IEEE CSS

Learning-Enabled Iterative Convex Optimization for Safety-Critical Model Predictive Control

SHUO LIU[1] (Student Member, IEEE), ZHE HUANG[1], JUN ZENG[2] (Member, IEEE), KOUSHIL SREENATH [2] (Senior Member, IEEE), AND CALIN A. BELTA [3] (Fellow, IEEE)

[1]Department of Mechanical Engineering, Boston University, Boston, MA 02215 USA
[2]Department of Mechanical Engineering, University of California, Berkeley, Berkeley, CA 94720 USA
[3]Departments of Electrical and Computer Engineering and Computer Science, University of Maryland, College Park, MD 20742 USA

CORRESPONDING AUTHOR: SHUO LIU (e-mail: liushuo@bu.edu)

ABSTRACT Safety remains a central challenge in control of dynamical systems, particularly when the boundaries of unsafe sets are complex (e.g., nonconvex, nonsmooth) or unknown. This paper proposes a learning-enabled framework for safety-critical Model Predictive Control (MPC) that integrates Discrete-Time High-Order Control Barrier Functions (DHOCBFs) with iterative convex optimization. Unlike existing methods that primarily address CBFs of relative degree one with fully known unsafe set boundaries, our approach generalizes to arbitrary relative degrees and addresses scenarios where only samples are available for the unsafe set boundaries. We extract pixels from unsafe set boundaries and train a neural network to approximate local linearizations. The learned models are incorporated into the linearized DHOCBF constraints at each time step within the MPC framework. An iterative convex optimization procedure is developed to accelerate computation while maintaining formal safety guarantees. The benefits of computational performance and safe avoidance of obstacles with diverse shapes are examined and confirmed through numerical results. By bridging model-based control with learning-based environment modeling, this framework advances safe autonomy for discrete-time systems operating in complex and partially known settings.

INDEX TERMS Constrained control, optimal control, control barrier function, nonlinear predictive control, machine learning.

I. INTRODUCTION
A. MOTIVATION
Safety-critical optimal control is a central problem in autonomous systems. For example, reaching a goal while avoiding obstacles and minimizing energy can be formulated as a constrained optimal control problem by using continuous-time Control Barrier Functions (CBFs) [1], [2]. By dividing the timeline into small intervals, the problem is reduced to a (possibly large) number of quadratic programs, which can be solved at real-time speeds. However, this approach may be overly aggressive because it does not anticipate future events.

In [3], the authors use discrete-time CBFs as constraints inside a discrete-time Model Predictive Control (MPC) prob-

lem, which provides a safer control policy as it incorporates information about future states over a receding horizon to establish greater safety margins. However, the computational time is large and grows rapidly with a larger horizon, since the optimization is usually nonlinear and non-convex. An additional problem with this nonlinear MPC formulation is the feasibility of the optimization. Moreover, it is difficult to obtain the CBFs from complex environments, i.e., the boundaries of unsafe sets in the environment are difficult to describe with known equations.

In this article, we address the above challenges by proposing a convex MPC with linearized, discrete-time CBFs, under an iterative approach. In contrast with the real-time iteration (RTI) approach introduced in [4], which solves the

problem through iterative Newton steps, and the learning MPC (LMPC) framework [50], which builds invariant safe sets from iteratively stored trajectories and requires repeated executions of the same task with an initial feasible trajectory for convergence, our approach iteratively solves a convex MPC at each time step by constructing linear safety constraints from unsafe sets, enabling safety from the first execution with no prior data or trial runs needed. While classical constrained MPC [52] addresses linear dynamics with polyhedral constraints, our formulation embeds high-order CBF constraints, a learned boundary detector, and bounded slack variables, enabling safety constraint enforcement for iteratively linearized systems and unsafe sets. If the boundaries of the unsafe sets are simple and known, we directly linearize them; if complex and unknown, a Deep Neural Network (DNN), referred to as a Safety Boundary Detector (SBD), predicts local linear approximations, which are then transformed into linearized, discrete-time CBFs to ensure safety.

B. RELATED WORK

1) Model Predictive Control (MPC): This paper can be seen in the context of MPC with safety constraints. MPC is widely used in control systems, such as controller design in robotic manipulation and locomotion [5], [6], to obtain a control strategy as a solution to an optimization problem. Stability was achieved in [7] by incorporating Discrete-time Control Lyapunov Functions (DCLFs) into a general MPC-based optimization problem to realize real-time control on a robotic system with limited computational resources. The authors of [8] emphasize safety in robot design and deployment. When safety means obstacle avoidance, some works achieve it through the introduction of repelling functions [1], [9], while others regard obstacle avoidance as one concrete example of safety criteria for robots [10]–[12]. In these works, safety is formulated as constraints in optimization problems.

