Safety-Critical Model Predictive Control with Discrete-Time Control Barrier Function

Jun Zeng*, Bike Zhang* and Koushil Sreenath

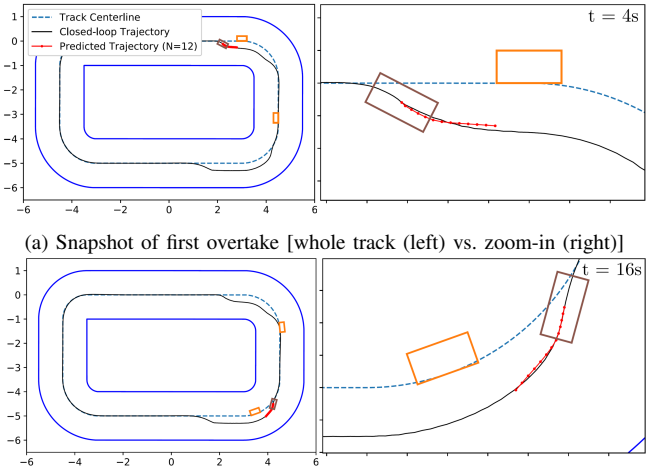
***Abstract*— The optimal performance of robotic systems is usually achieved near the limit of state and input bounds. Model predictive control (MPC) is a prevalent strategy to handle these operational constraints, however, safety still remains an open challenge for MPC as it needs to guarantee that the system stays within an invariant set. In order to obtain safe optimal performance in the context of set invariance, we present a safety-critical model predictive control strategy utilizing discrete-time control barrier functions (CBFs), which guarantees system safety and accomplishes optimal performance via model predictive control. We analyze the stability and the feasibility properties of our control design. We verify the properties of our method on a 2D double integrator model for obstacle avoidance. We also validate the algorithm numerically using a competitive car racing example, where the ego car is able to overtake other racing cars.**

I. Introduction

A. Motivation

Safety-critical optimal control and planning is one of the fundamental problems in robotic applications. In order to ensure the safety of robotic systems while achieving optimal performance, the tight coupling between potentially conflicting control objectives and safety criteria is considered in an optimization problem. Some recent work formulates this problem using control barrier functions, but only using current state information without prediction, see [1], [2], which yields a greedy control policy. Model predictive control can give a less greedy policy, as it takes future state information into account. However, the safety criteria in a predictive control framework is usually enforced as distance constraints defined under Euclidean norms, such as the distance between the robot and obstacles being larger than a safety margin. This distance constraint will not confine the optimization until the reachable set along the horizon intersects with the obstacles. In other words, the robot will not take actions to avoid the obstacles until it is close to them. One way to solve this problem is to use a larger horizon, but that will increase the computational complexity in the optimization.

We address this challenge above by directly unifying model predictive control with discrete-time control barrier functions together into one optimization problem. This results in a safety-critical model predictive control formulation, called MPC-CBF in this paper. In this formulation, the CBF constraints could enforce the system to avoid obstacles even when the reachable set along the horizon is far away from the obstacles. We validate this control design using a 2D double integrator for obstacle avoidance, and also demonstrate that this method enables a racing car to safely compete with other cars in a racing competition, shown in Fig. 1.



(a) Snapshot of first overtake [whole track (left) vs. zoom-in (right)]

(b) Snapshot of second overtake [whole track (left) vs. zoom-in (right)]

Fig. 1: The proposed safety-critical model predictive control is applied to a competitive car racing example. Snapshots of the ego car (brown) are shown overtaking other racing cars (orange) from both right and left sides while maintaining a target speed. The closed-loop trajectory and predicted open-loop trajectory are colored in black and red respectively, and the blue solid lines depict the boundary of racing track.

B. Related Work

1) Model Predictive Control: MPC is widely used for robotic systems, such as robotic manipulation and locomotion [3], [4], to achieve optimal performance while satisfying different constraints. One of the most important criteria to deploy robots for real-world tasks is safety. There is some existing work about model predictive control considering system safety [5], [6]. The safety criteria in the context of MPC is usually formulated as constraints in an optimization problem [7], [8], such as obstacle constraints and actuation limits. One concrete scenario regarding safety criteria for robots is obstacle avoidance. The majority of literature focuses on collision avoidance using simplified models, and considers distance constraints with various Euclidean norms [9]–[11], which we call MPC-DC in this paper.

However, these obstacle avoidance constraints under Eu-

* Authors have contributed equally and names are in alphabetical order.
Jun Zeng, Bike Zhang and Koushil Sreenath are with the Department of Mechanical Engineering, University of California, Berkeley, CA, 94720, USA, {zengjunsjtu, bikezhang, koushils}@berkeley.edu

Code and simulation video are available at https://github.com/HybridRobotics/MPC-CBF

clidean norms will not confine the robot's movement unless the robot is relatively close to the obstacles. To make the robot take actions to avoid obstacles even far away from it, we usually need a larger horizon which increases the computational time in the optimization. This encourages us to formulate a new type of model predictive control, which can guarantee safety in the context of set invariance with CBF constraints confining the robot's movement during the optimization. Recently in [12], model predictive control is introduced with control Lyapunov functions to ensure stability.

2) Control Barrier Functions: CBFs have recently been introduced as a promising way to ensure set invariance by considering the system dynamics and implying forward invariance of the safe set. Furthermore, a safety-critical control design for continuous-time systems was proposed by unifying a control Lyapunov function (CLF) and a control barrier function through a quadratic program (CLF-CBF-QP) [13]. This method could be deployed as a real-time optimization-based controller with safety-critical constraints, shown in [1], [14], [15]. Besides the continuous-time domain, the formulation of CBFs was generalized into discrete-time systems (DCLF-DCBF) in [2].

3) Model Predictive Control with Control Barrier Functions: There is some existing work that tries to combine the advantages of MPC and CBFs. Barrier functions have been used in MPC in [16], which convert constraints to cost but not related to safety-critical control. In [5], they use continuous-time CBFs as constraints inside discrete-time MPC. MPC and CBFs are organized as a high-level planner and a low-level tracker in [6]. This method treats MPC and CBFs separately at different levels.

