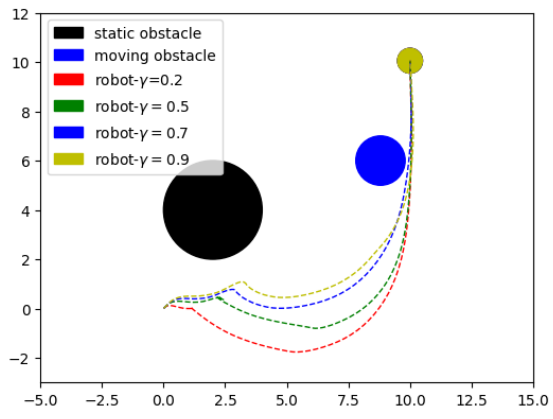

Fig. 4: Planning results using MPC-CBF in Case II.

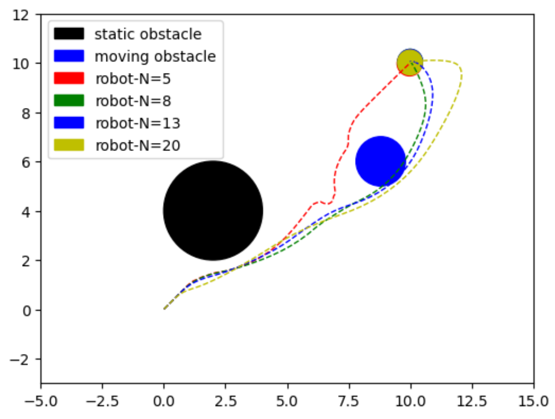

Fig. 3: Planning results using MPC-DC in Case II.

Horizon	Time	Total Cost	Collision Avoidance
N = 5	10.8	752102	×
N = 8	7.0	599406	×
N = 13	7.5	643789	×
N = 20	7.8	644428	×

TABLE III Comparison on the performance of different prediction horizon of MPC-DC algorithm in case II.

Case II. One Static and One Moving Obstacle

In the second case, with the existence of both static and moving obstacle, MPC-DC algorithm still fails to avoid collision with obstacle at all cases of prediction horizon N. It shows the similar trend for travel time and total cost as in case I. As the prediction horizon increasing, the total time for finishing traveling tends to decrease and the total cost tends to increase.

All MPC-CBF algorithms are again tested with the prediction horizon $N = 10$ and it is still obvious that CBF helps robot to become more good at avoiding obstacle as shown in Figure 4. Smaller value of γ will result in longer time for travel and increase of total cost, but they failed to avoid collision with the dynamic obstacle ($\gamma = 0.2, 0.5$) as shown in Table IV.

V. Conclusion

In conclusion, this project studied both the concept of MPC and CBF through implementing the solution of two specific problems using python and CasADi. MPC-DC utilizes the euclidean distance as the safety constraints directly and it is easier both in terms of formulation and implementation. MPC-CBF has a more complex formulation as it requires choosing an appropriate CBF at first, and then the value of hyperparameter γ might need tunning for better performance. In the case of 2D double-integrator model, CBF helps the robot to avoid collision with obstacle in both two cases. Further study can be extending to a more complex nonlinear plant model, investigating learning method or adding disturbance and noise and so forth. In addition, another CBF-based method (CBF-QP) can be explored, which requires a nominal controller and the cost function is simply a quadratic term of the difference between actual controller input and nominal controller input.

γ	Time	Total Cost	Collision Avoidance
γ = 0.2	11.7	2405130	×
γ = 0.5	11.4	2016030	×
γ = 0.7	8.7	1226210	✓
γ = 0.9	8.8	1190030	✓

TABLE IV Comparison on the performance of different hypermarater γ of MPC-CBF algorithm in case II.

References

[1] X. Qian, I. Navarro, A. de La Fortelle, and F. Moutarde, "Motion planning for urban autonomous driving using bézier curves and mpc," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, Rio de Janeiro, Brazil, 2016, pp. 826–833.

[2] A. S. Sathyajith, J. Gillis, G. Pipeleers, and J. Swevers, "Real-time robot arm motion planning and control with nonlinear model predictive control using augmented lagrangian on a first-order solver," in *2020 European Control Conference (ECC)*, St. Petersburg, Russia, 2020, pp. 507–512.

[3] J. P. Sleiman, F. Farshidian, M. V. Minniti, and M. Hutter, "A unified mpc framework for whole-body dynamic locomotion and manipulation," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4688–4695, July 2021.

[4] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *53rd IEEE Conference on Decision and Control*, Los Angeles, CA, USA, 2014, pp. 6271–6278.

[5] J. Zeng, B. Zhang, and K. Sreenath, "Safety-critical model predictive control with discrete-time control barrier function," in *2021 American Control Conference (ACC)*, New Orleans, LA, USA, 2021, pp. 3882–3889.

[6] R. Grandia, A. J. Taylor, A. D. Ames, and M. Hutter, "Multi-layered safety for legged robots via control barrier functions and model predictive control," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 8352–8358.

[7] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI, 2019, available: https://doi.org/10.1609/aaai.v33i01.33013387.

[8] A. Taylor, A. Singletary, Y. Yue, and A. Ames, "Learning for safety-critical control with control barrier functions," in *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, ser. Proceedings of Machine Learning Research, vol. 120. PMLR, 2020, pp. 708–717.

[9] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European Control Conference (ECC)*, Naples, Italy, 2019, pp. 3420–3431.