2) Control Barrier Functions (CBFs): CBFs are used to enforce safety, i.e., rendering a set forward invariant for a dynamical system. Recent studies have demonstrated that to ensure the stabilization of an affine control system, while also adhering to safety constraints and control limitations, CBFs can be integrated with Control Lyapunov Functions (CLFs). This integration facilitates the creation of a sequence of single-step optimization programs [1], [2], [13], [14]. If the cost is quadratic, the optimizations are quadratic programs (QP), and the solutions can be deployed in real time [1], [15]. For safety constraints expressed using functions with high relative degree with respect to the dynamics of the system, exponential CBFs [16] and high-order CBFs (HOCBFs) [17]–[19] were proposed. Adaptive, robust, stochastic and feasibility-guaranteed versions of safety-critical control with CBFs were introduced in [20]–[25]. Discrete-time CBFs (DCBFs) were introduced in [26] as a means to enable safety-critical control for discrete-time systems. They were used in a nonlinear MPC (NMPC)

framework to create NMPC-DCBF [3], wherein the DCBF constraint was enforced through a predictive horizon. Generalized discrete-time CBFs (GCBFs) and discrete-time high-order CBFs (DHOCBFs) were proposed in [27] and [28] respectively, where the DCBF constraint only acted on the first time-step, i.e., a single-step constraint. MPC with DCBF has been used in various fields, such as autonomous driving [29] and legged robotics [30]. To improve the optimization feasibility, a decay-rate relaxing technique [31] was introduced for NMPC with DCBF [32] for every time-step to simultaneously boost safety and feasibility. However, the overall optimization remains nonlinear and non-convex, resulting in slow processing times for systems with large horizons and nonlinear dynamics (see [32], which is limited to relative-degree one). In this paper, we extend the relaxation techniques introduced in [32] to accommodate high relative-degree CBFs, and largely improve the computational time compared to all existing work.

3) Learning-Based CBFs: Many CBF-based methods require explicit equations for the boundaries of the safety sets. When these are not available, they can be learned. In [33], the authors utilized a Support Vector Machine (SVM) to parameterize a CBF. They employed a supervised learning approach to classify regions of the state-space as either safe or unsafe. In [34], the authors proposed a method that incrementally learns a linear CBF by grouping expert demonstrations into linear subspaces and creating low-dimensional representations through fitting. The authors of [35] proposed to construct polynomial CBFs to represent complex obstacles or unreachable regions using logistic regression. However, the above works do not provide theoretical guarantees of correctness of the learned CBFs. The work in [36] proposed and analyzed an optimization-based method for learning a CBF that provides provable safety guarantees given some mild assumptions. Nevertheless, it still requires that the equation of the boundary of the safe set is known. Recently, the authors of [37] defined a learning-based discriminating hyperplane trained on trajectory data from a black-box system to eliminate the dependence on any specific equations for the safe sets. However, the trained hyperplane can only work for that system. We use a DNN to predict the nearest boundary point; the outward normal is then computed from this point, defining a local linear (tangent) boundary of the safe set that we embed as a DHOCBF. The run-time constraint is linear, but mapping raw geometry to the nearest point is nonlinear, which motivates using a DNN.

4) Beyond CBF-Based Methods: Many works tackle complex or unknown obstacles—reachability-based safety filters [47], planning on reconstructed or occupancy maps [48], and certified MPC [49]. They typically scale poorly beyond low dimensions or fixed maps, lack per-step safety certificates under the true dynamics and input bounds, or require costly recomputation or large polyhedral encodings. Although safe reinforcement learning [53], temporal logic based planning

[54], and multi-agent safety methods [55] have considered high-dimensional problems, these approaches generally rely on set-based abstractions or task-level specifications rather than explicit handling of arbitrary and irregular obstacle geometries. Hence, they are not directly applicable to environments with complex-shaped obstacles. In contrast, we achieve fast, convex per-step safety constraints via DHOCBFs in convex MPC with learned linear boundaries from complex-shaped obstacles—without global discretization of the environment.