Inspired by the previous work of model predictive control and control barrier functions, DCLF-DCBF can be improved by taking future state prediction into account, yielding a better control policy. This motivates us to investigate the control design of predictive control under the constraints imposed by CBFs. In this paper, we focus on the discrete-time formulation of control barrier functions applied to model predictive control, which encodes the safety from discrete-time CBFs in MPC.

C. Contribution

The contributions of this paper are as follows.

- We present a MPC-CBF control design for safety-critical tasks, where the safety-critical constraints are enforced by discrete-time control barrier functions.
- We analyze the stability of our control design with sufficient conditions, and qualitatively discuss the feasibility in terms of set intersections between reachable sets of MPC and safe sets enforced by CBF constraints along the horizon.
- Our proposed method is shown to outperform both MPC-DC and DCLF-DCBF. It enables prediction capability to DCLF-DCBF for performance improvement, and it also guarantees safety via discrete-time CBF constraints in the context of set invariance.
- We verify the properties of our control design using a 2D double integrator for obstacle avoidance. Our algorithm is generally applicable and also validated in a more complex scenario, where MPC-CBF enables a car racing on a track while safely overtaking other cars.a

D. Paper Structure

This paper is organized as follows: in Sec. II, we present the background of model predictive control and control barrier functions. In Sec. III, we introduce the safety-critical model predictive control design using discrete-time control barrier functions (MPC-CBF). The analysis of stability and feasibility properties is presented and the relations with DCLF-DCBF and MPC-DC are also discussed. To validate the control design and verify the properties of our formulation, a 2D double integrator for obstacle avoidance and a car racing competition example are demonstrated in Sec. IV. Sec. V provides concluding remarks.

II. Background

Our proposed safety-critical model predictive control design builds on model predictive control and control barrier functions. We now present necessary preliminaries.

A. Model Predictive Control

Consider the problem of regulating to the origin of a discrete-time control system

$$x_{t+1} = f(x_t, u_t), \tag{1}$$

where $x_t \in \mathcal{X} \subset \mathbb{R}^n$ represents the state of the system at time step $t \in \mathbb{Z}^+$, $u_t \in \mathcal{U} \subset \mathbb{R}^m$ is the control input, and f is locally Lipschitz.

Assume that a full measurement or estimate of the state x_t is available at the current time step t. Then a finite-time optimal control problem is solved at time step t. When there are safety criteria, such as obstacle avoidance, the obstacles are usually formulated using distance constraints. The finite-time optimal control formulation is shown in (2).

MPC-DC:

$$J_t^*(x_t) = \min_{u_{t:t+N-1|t}} p(x_{t+N|t}) + \sum_{k=0}^{N-1} q(x_{t+k|t}, u_{t+k|t}) \tag{2a}$$

$$\text{s.t.} \quad x_{t+k+1|t} = f(x_{t+k|t}, u_{t+k|t}),\ k = 0, ..., N-1 \tag{2b}$$

$$x_{t+k|t} \in \mathcal{X}, u_{t+k|t} \in \mathcal{U},\ k = 0, ..., N-1 \tag{2c}$$

$$x_{t|t} = x_t, \tag{2d}$$

$$g(x_{t+k|t}) \geq 0,\ k = 0, ..., N-1. \tag{2e}$$

Here $x_{t+k|t}$ denotes the state vector at time step t + k predicted at time step t obtained by starting from the current state x_t (2d), and applying the input sequence $u_{t:t+N-1|t}$ to the system dynamics (2b). In (2a), the terms $q(x_{t+k|t}, u_{t+k|t})$ and $p(x_{t+N|t})$ are referred to as stage cost and terminal cost respectively, and N is the time horizon. The state and input constraints are given by (2c), and distance constraints for

safety criteria are represented by function g, in (2e), which could be defined under various Euclidean norms.

Let $u^*_{t:t+N-1|t} = \{u^*_{t|t}, ..., u^*_{t+N-1|t}\}$ be the optimal solution of (2) at time step t. The resulting optimized trajectory using $u^*_{t:t+N-1|t}$ is referred as an open-loop trajectory. Then, the first element of $u^*_{t:t+N-1|t}$ is applied to system (1). This feedback control law is given below,

$$u(t) = u^*_{t|t}(x_t). \tag{3}$$

The finite-time optimal control problem (2) is repeated at next time step $t+1$, based on the new estimated state $x_{t+1|t+1} = x_{t+1}$. It yields the model predictive control strategy. The resulting trajectory using (3) is referred as a closed-loop trajectory. More details can be referred to in [17].

B. Control Barrier Functions

We now present discrete-time control barrier functions that will be used together with model predictive control for our control design, which will be introduced in Sec. III.

For safety-critical control, we consider a set $\mathcal{C}$ defined as the superlevel set of a continuously differentiable function $h : \mathcal{X} \subset \mathbb{R}^n \to \mathbb{R}$,

$$\mathcal{C} = \{x \in \mathcal{X} \subset \mathbb{R}^n : h(x) \geq 0\}. \tag{4}$$

Throughout this paper, we refer to $\mathcal{C}$ as a safe set. The function h becomes a control barrier function if $\dfrac{\partial h}{\partial x} \neq 0$ for all $x \in \partial\mathcal{C}$ and there exists an extended class $\mathcal{K}_\infty$ function γ such that for the control system (1) h satisfies,

$$\exists\ u \text{ s.t. } \dot{h}(x,u) \geq -\gamma(h(x)),\ \gamma \in \mathcal{K}_\infty. \tag{5}$$

This condition can be extended to the discrete-time domain which is shown as follows.

$$\Delta h(x_k, u_k) \geq -\gamma h(x_k),\ 0 < \gamma \leq 1, \tag{6}$$

where $\Delta h(x_k, u_k) := h(x_{k+1}) - h(x_k)$. Satisfying constraint (6), we have $h(x_{k+1}) \geq (1-\gamma)h(x_k)$, i.e, the lower bound of control barrier function $h(x)$ decreases exponentially with the rate $1-\gamma$.

Remark 1: Note that in (6), we defined γ as a scalar instead of a $\mathcal{K}_\infty$ function as in (5). Generally, for the discrete-time domain, γ could also be considered as a class $\mathcal{K}$ function that also additionally satisfies $0 < \gamma(h(x)) \leq h(x)$ for any $h(x)$. However, we will continue to use the scalar form γ in this paper to simplify the notations for further discussions.

Besides the system safety, we are also interested in stabilizing the system with a feedback control law u under a control Lyapunov function V, i.e.,

$$\exists\ u \text{ s.t. } \dot{V}(x,u) \leq -\alpha(V(x)),\ \alpha \in \mathcal{K}. \tag{7}$$

We can also generalize it to the discrete-time domain,

$$\Delta V(x_k, u_k) \leq -\alpha V(x_k),\ 0 < \alpha \leq 1, \tag{8}$$

where $\alpha > 0$ and $\Delta V(x_k, u_k) := V(x_{k+1}) - V(x_k)$. Similarly as above, the upper bound of control Lyapunov function decreases exponentially with the rate $1-\alpha$.

The discrete-time control Lyapunov function and control barrier function can be unified into one optimization program (DCLF-DCBF), which achieves the control objective and guarantees system safety. This formulation was first introduced in [2] and is presented as follows,

DCLF-DCBF:

$$u^*_k = \underset{(u_k,\delta)\in\mathbb{R}^{m+1}}{\operatorname{argmin}} \quad u_k^T H(x) u_k + l \cdot \delta^2 \tag{9a}$$

$$\Delta V(x_k, u_k) + \alpha V(x_k) \leq \delta \tag{9b}$$

$$\Delta h(x_k, u_k) + \gamma h(x_k) \geq 0 \tag{9c}$$

$$u_k \in \mathcal{U}, \tag{9d}$$

where l is positive, and $\delta \geq 0$ is a slack variable that allows the Lyapunov function to grow when the CLF and CBF constraints are conflicting. The safe set $\mathcal{C}$ in (4) is invariant along the trajectories of the discrete-time system with controller (9) if $h(x_0) \geq 0$ and $0 < \gamma \leq 1$.

III. Control Design

After presenting a background of model predictive control and control barrier functions, we formulate the safety-critical model predictive control logic in this section.

A. Formulation

Consider the problem of regulating to a target state for the discrete-time system (1) while ensuring safety in the context of set invariance. The proposed control logic MPC-CBF solves the following constrained finite-time optimal control problem with horizon N at each time step t,

MPC-CBF:

$$J^*_t(x_t) = \min_{u_{t:t+N-1|t}} p(x_{t+N|t}) + \sum_{k=0}^{N-1} q(x_{t+k|t}, u_{t+k|t}) \tag{10a}$$

$$\text{s.t.} \quad x_{t+k+1|t} = f(x_{t+k|t}, u_{t+k|t}),\ k = 0, ..., N-1 \tag{10b}$$

$$x_{t+k|t} \in \mathcal{X},\ u_{t+k|t} \in \mathcal{U},\ k = 0, ..., N-1 \tag{10c}$$

$$x_{t|t} = x_t, \tag{10d}$$

$$\Delta h(x_{t+k|t}, u_{t+k|t}) \geq -\gamma h(x_{t+k|t}),\ k = 0, ..., N-1 \tag{10e}$$

where (10d) represents the initial condition constraint, (10b) describes the system dynamics, and (10c) shows the state/input constraints along the horizon. The CBF constraints imposed in (10e) are designed to guarantee the forward invariance of the safe set $\mathcal{C}$ associated with the discrete-time control barrier functions, where Δh is introduced in (6). Here we have

$$\Delta h(x_{t+k|t}, u_{t+k|t}) = h(x_{t+k+1|t}) - h(x_{t+k|t}).$$

The optimal solution to (10) at time t is a sequence of inputs as $u^*_{t:t+N-1|t} = [u^*_{t|t}, ..., u^*_{t+N-1|t}]$. Then, the first element of the optimizer vector is applied,

$$u(t) = u^*_{t|t}(x_t). \tag{11}$$

This constrained finite-time optimal control problem (10) is repeated at time step $t+1$, based on the new state $x_{t+1|t+1}$, yielding a receding horizon control strategy: safety-critical model predictive control.

The system dynamics constraint in (10b) could be linear if we have a linear system, and (10c) could also be linear if $\mathcal{X}$ and $\mathcal{U}$ are defined as polytopes in the state and input space, respectively. The discrete-time control barrier functions constraints in (10e) are generally non-convex unless the CBFs are linear. This makes the whole optimization in (10) generally become a nonlinear programming problem (NLP).

B. Stability

In DCLF-DCBF, control Lyapunov functions are introduced as optimization constraints (9b) with corresponding slack variable as additional term in the cost function (9a). This allows for the achievement of control objectives represented by CLFs and unifies the formulation under one optimization with CBFs. In our MPC-CBF control design, we have the terminal cost $p(x_{t+N|t})$ in (10a) as a control Lyapunov function, which guarantees the stability of the system along the closed-loop trajectory. This stability of the closed-loop system is guaranteed if the following holds,

$$p(f(x_{t+k|t}, u_{t+k|t})) - p(x_{t+k}) + q(x_{t+k}, u_{t+k}) \leq 0, \ \forall t, k. \tag{12}$$

This inequality provides the sufficient conditions for stability, and comes from the function J_t^* that should decrease along the closed-loop trajectories, where we have $J_t^*(x_{t+1}) \leq J_t^*(x_t)$. The proof of stability based on (12) could be found in [17, Thm. 12.2].

C. Feasibility

Recursive feasibility is generally not guaranteed for MPC-DC defined in (2) [17], [18] and other general NLP problems [19, Chap. 11]. In this paper, we qualitatively analyze the feasibility problem of MPC-CBF based on set analysis. Since the optimization (10) could be a nonlinear programming problem, we are interested in finding under which circumstances this optimization becomes feasible, i.e., the feasible set under the constraints (10b)-(10e) is not empty.