C. CONTRIBUTIONS

This paper introduces a novel approach to safety-critical MPC that is significantly faster than existing approaches and enhances safety within complex environments where unsafe set boundaries are irregular and poorly defined. The core of our methodology is a learning-based technique for predicting and linearizing the boundaries of unsafe sets. The key contributions of our work are as follows:

- A control strategy that ensures safety by enforcing constraints derived from linearized DHOCBFs. We relax these constraints using bounded slack variables to improve feasibility while preserving safety. (Sec. IV-C)
- A model predictive control framework that integrates linearized system dynamics and DHOCBFs as constraints within a convex optimization problem. This problem is solved iteratively, achieving fast computation speeds ideal for real-time applications. (Sec. IV)
- A machine learning-based method to accurately predict the linearized boundaries of unsafe sets. The complement of these boundaries defines linearized approximations of safe sets, which are then incorporated into DHOCBF constraints. This is particularly effective in scenarios where unsafe set boundaries are complex, irregular, or otherwise difficult to model with analytical equations (e.g., unknown boundaries). (Sec. IV-C-2)

This work significantly extends our previous conference paper [38], where we first introduced the idea of linearizing DHOCBFs under the assumption of known and simple unsafe set boundaries (e.g., circles). In this manuscript, we go beyond that by developing a general framework to learn linearized approximations of complex, unknown boundaries and integrating this framework into a convex finite-time optimal control scheme. Specifically, the learned linearized boundaries are embedded as DHOCBFs to ensure safety requirements. Furthermore, we provide the details of the technique based on slack variables within linearized DHOCBFs for improving both feasibility and safety — these details were omitted in [38]. Compared with [38], which considered only a unicycle model, this work additionally includes a higher-dimensional vehicle model to examine the scalability of the proposed framework, and validates it in simulations where both robots navigate narrow passages with convex and nonconvex obstacles, achieving markedly

Acronyms	Meaning
CBF	Control Barrier Function
DCBF	Discrete-Time Control Barrier Function
DHOCBF	Discrete-Time High-Order Control Barrier Function
MPC	Model Predictive Control
NMPC	Nonlinear Model Predictive Control
iMPC	Iterative Model Predictive Control
DNN	Deep Neural Network
SBD	Safety Boundary Detector
CFTOC	Convex Finite-Time Optimal Control
MSE	Mean Squared Error

better performance in computational efficiency, safety, and feasibility than existing methods.

II. PRELIMINARIES

In this work, safety is defined as forward invariance of a set $\mathcal{C}$, i.e., a system is said to be *safe* if it stays in $\mathcal{C}$ for all times, given that it is initialized in $\mathcal{C}$. We consider the set $\mathcal{C}$ as the superlevel set of a function $h : \mathbb{R}^n \to \mathbb{R}$:

$$\mathcal{C} := \{\mathbf{x} \in \mathbb{R}^n : h(\mathbf{x}) \geq 0\}. \tag{1}$$

We consider a discrete-time control system in the form

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t), \tag{2}$$

where $\mathbf{x}_t \in \mathcal{X} \subset \mathbb{R}^n$ represents the state of the system at time step $t \in \mathbb{N}$, $\mathbf{u}_t \in \mathcal{U} \subset \mathbb{R}^q$ is the control input, and the function $f : \mathbb{R}^n \times \mathbb{R}^q \to \mathbb{R}^n$ is locally Lipschitz.

Definition 1 (Relative degree [39]). *The output $y_t = h(\mathbf{x}_t)$ of system* (2) *is said to have relative degree m if*

$$y_{t+i} = h(\bar{f}_{i-1}(f(\mathbf{x}_t, \mathbf{u}_t))), \ i \in \{1, 2, \ldots, m\},$$
$$s.t. \ \frac{\partial y_{t+m}}{\partial \mathbf{u}_t} \neq \boldsymbol{0}_q, \frac{\partial y_{t+i}}{\partial \mathbf{u}_t} = \boldsymbol{0}_q, \ i \in \{1, 2, \ldots, m-1\}, \tag{3}$$

i.e., m is the number of steps (delay) in the output y_t in order for any component of the control input $\mathbf{u}_t$ to explicitly appear ($\boldsymbol{0}_q$ is the zero vector of dimension q).

In the above definition, we use $\bar{f}(\mathbf{x})$ to denote the uncontrolled state dynamics $f(\mathbf{x}, 0)$. The subscript i of function $\bar{f}(\cdot)$ denotes the i-times recursive compositions of $\bar{f}(\cdot)$, i.e., $\bar{f}_i(\mathbf{x}) = \underbrace{\bar{f}(\bar{f}(\ldots, \bar{f}(\bar{f}_0(\mathbf{x}))))}_{i\text{-times}}$ with $\bar{f}_0(\mathbf{x}) = \mathbf{x}$. Note that in discrete time the relative degree m only means that the control input influences the safety function after m steps; this look-ahead is not the same as a physical actuator delay.