Given current state x_t in (10d), the reachable set at horizon step k is defined as a reachable region in the state space, satisfying system dynamics in (10b), input/state constraints in (10c) and initial condition in (10d). This reachable set $\mathcal{R}(x_t, \mathcal{X}, \mathcal{U}, k)$ is defined as follows,

$$\begin{aligned}\mathcal{R}(x_t, \mathcal{X}, \mathcal{U}, k) = \{x_{t+k|t} \in \mathcal{X} : \forall i = 0, ..., k-1, \\ x_{t+i+1|t} = f(x_{t+i|t}, u_{t+i|t}), \\ x_{t+i|t} \in \mathcal{X}, u_{t+i|t} \in \mathcal{U}, x_{t|t} = x_t\}.\end{aligned} \tag{13}$$

We also define the set of state space satisfying the CBF constraints in (10e) and initial condition in (10d) as,

$$\begin{aligned}\mathcal{S}_{cbf}(k) = \{ x \in \mathcal{X} : \\ h(x) - h(x_{t+k-1|t}) \geq -\gamma h(x_{t+k-1|t}) \} ,\end{aligned} \tag{14}$$

where $\mathcal{S}_{cbf}(k)$ describes superlevel sets of h satisfying the control barrier function constraints (10e) at each time step along the open-loop trajectory.

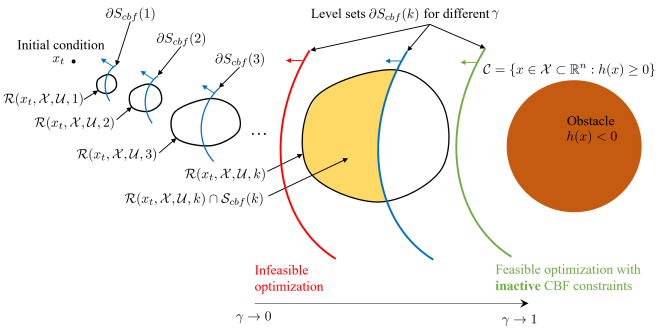


Fig. 2: Feasibility of MPC-CBF. The reachable set $\mathcal{R}(x_t, \mathcal{X}, \mathcal{U}, k)$ propagates along the horizon from the initial condition x_t. For horizon step k in the open-loop, the level sets $\partial \mathcal{S}_{cbf}(k)$ are shown in different colors with three choices of γ and each corresponding $\mathcal{S}_{cbf}(k)$ lies on the left hand side of level sets, indicated by the arrows in different colors

.

We illustrate $\mathcal{R}(x_t, \mathcal{X}, \mathcal{U}, k)$ and $\mathcal{S}_{cbf}(k)$ in the state space shown in Fig. 2, given the initial condition x_t. Then, the feasibility of the optimization in (10) turns out to be whether the intersection between the feasible set at each horizon step, $\mathcal{R}(x_t, \mathcal{X}, \mathcal{U}, k)$, and the superlevel set of $h(x)$ satisfying the CBF constraints, $\mathcal{S}_{cbf}(k)$, is nonempty for all $k = 1, ...N$.

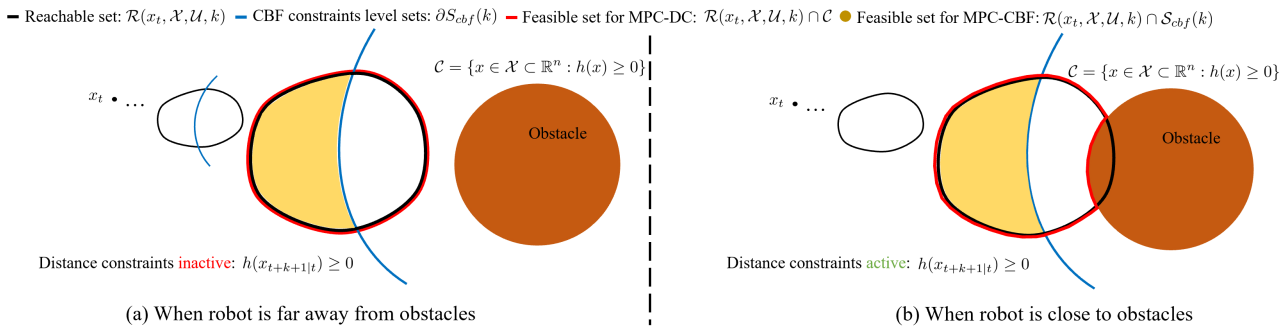
Remark 2: Note that $\mathcal{S}_{cbf}(k)$ is not empty when h is a valid control barrier function. Furthermore, $\mathcal{R}(x_t, \mathcal{X}, \mathcal{D}, k)$ is also guaranteed to be nonempty, if we choose $\mathcal{X}$ as a control invariant set as discussed in [17, Thm. 11.1 and 11.2].

In order to better understand this problem, we illustrate the level sets of control barrier function constraints as,

$$\partial \mathcal{S}_{cbf}(k) = \{x \in \mathcal{X} : h(x) = (1-\gamma) h(x_{t+k-1|t})\},$$

with several choices of γ. The superlevel set $\mathcal{S}_{cbf}(k)$ are the regions illustrated on the left hand side of these level sets, with an example where the robot approaches the obstacle from the left to the right, shown in Fig. 2. We can see that, if γ becomes relatively small, the safe set will be more constrained. In this case, the system tends to be safer, but the intersection between $\mathcal{R}(x_t, \mathcal{X}, \mathcal{U}, k)$ and $\mathcal{S}_{cbf}(k)$ might be infeasible if γ becomes too small. When the γ becomes larger, the region of $\mathcal{S}_{cbf}(k)$ in the state space will be increased. This will make the optimization more likely to be feasible, however, the CBF constraints might not be active during the optimization, if γ is too large. In this case, $\mathcal{R}(x_t, \mathcal{X}, \mathcal{U}, k)$ will become a proper subset of $\mathcal{S}_{cbf}(k)$.

Remark 3: When γ becomes relatively small, the MPC-CBF controller makes a smaller subset of the safe set $\mathcal{C}$ in (4) invariant, and thus is more safer, but this might also make the optimization infeasible. A larger γ will make the optimization more likely to be feasible, but the CBF constraints might not be active during the optimization. We expect that γ is chosen appropriately, such that the intersection between these two sets will not be empty and becomes a proper subset of $\mathcal{R}(x_t, \mathcal{X}, \mathcal{U}, k)$. This leads to a tradeoff between safety and feasibility in terms of the choice of γ. However, it still remains an open challenge for how to automatically choose the γ.