We assume that $h(\mathbf{x})$ has relative degree m with respect to system (2) based on Def. 1. Starting with $\psi_0(\mathbf{x}_t) := h(\mathbf{x}_t)$, we define a sequence of discrete-time functions $\psi_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, m$ as:

$$\psi_i(\mathbf{x}_t) := \triangle\psi_{i-1}(\mathbf{x}_t) + \alpha_i(\psi_{i-1}(\mathbf{x}_t)), \tag{4}$$

where $\triangle\psi_{i-1}(\mathbf{x}_t) := \psi_{i-1}(\mathbf{x}_{t+1}) - \psi_{i-1}(\mathbf{x}_t)$, and $\alpha_i(\cdot)$ denotes the ith class κ function which satisfies $\alpha_i(\psi_{i-1}(\mathbf{x}_t)) \leq$

$\psi_{i-1}(\mathbf{x}_t)$ for $i = 1, \ldots, m$. A sequence of sets $\mathcal{C}_i$ is defined based on (4) as

$$\mathcal{C}_i := \{\mathbf{x} \in \mathbb{R}^n : \psi_i(\mathbf{x}) \geq 0\}, \ i \in \{0, \ldots, m-1\}. \quad (5)$$

Imposing $\psi_i(\mathbf{x}_t) \geq 0$ at time t enforces safety i steps ahead, since Eq. (4) sequentially ties ψ_i to $\mathbf{x}_t, \ldots, \mathbf{x}_{t+i}$.

Definition 2 (DHOCBF [28]). *Let $\psi_i(\mathbf{x}), \ i \in \{1, \ldots, m\}$ be defined by (4) and $\mathcal{C}_i, \ i \in \{0, \ldots, m-1\}$ be defined by (5). A function $h : \mathbb{R}^n \to \mathbb{R}$ is a Discrete-time High-Order Control Barrier Function (DHOCBF) with relative degree m for system (2) if there exist $\psi_m(\mathbf{x})$ and $\mathcal{C}_i$ such that*

$$\psi_m(\mathbf{x}_t) \geq 0, \ \forall x_t \in \mathcal{C}_0 \cap \cdots \cap \mathcal{C}_{m-1}, t \in \mathbb{N}. \quad (6)$$

Theorem 1 (Safety Guarantee [28]). *Given a DHOCBF $h(\mathbf{x})$ from Def. 2 with corresponding sets $\mathcal{C}_0, \ldots, \mathcal{C}_{m-1}$ defined by (5), if $\mathbf{x}_0 \in \mathcal{C}_0 \cap \cdots \cap \mathcal{C}_{m-1}$, then any Lipschitz controller $\mathbf{u}_t$ that satisfies the constraint in (6), $\forall t \geq 0$ renders $\mathcal{C}_0 \cap \cdots \cap \mathcal{C}_{m-1}$ forward invariant for system (2), i.e., $\mathbf{x}_t \in \mathcal{C}_0 \cap \cdots \cap \mathcal{C}_{m-1}, \forall t \geq 0$.*

Remark 1. *The function $\psi_i(\mathbf{x})$ in (4) is called an ith order Discrete-time CBF (DCBF). If the constraints in an optimal control problem include only DCBF constraints ($\psi_i(\mathbf{x}_t) \geq 0$), we must formulate the DCBF constraints up to the mth order to ensure that the control input $\mathbf{u}_t$ is explicitly represented based on Def. 1. However, if the constraints also include the system constraint (2), where the control input $\mathbf{u}_t$ is already explicitly represented, we can flexibly select a suitable order for the DCBF constraints to minimize computational demands. In other words, the highest order for DCBF could be m_{cbf} with $m_{cbf} \leq m$. We can simply define an ith order DCBF $\psi_i(\mathbf{x})$ in (4) as*

$$\psi_i(\mathbf{x}_t) := \triangle\psi_{i-1}(\mathbf{x}_t) + \gamma_i\psi_{i-1}(\mathbf{x}_t), \quad (7)$$

where $\alpha(\cdot)$ is linear and $0 < \gamma_i \leq 1, i \in \{1, \ldots, m_{cbf}\}$.

The expression presented in (7) adheres to the structure of the first-order DCBF introduced in [26] and can be applied to define a DHOCBF with any relative degree for the classical constrained, safety-critical optimal control problem:

$$J(\mathbf{u}_t, \mathbf{x}_k) = \min_{\mathbf{u}_t} \mathbf{u}_t^T\mathbf{u}_t + \sum_{k=t}^{t+1}(\mathbf{x}_k - \mathbf{x}_e)^T(\mathbf{x}_k - \mathbf{x}_e) \quad (8a)$$

$$s.t. \ \ \mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t), \quad (8b)$$

$$\psi_{i-1}(\mathbf{x}_{t+1}) \geq (1 - \gamma_i)\psi_{i-1}(\mathbf{x}_t), \ 0 < \gamma_i \leq 1, \quad (8c)$$

$$\mathbf{u}_t \in \mathcal{U} \subset \mathbb{R}^q, \ \mathbf{x}_k \in \mathcal{X} \subset \mathbb{R}^n, \quad (8d)$$

where (8a) defines the objective function of the optimization problem, $\mathbf{x}_e$ denotes a reference state, $0 \leq t \leq T, t \in \mathbb{N}$ and the function $\psi_{i-1}, i \in \{1, \ldots, m_{\text{cbf}}\}$ in (8c) ensures that the state $\mathbf{x}_k$ of system (8b) stays within a safe set $\mathcal{C}$ according to (1). The constraint (8d) bounds the control input and state, which may conflict with constraints (8b) and (8c), leading to infeasibility. Note that solving the problem described above can only yield the current optimal control input, resulting in

a greedy control policy that considers short-term (one-step) safety and may overlook safer solutions. As discussed in [3], there exists a fundamental trade-off between safety and feasibility in control design, and enhancing both simultaneously is challenging. In [3], the authors show that MPC with DCBFs can give a safer control policy, as it takes future state and control information into account. A version of nonlinear MPC that incorporates DCBFs (called NMPC-DCBF) with a relaxation technique that simultaneously enhances feasibility and safety was developed in [32]. This approach, although it can flexibly incorporate the system's physical constraints and safety requirements as soft or hard constraints into the control strategy, often leads to non-convex optimizations, resulting in inherently high computational complexity. In this paper, we show how to linearize the system and DCBF constraints to obtain a convex finite-time optimal control (CFTOC) framework within an MPC setting, which significantly improves computational efficiency.

III. PROBLEM FORMULATION AND APPROACH

Our objective is to find a closed-loop control strategy for system (2) over a time interval $[0, T]$ that minimizes the deviation from a reference state and satisfies safety requirements and constraints on states and control inputs.

Safety Requirement: System (2) should always satisfy a safety requirement of the form:

$$h(\mathbf{x}_t) \geq 0, \ \mathbf{x}_t \in \mathbb{R}^n, \ 0 \leq t \leq T, \quad (9)$$

where $h : \mathbb{R}^n \to \mathbb{R}$.

Control and State Limitation Requirements: The controller $\mathbf{u}_t$ and state $\mathbf{x}_t$ must satisfy (8d) for $0 \leq t \leq T$.

Objective: We consider the following cost:

$$J(\mathbf{u}_{t,k}, \mathbf{x}_{t,k}) = \sum_{k=0}^{N-1} \mathbf{u}_{t,k}^T\mathbf{u}_{t,k} + \sum_{k=0}^{N}(\mathbf{x}_{t,k} - \mathbf{x}_e)^T(\mathbf{x}_{t,k} - \mathbf{x}_e), \quad (10)$$

over a receding horizon $N < T$, where $\mathbf{x}_{t,k}, \mathbf{u}_{t,k}$ are the state and input predictions (according to the system's dynamics) at time $t + k$ made at the current time t, $0 \leq t \leq T$. We denote $\mathbf{x}_{t,0} = \mathbf{x}_t$, $\mathbf{u}_{t,0} = \mathbf{u}_t$ and the reference state as $\mathbf{x}_e$. A control policy is *feasible* if all the constraints guaranteeing the aforementioned requirements are mutually non-conflicting for all $0 \leq t \leq T$. In this paper, we consider the following problem:

Problem 1. *Find a feasible control policy for system (2) such that the safety requirement, control and state limitations are satisfied, and the cost (10) is minimized. The safety requirement is defined with respect to unsafe sets whose boundaries may be either explicitly **known** (e.g., analytic functions) or **unknown** and only available through samples.*