(a) When robot is far away from obstacles

(b) When robot is close to obstacles

Fig. 3: For MPC-CBF, the feasible set at each horizon step k is the intersection between $\mathcal{R}(x_t, \mathcal{X}, \mathcal{U}, k)$ and $\mathcal{S}_{cbf}(k)$, colored in yellow. For MPC-DC, the feasible set at each horizon step is the intersection between $\mathcal{R}(x_t, \mathcal{X}, \mathcal{U}, k)$ and $\mathcal{C}$, colored with borders in red. We see clearly that MPC-CBF is safer than MPC with distance constraints with smaller set invariance. Moreover, distance constraints might be inactive when the robot is still far away from obstacles, illustrated in (a), where $\mathcal{R}(x_t, \mathcal{X}, \mathcal{U}, k)$ is a subset of $\mathcal{C}$. CBF constraints could still confine the robot's reachable set $\mathcal{R}(x_t, \mathcal{X}, \mathcal{U}, k)$ with appropriate choices of γ even when the robot is far away from obstacles, shown in (a) and (b).

Remark 4: Given x_t, $\mathcal{X}$, $\mathcal{U}$, we could pick a value of γ to find a tradeoff between safety and feasibility. However, when the system evolves, this γ might no longer satisfy our safety demand or guarantee the optimization feasibility. Therefore, for a given fixed γ, we generally only have pointwise feasibility and a persistently feasible formulation is still an open problem and is part of future work.

D. Relation with DCLF-DCBF

When $N = 1$, the formulation in (10) could be simplified as an optimization over one step system input u_k^*,

$$u_k^* = \underset{u_k \in R^m}{\operatorname{argmin}} \quad p(f(x_k, u_k)) + q(x_k, u_k) \tag{15a}$$

$$\Delta h(x_k, u_k) + \gamma h(x_k) \geq 0, \tag{15b}$$

$$u_k \in \mathcal{U}, \tag{15c}$$

where $p(f(x_k, u_k))$ and $q(x_k, u_k)$ are the terminal cost and stage cost which we have seen previously in (10a). The optimization (15) is similar to the DCLF-DCBF formulation (9). The stage cost $q(x_k, u_k)$ minimizes the system input, similar as $u_k^T H(x) u_k$ in (9a). The terminal cost minimizes the control Lyapunov function $p(f(x_k, u_k))$ in (15a), instead of using CLF constraints as (9b). As we no longer use the CLF constraint, we do not need a slack variable, such as δ in (9a), to guarantee the feasibility.

Remark 5: As the CLF constraints in (9b) are transferred into the cost function in (15) with $N = 1$, the formulation in (15) becomes similar to DCLF-DCBF. To sum up, our MPC-CBF formulation operates in a similar manner of DCLF-DCBF with prediction $N = 1$.

E. Relation with MPC-DC

When γ approaches its upper bound of 1, the CBF constraints in (10e) becomes,

$$h(x_{t+k+1|t}) \geq 0.$$

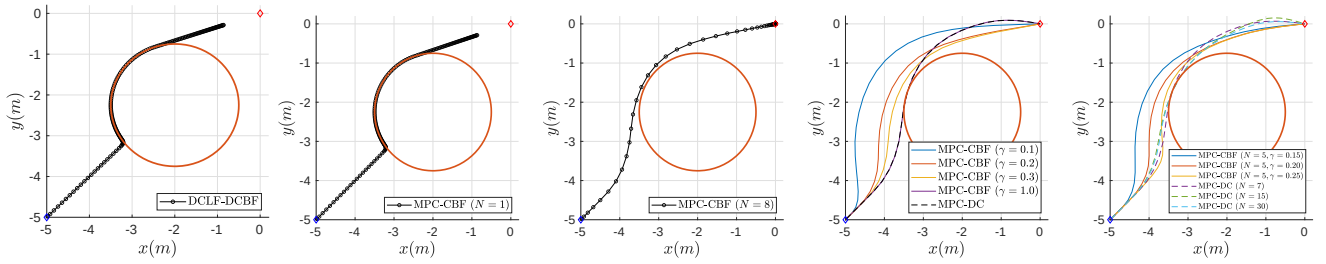
If $g(x)$ in (2e) and $h(x)$ are the same, these CBF constraints are almost the same as distance constraints defined in (2e) except for one horizon step difference. In other words, the CBF constraints function are at the next predicted horizon step instead of the current horizon step. Moreover, the feasible set at each horizon step k in MPC-DC formulation becomes $\mathcal{R}(x_t, \mathcal{X}, \mathcal{U}, k) \cap \mathcal{C}$. While, as we have seen previously, the feasible set in MPC-CBF formulation at each horizon step k is $\mathcal{R}(x_t, \mathcal{X}, \mathcal{U}, k) \cap \mathcal{S}_{cbf}(k)$. Note that $\mathcal{S}_{cbf}(k)$ is a subset of $\mathcal{C}$. Therefore, the MPC-CBF formulation in (10) has a smaller safe set than the MPC-DC formulation.

In the case the reachable set $\mathcal{R}(x_t, \mathcal{X}, \mathcal{U}, k)$ is a proper subset of the safe set $\mathcal{C}$, then the distance constraints in (2e) are not active in the optimization (2), as shown in Fig. 3a. In other words, the distance constraints will not be active in the optimization (2) until the reachable set along the horizon intersects with the unsafe regions, i.e., the reachable set intersects the obstacles as shown in Fig. 3b. Using our MPC-CBF formulation, the CBF constraints in (10e) could always confine the reachable set $\mathcal{R}(x_t, \mathcal{X}, \mathcal{U}, k)$ with an appropriate choice of γ whenever the robot tends to approach the obstacles. In this case, even when the reachable set $\mathcal{R}(x_t, \mathcal{X}, \mathcal{U}, k)$ is a proper subset of the safe set $\mathcal{C}$, the reachable set could be still constrained by its intersection with $\mathcal{S}_{cbf}(k)$, as shown in Fig. 3a.

Remark 6: We discuss the constraint activation only in the cases when the robot is moving towards the obstacles, i.e., there exists a control input u such that $h(f(x, u)) < h(x)$. When the robot is moving away from the obstacles, both distance constraints and CBF constraints are inactive, which is intuitive since the robot is always safe in this case.

Remark 7: Our MPC-CBF formulation is safer than MPC-DC in the context of smaller set invariance, where we can see that $\mathcal{R}(x_t, \mathcal{X}, \mathcal{U}, k) \cap \mathcal{S}_{cbf}(k)$ is a proper subset of $\mathcal{R}(x_t, \mathcal{X}, \mathcal{U}, k) \cap \mathcal{C}$, as shown in Fig. 3.

Remark 8: In practice, we may need to use a smaller horizon for speeding up the optimization. To achieve a similar performance compared to the MPC-DC formulation in (2), we could apply smaller γ in our MPC-CBF control design in (10) and use a smaller horizon. This could help us to reduce the complexity of the optimization for obstacle avoidance, as will be shown in Fig. 4e.



(a) (b) (c) (d) (e)

Fig. 4: A 2D double integrator avoids an obstacle using different control designs. The obstacle is represented by a red circle and the start and target positions are located at $(-5, -5)$ and $(0, 0)$, labelled as blue and red diamonds, respectively. (a) uses a DCLF-DCBF controller; (b) a MPC-CBF controller with $N = 1$; (c) a MPC-CBF controller with $N = 8$ and $\gamma = 0.5$; (d) a MPC-DC controller with $N = 8$ and four MPC-CBF controllers with $N = 8$ and different choices of γ; (e) three MPC-CBF controller with $N = 5$ and different values of γ and three MPC-DC controllers with different values of horizon N.

IV. Examples

Having presented the proposed MPC-CBF control design, we now numerically validate it using a 2D double integrator for obstacle avoidance and analyze its properties. We also apply this control design to a competitive car racing problem. For solving the following nonlinear optimization problems in MPC-CBF/MPC-DC/DCLF-DCBF, we use the solver IPOPT [20].

A. 2D Double Integrator for Obstacle Avoidance

Consider a linear discrete-time 2D double integrator system,

$$x_{k+1} = Ax_k + Bu_k, \tag{16}$$

where the sampling time Δt is set as $0.2s$

A MPC-CBF is designed as in (10) and (11) for 2D double integrator to avoid an obstacle, where the stage cost and and terminal cost are as

$$q(x_k, u_k) = x_k' Q x_k + u_k' R u_k, \tag{17}$$
$$p(x_N) = x_N' P x_N, \tag{18}$$

where $Q = 10 \cdot \mathcal{I}_4$, $R = \mathcal{I}_2$ and $P = 100 \cdot \mathcal{I}_4$. The system is subject to state constraint $\mathcal{X}$ and input constraints $\mathcal{U}$,

$$\mathcal{X} = \{x_k \in \mathbb{R}^n : x_{min} \leq x_k \leq x_{max}\},$$
$$\mathcal{U} = \{u_k \in \mathbb{R}^m : u_{min} \leq u_k \leq u_{max}\}.$$

The lower and upper bounds are,

$$x_{min} = -5 \cdot \mathcal{I}_{4\times 1},\ x_{max} = 5 \cdot \mathcal{I}_{4\times 1},$$
$$u_{min} = -\mathcal{I}_{2\times 1},\ u_{max} = \mathcal{I}_{2\times 1}.$$

For discrete-time control barrier function constraint (10e), we choose a quadratic barrier function for obstacle avoidance,

$$h_k = (x_k(1) - x_{obs})^2 + (x_k(2) - y_{obs})^2 - r_{obs}^2, \tag{19}$$

where x_{obs}, y_{obs}, and r_{obs} describe x/y-coordinate and radius of the obstacle with $x_{obs} = -2m$, $y_{obs} = -2.25m$ and $r_{obs} = 1.5m$, shown as a red circle in Fig. 4. The start and target positions are $(-5, -5)$ and $(0, 0)$, which are labelled as blue and red diamonds in Fig. 4, respectively.

1) Comparison with DCLF-DCBF: In order to compare the performance between our proposed MPC-CBF and DCLF-DCBF, we develop a DCLF-DCBF controller for the same obstacle avoidance task, which is based on (9).

For the design of DCLF-DCBF, we use R in (17) as H in (9) to ensure that we have the same penalty on inputs. The discrete-time CBF constraints of DCLF-DCBF are the same as the ones used in MPC-CBF. Since the terminal cost p is the control Lyapunov function of model predictive control, we choose p in MPC-CBF example as the control Lyapunov function that is used to construct discrete-time CLF constraints in (9). Based on these choices, it is fair to compare a DCLF-DCBF controller with a MPC-CBF controller.

The simulation result of MPC-CBF and DCLF-DCBF comparison is shown in Fig. 4a, 4b and 4c. The trajectory is denoted in black line with small circles representing each step. The trajectory of DCLF-DCBF controller with $\gamma = 0.4$ is presented in Fig. 4a, where the system does not start to avoid the obstacle until it is close to it. Fig. 4b shows the trajectory for MPC-CBF controller with horizon $N = 1$ and γ = 0.4. This trajectory is similar to that of DCLF-DCBF controller. Based on our analysis in Sec. III-D, the performances of DCLF-DCBF and MPC-CBF with $N = 1$ are almost the same, which is validated in this simulation. Fig. 4c shows the trajectory of MPC-CBF controller with horizon $N = 8$ and $\gamma = 0.4$. We can see that this controller can drive the system to avoid the obstacle earlier than the DCLF-DCBF controller. Also, among these three controllers, it is the only one that can reach the goal position in the limited simulation time.

2) Comparison with MPC-DC: A MPC-DC controller is developed based on (2) using the same parameters as MPC-CBF presented before except the discrete-time CBF constraint, which is replaced by a Euclidean norm distance constraint g shown in (2e). The function g has the same expression as h, defined in (19).