To satisfy the safety requirement, the authors of [32] incorporated the DCBF constraint

$$h(\mathbf{x}_{t,k+1}) \geq \omega_{t,k}(1 - \gamma)h(\mathbf{x}_{t,k}), \ 0 < \gamma \leq 1 \quad (11)$$

into MPC with a relaxation variable $\omega_{t,k} \in \mathbb{R}$, which simultaneously enhances feasibility and safety. They also formulated (2) and (8d) as constraints in the MPC to meet additional requirements, which results in the entire optimization problem being nonconvex, as (11) and (2) are often nonlinear. This method, referred to as NMPC-DCBF, can lead to slow processing times for systems with large horizons. Another issue is that, when the boundary of the unsafe set is difficult to express with explicit equations, it becomes challenging to obtain a DCBF candidate $h(\mathbf{x})$. Some approaches [3], [40] exist for the case when the boundaries of the unsafe sets are circular or polytopic. In many real-world situations, this assumption is restrictive. Meanwhile, using a conservative known surrogate (e.g., polygon or circle) to replace the ground-truth obstacle set yields a smaller safe set (see e.g., [40], where the authors used NMPC to ensure safe path following for a unicycle robot). However, due to the presence of numerous polytopic obstacles in the map, defining the safe region using linear equations corresponding to the edges of the polytopes results in a very small safe region. This limited the NMPC to handling only nearby obstacles, making distant ones difficult to address.

Approach: In all tested scenarios, the proposed iMPC-DHOCBF achieved a remarkable online computation speedup of tens to hundreds of times over the NMPC baseline, while consistently preserving safety and maintaining high feasibility rates under tight input bounds. These results highlight that our approach not only accelerates online optimization but also ensures consistent safety performance. The key to these gains lies in transforming the original nonconvex safety constraints into efficiently solvable convex forms via a Safety Boundary Detector (SBD). This SBD obtains linearized unsafe set boundaries either directly, when they are simple and **known**, or via a trained deep neural network, when they are complex or **unknown**. Each linearized boundary equation serves as a DHOCBF candidate $h(\mathbf{x})$ for constructing its corresponding constraint. Since each unsafe set corresponds to a single DHOCBF constraint—unlike in [40], where multiple DCBF constraints are used per unsafe set—the resulting safe set, defined as the intersection of all DHOCBF constraints, achieves broader coverage with fewer constraints. We integrate these constraints into an MPC framework that also incorporates state and input bounds, relaxing each DHOCBF constraint with a bounded slack variable to improve feasibility while still guaranteeing safety. The system dynamics constraint is additionally linearized, yielding a fast-computing convex optimization problem designed to be solved iteratively, with decision variable errors reduced after a certain number of iterations.

IV. ITERATIVE CONVEX MPC WITH DHOCBF

In this section, we present an iterative convex MPC for general DHOCBFs defined in Sec. II.

A. ITERATIVE CONVEX MPC

Our proposed approach, which is depicted in Fig. 1, involves iterative optimizations at each time step t, which is denoted as *iterative* MPC-DHOCBF (iMPC-DHOCBF). Our iterative optimization problem contains three parts for each iteration j: (1) solving a CFTOC problem with linearized dynamics and DHOCBF to get optimal values of states and inputs $\mathbf{X}_t^{*,j} = [\mathbf{x}_{t,0}^{*,j}, \ldots, \mathbf{x}_{t,N}^{*,j}], \mathbf{U}_t^{*,j} = [\mathbf{u}_{t,0}^{*,j}, \ldots, \mathbf{u}_{t,N-1}^{*,j}]$, (2) checking convergence, (3) updating the state and input vectors for next iteration, i.e., $\bar{\mathbf{X}}_t^{j+1} = \mathbf{X}_t^{*,j}, \bar{\mathbf{U}}_t^{j+1} = \mathbf{U}_t^{*,j}$. Notice that the open-loop trajectory with updated states $\bar{\mathbf{X}}_t^j = [\bar{\mathbf{x}}_{t,0}^j, \ldots, \bar{\mathbf{x}}_{t,N-1}^j]$ and inputs $\bar{\mathbf{U}}_t^j = [\bar{\mathbf{u}}_{t,0}^j, \ldots, \bar{\mathbf{u}}_{t,N-1}^j]$ is passed between iterations, which allows iterative linearization for both system dynamics and DHOCBF locally.

The iteration is finished when the convergence error function $e(\mathbf{X}_t^{*,j}, \mathbf{U}_t^{*,j}, \bar{\mathbf{X}}_t^j, \bar{\mathbf{U}}_t^j)$ is within a user-defined normalized convergence criterion. To restrict the number of iterations, we limit $j < j_{\max}$, where $j_{\max}$ denotes the maximum numbers of iterations. Therefore, the iterative optimization process halts upon reaching a local minimum for the cost function, meeting the convergence criterion, or when the iteration count equals $j_{\max}$. The optimized states $\mathbf{X}_t^*$ and inputs $\mathbf{U}_t^*$ from the last iteration are passed to the iMPC-DHOCBF for the next time instant as $\bar{\mathbf{U}}_{t+1}^0, \bar{\mathbf{X}}_{t+1}^0$. At each time step, we record the updated states $\mathbf{x}_{t+1}$ propagated by the system dynamics $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_{t,0}^*)$, enabling us to extract a closed-loop trajectory. Note that if we have the boundaries of unsafe sets, the SBD can directly linearize safe boundaries by finding the point nearest to the robot's current location and drawing a tangent surface at that point. If we do not have the boundaries of unsafe sets, SBD relies on a DNN that is trained before the start of the iMPC. The advantage of this approach is that we have pre-trained a DNN which can learn in advance information about all unsafe sets in the environment, especially when the boundaries of the unsafe sets cannot be accurately identified and described. Consequently, the time required to perform inference with this DNN for obtaining the linearized DHOCBF is essentially fixed in each iteration and can gradually decrease as adjustments are made to the DNN, ensuring the efficiency of solving each CFTOC.

B. LINEARIZATION OF DYNAMICS

At iteration j, a control vector $\mathbf{u}_{t,k}^j$ is obtained by linearizing the system around $\bar{\mathbf{x}}_{t,k}^j, \bar{\mathbf{u}}_{t,k}^j$:

$$\mathbf{x}_{t,k+1}^j - \bar{\mathbf{x}}_{t,k+1}^j = A_{t,k}^j(\mathbf{x}_{t,k}^j - \bar{\mathbf{x}}_{t,k}^j) + B_{t,k}^j(\mathbf{u}_{t,k}^j - \bar{\mathbf{u}}_{t,k}^j) + c_{t,k}^j, \quad (12)$$

where $c_{t,k}^j = f(\bar{\mathbf{x}}_{t,k}^j, \bar{\mathbf{u}}_{t,k}^j) - \bar{\mathbf{x}}_{t,k+1}^j$, $0 \le j < j_{\max}$; k and j represent open-loop time step and iteration indices, respectively. We also have

$$A_{t,k}^j = D_{\mathbf{x}} f(\bar{\mathbf{x}}_{t,k}^j, \bar{\mathbf{u}}_{t,k}^j), \ B_{t,k}^j = D_{\mathbf{u}} f(\bar{\mathbf{x}}_{t,k}^j, \bar{\mathbf{u}}_{t,k}^j), \quad (13)$$

where $D_{\mathbf{x}}$ and $D_{\mathbf{u}}$ denote the Jacobian of the system dynamics $f(\mathbf{x}, \mathbf{u})$ with respect to the state $\mathbf{x}$ and the input $\mathbf{u}$. This approach allows us to linearize the system at $(\bar{\mathbf{x}}_{t,k}^j, \bar{\mathbf{u}}_{t,k}^j)$
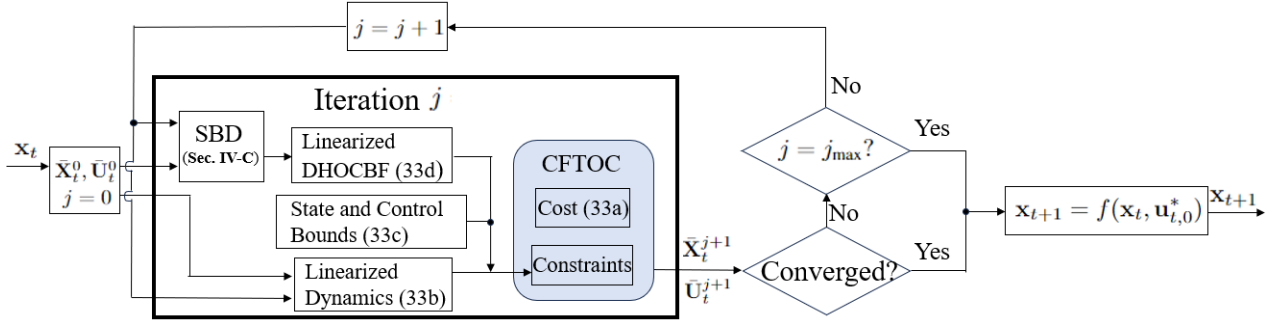
FIGURE 1: Schematic of the iterative process for solving convex MPC at time t. The safety boundary detector (SBD) generates linearized boundaries, which are converted into DHOCBF constraints. At each iteration, a convex finite-time optimal control (CFTOC) problem with linearized dynamics and quadratic cost is solved. The problem remains convex due to linear constraints and a positive semidefinite weight matrix. A convergence check determines whether to iterate or update the state $\mathbf{x}_{t+1}$.

locally between iterations. The convex system dynamics constraints are provided in (12) since all the nominal vectors $(\bar{\mathbf{x}}_{t,k}^j, \bar{\mathbf{u}}_{t,k}^j)$ at the current iteration are constant and constructed from the previous iteration $j-1$.