Fig. 4d and 4e show the simulation result of the comparison between MPC-CBF and MPC-DC. In Fig. 4d, MPC-CBF controllers with γ = 0.1, 0.2, 0.3, 1.0 are described in blue, orange, yellow and purple lines, respectively. The trajectory

controller	status	N	γ	mean/std (s)	min (s)	max (s)	min dist w.r.t. obs	cost integral
MPC-CBF	completed	5	0.10	0.028±0.012	0.014	0.057	1.483	7.620
MPC-CBF	completed	5	0.20	0.028±0.011	0.013	0.064	0.791	7.464
MPC-CBF	completed	5	0.30	0.028±0.011	0.013	0.061	0.441	8.314
MPC-CBF	completed	5	0.40	0.028±0.011	0.013	0.059	0.288	8.292
MPC-CBF	completed	5	0.50	0.028±0.010	0.012	0.053	0.110	8.813
controller	status	N		mean/std (s)	min (s)	max (s)	min dist w.r.t. obs	cost integral
MPC-DC	completed	7		0.033±0.013	0.014	0.065	0.000	9.102
MPC-DC	completed	15		0.048±0.016	0.027	0.084	0.000	8.537
MPC-DC	completed	30		0.062±0.031	0.018	0.136	0.000	8.528

TABLE I: MPC-DC and MPC-CBF benchmark in terms of prediction horizon, lap time, minimal distance to obstacle and cost integral.

of MPC-DC controller is shown in black dashed line. As γ decreases, the system starts to avoid the obstacle earlier, which means a smaller safe set as analyzed in Sec. III-C, while on the other hand the trajectory of MPC-DC is the closest to the obstacle. We also notice that the trajectories of MPC-DC and MPC-CBF with $\gamma = 1$ are almost the same, which validates our analysis in Sec. III-E.

The trajectories of MPC-CBF controllers with $N = 5$ and different choices of γ and MPC-DC controllers with different values of horizon N are shown in Fig. 4e. We notice that MPC-CBF controller with smaller γ and MPC-DC with larger horizon N can make the system avoid obstacles earlier. This verifies our analysis in Sec. III-C, since smaller γ in MPC-CBF and larger horizon N in MPC-DC make the trajectory deviate from obstacles earlier. We also observe that even with an extremely large horizon N, e.g. $N = 30$, the system only has noticeable obstacle avoidance behavior when it is close to obstacles. In contrary, a relatively small γ is able to make the system avoid obstacles even far away from obstacles.

In Fig. 4e, MPC-CBF with $N = 5$ and $\gamma = 0.25$ starts to turn to avoid the obstacle with a similar behavior as MPC-DC with $N = 7$. This property is discussed in Remark 8. Since discrete-time CBF enforces the invariance of safe set, it allows a smaller N for MPC-CBF with a smaller γ to achieve a comparable performance as MPC-DC with a larger N.

In Table I, we benchmark the MPC-CBF and MPC-DC in terms of prediction horizon, computation time, minimal distance to the obstacle and cost integral $\sum_k u_k^T u_k \Delta t$ over the trajectory. We can observe that less prediction horizon of MPC-CBF leads to less computational time. MPC-DC always reaches to the boundary of the obstacle, however, MPC-CBF could automatically set a safety margin depending on different choices of γ. For this specific scenario, the cost integral over the trajectory of MPC-CBF is generally less than the one of MPC-DC.

B. Competitive Car Racing

We have evaluated the MPC-CBF design using a 2D double integrator and compared its performance with DCLF-DCBF and MPC-DC. We proceed to implement MPC-CBF in a more complex scenario: competitive car racing. In some early car racing control [21], [22], they only consider static obstacles on the track while we deal with dynamic obstacles such as other cars using MPC-CBF.

1) Vehicle Model: We use curvilinear coordinates to describe vehicle states of the ego and other cars in a racing competition. In this paper, we use the nonlinear lateral vehicle dynamics model in [23, Ch. 2] for system dynamics,

$$x_{t+1} = f(x_t, u_t), \tag{20}$$

where x_t and u_t represent the state and input of the vehicle at time step t and their definitions are as follow,

$$x_t = [v_{x_t},\ v_{y_t},\ \phi_t,\ e_{\phi_t},\ e_{y_t},\ s_t]^T,\ u_t = [a_t,\ \delta_t]^T, \tag{21}$$

where s_t represents the curvilinear distance travelled along the centerline of the track, e_{y_t} and e_{ϕ_t} are the deviation distance and heading angle error between vehicle and path. v_{x_t}, v_{y_t}, ϕ_t are the vehicle longitudinal velocity, lateral velocity and yaw rate in the curvilinear coordinates, respectively. A representation of the state in the curvilinear coordinate is shown in Fig. 5. The inputs are longitudinal acceleration a_t and steering angle δ_t. In the car racing, we assume to have K racing cars competing with the ego car, and we use the superscript i to distinguish the i-th ($i = 1, 2, ..., K$) competiting vehicle from the ego one, shown in Fig. 5.

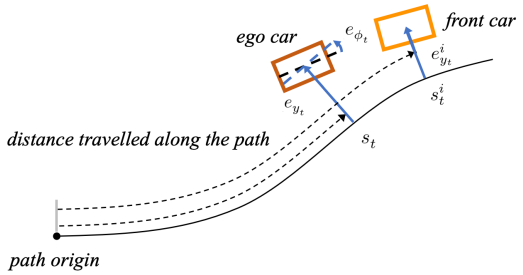


Fig. 5: Representation of the ego car and front car in the curvilinear coordinate frame.

2) Control Design: A MPC-CBF is developed for this competitive car racing example using (10). The stage cost function is designed as follows,

$$\begin{aligned} q(x_{t+k|t}, u_{t+k|t}) =& (x_{t+k|t} - x_r)^T Q (x_{t+k|t} - x_r) \\ &+ u_{t+k|t}^T R u_{t+k|t} \end{aligned} \tag{22}$$

where $x_r = (v_t, 0, 0, 0, 0, 0)$, $Q = diag(10, 0, 0, 0, 0, 10)$ and $R = diag(1, 1)$. This cost function allows the ego car to track the centerline with a target speed v_t while minimizing the tracking error from the centerline.

The motion of overtaking other racing cars is considered as CBF constraints in (10e). At time step t, each CBF h_t^i

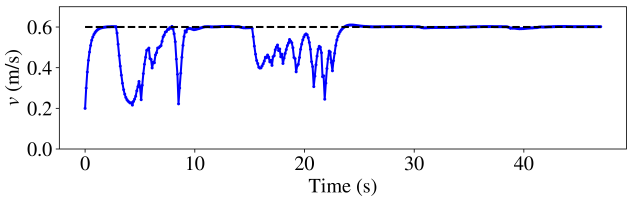


Fig. 6: Speed profile during the car racing competition in one lap of the simulation. The dashed black line shows the desired speed $v_t = 0.6$m/s.

represents the safety criterion between ego car at (s_t, e_{y_t}) and i-th other racing car at $(s_t^i, e_{y_t}^i)$, described in the curvilinear coordinates, shown in Fig. 5. We choose CBF in a quartic form as follows,

$$h_t^i = \frac{(s_t - s_t^i)^4}{(2l_1)^4} + \frac{(e_{y_t} - e_{y_t}^i)^4}{(2l_2)^4} - 1, \tag{23}$$

where we assume all racing cars including ego car hold the shape of rectangle with a length as $2l_1$ and width as $2l_2$. Notice that we assume that we have perfect estimation about (s_t, e_{y_t}) and $(s_t^i, e_{y_t}^i)$.

3) Simulation & Results: During the competition, we expect ego car to track the centerline with a target speed $v_t = 0.6$m/s. MPC-CBF with a horizon $N = 12$ updates at 10 Hz. The system dynamics is simulated at 1000 Hz and the controller sampling time is $0.1s$. Our ego vehicle is simulated with the nonlinear lateral vehicle dynamics model and we use the linearized dynamics along the centerline to formulate our control design.

In the simulation, we deploy several racing cars to compete with ego car. In order to better illustrate results, a snapshot of overtaking motion with a zoom-in view is shown in Fig. 1. Ego car begins with an initial speed $v_0 = 0.2$ m/s at the origin of the centerline and two other racing cars start in front of the ego car. Two cars are simulated to move at 0.2 m/s while keeping a constant distance deviation e_y^i from the centerline, where $e_y^1 = 0.1$ m and $e_y^2 = -0.1$ m. Fig. 1 demonstrates that the MPC-CBF allows ego car to safely race and overtake other cars in both left and right directions.

Fig. 6 shows the speed profile, where the dashed black line shows the desired speed. We can see that ego car always tries to catch up to the target speed during the competitive car racing. In Fig. 1, we could observe two motions of overtaking front racing cars. Since these two racing vehicles hold opposite distance deviations from the centerline, ego car overtakes them with right and left turns respectively. An illustrative animation is included in the code repository.

V. Conclusion

A safety-critical model predictive control design is proposed in this paper, where discrete-time control barrier function constraints are used in a receding horizon fashion to ensure safety. We present an analysis of its stability and feasibility, and describe its relation with MPC-DC and DCLF-DCBF. To verify our analysis, we use a 2D double integrator for obstacle avoidance, where we can see that MPC-CBF outperforms both MPC-DC and DCLF-DCBF. The proposed control logic is also applied to a more complex scenario: competitive car racing, where our ego car can race and safely overtake other racing cars.

Acknowledgement

We thank Ugo Rosolia for his insightful discussions.

References

[1] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *IEEE International Conference on Decision and Control*, 2014.

[2] A. Agrawal and K. Sreenath, "Discrete Control Barrier Functions for Safety-Critical Control of Discrete Systems with Application to Bipedal Robot Navigation," in *Robotics: Science and Systems*, 2017.

[3] F. R. Hogan and A. Rodriguez, "Reactive planar non-prehensile manipulation with hybrid model predictive control," *The International Journal of Robotics Research*, 2020.

[4] N. Scianca, D. De Simone, L. Lanari, and G. Oriolo, "MPC for humanoid gait generation: Stability and feasibility," *IEEE Transactions on Robotics*, 2020.

[5] T. D. Son and Q. Nguyen, "Safety-critical control for non-affine nonlinear systems with application on autonomous vehicle," in *IEEE International Conference on Decision and Control*, 2019.

[6] U. Rosolia and A. D. Ames, "Multi-rate control design leveraging control barrier functions and model predictive control policies," *arXiv preprint arXiv:2004.01761*, 2020.

[7] Y. Yoon, J. Shin, H. J. Kim, Y. Park, and S. Sastry, "Model-predictive active steering and obstacle avoidance for autonomous ground vehicles," *Control Engineering Practice*, vol. 17, no. 7, pp. 741–750, 2009.

[8] J. V. Frasch, A. Gray, M. Zanon, H. J. Ferreau, S. Sager, F. Borrelli, and M. Diehl, "An auto-generated nonlinear MPC algorithm for real-time obstacle avoidance of ground vehicles," in *European Control Conference*, 2013, pp. 4136–4141.

[9] V. Turri, A. Carvalho, H. E. Tseng, K. H. Johansson, and F. Borrelli, "Linear model predictive control for lane keeping and obstacle avoidance on low curvature roads," in *International Conference on Intelligent Transportation Systems*, 2013.

[10] U. Rosolia, S. De Bruyne, and A. G. Alleyne, "Autonomous vehicle control: A nonconvex approach for obstacle avoidance," *IEEE Transactions on Control Systems Technology*, 2016.

[11] X. Zhang, A. Liniger, and F. Borrelli, "Optimization-based collision avoidance," *IEEE Transactions on Control Systems Technology*, 2020.

[12] R. Grandia, A. J. Taylor, A. Singletary, M. Hutter, and A. D. Ames, "Nonlinear Model Predictive Control of Robotic Systems with Control Lyapunov Functions," in *Robotics: Science and Systems*, 2020.

[13] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *European Control Conference*, 2019.

[14] G. Wu and K. Sreenath, "Safety-critical and constrained geometric control synthesis using control lyapunov and control barrier functions for systems evolving on manifolds," in *American Control Conference*, 2015.

[15] Q. Nguyen, A. Hereid, J. W. Grizzle, A. D. Ames, and K. Sreenath, "3d dynamic walking on stepping stones with control barrier functions," in *IEEE International Conference on Decision and Control*, 2016.

[16] A. G. Wills and W. P. Heath, "Barrier function based model predictive control," *Automatica*, vol. 40, no. 8, pp. 1415–1422, 2004.

[17] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[18] G. Lars and P. Jürgen, "Nonlinear model predictive control theory and algorithms," 2011.

[19] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[20] L. T. Biegler and V. M. Zavala, "Large-scale nonlinear programming using ipopt: An integrating framework for enterprise-wide dynamic optimization," *Computers & Chemical Engineering*, 2009.

[21] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1: 43 scale rc cars," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 628–647, 2015.

[22] R. Verschueren, S. De Bruyne, M. Zanon, J. V. Frasch, and M. Diehl, "Towards time-optimal race car driving using nonlinear mpc in real-time," in *53rd IEEE conference on decision and control*. IEEE, 2014.

[23] R. Rajamani, *Vehicle dynamics and control*. Springer Science & Business Media, 2011.