Remark 2 (*Warm Start*). *We linearize the safe set constraint by projecting the nominal prediction to the nearest boundary and enforcing the outward-normal half-space, which yields an approximation of the safe set. This approximation holds only if the nominal N-step rollout remains outside obstacles. In this case, we use a cheap safe warm start: for $t > 0$ we shift the previous optimal input sequence and pad the tail; for $t = 0$ we select from sampled admissible inputs whose nominal rollout ($\bar{\mathbf{X}}_0^0$) remains outside obstacles over the N-step horizon. This improves feasibility, while the DHOCBF constraints enforce closed-loop safety at each step.*

C. LINEARIZED SBD-GENERATED DCBF AND DHOCBF

*1) Safety Boundary Detector for **Known** Unsafe Boundaries:* In this section, we show how to linearize the DCBF up to the highest order with known unsafe boundary $h(\mathbf{x}) = 0$. At iteration j and time step $t + k$, as shown in Fig. 2, in order to linearize $h(\mathbf{x}_{t,k}^j)$, an explicit dashed line is projected in the state space to the nearest point $\tilde{\mathbf{x}}_{t,k}^j$ on the boundary of the unsafe set from each state $\bar{\mathbf{x}}_{t,k}^j$. Note that $\bar{\mathbf{x}}_{t,k}^j$ is the nominal state vector from iteration $j - 1$ for the linearization at iteration j, which means $\bar{\mathbf{x}}_{t,k}^j = \mathbf{x}_{t,k}^{j-1}$. The hyperplane (solid line in the 2D illustration) perpendicular to the dashed line and passing through the nearest point $\tilde{\mathbf{x}}_{t,k}^j$ is denoted as $h_\parallel(\mathbf{x}_{t,k}^j, \tilde{\mathbf{x}}_{t,k}^j)$. This allows us to incorporate the above process into the SBD to define a linearized safe set by $h_\parallel(\mathbf{x}_{t,k}^j, \tilde{\mathbf{x}}_{t,k}^j) \geq 0$, $\forall t \in \mathbb{N}$ by the green region. Note that $\tilde{\mathbf{x}}_{t,k}^j$ generally represents the optimized value of the minimum distance problem with distance function $h(\cdot)$ between $\bar{\mathbf{x}}_{t,k}^j$ and safe set $\mathcal{C}$. Since $h(\cdot)$ is known and continuous, for common shapes of unsafe sets, the expression of $\tilde{\mathbf{x}}_{t,k}^j$ as a function of $\bar{\mathbf{x}}_{t,k}^j$ is explicit. For example, when $h(\cdot)$ describes a l_2-norm function with the unsafe set being
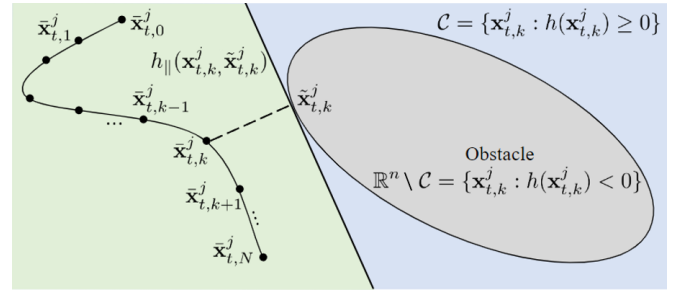

FIGURE 2: Linearization of DHOCBF: $h_\parallel(\mathbf{x}_{t,k}^j, \tilde{\mathbf{x}}_{t,k}^j) \geq 0$ represents the linearized safe set locally and is colored in green. Note that $h_\parallel(\mathbf{x}_{t,k}^j, \tilde{\mathbf{x}}_{t,k}^j) \geq 0$ guarantees $h(\mathbf{x}_{t,k}^j) \geq 0$ (colored in blue plus green), which ensures collision avoidance (outside the grey region).

a circular shape, $\tilde{\mathbf{x}}_{t,k}^j$ is exactly the intersection point of the line determined by $\bar{\mathbf{x}}_{t,k}^j$ and the center of the circular region with the circumference. The relative degree of $h_\parallel(\mathbf{x}_{t,k}^j, \tilde{\mathbf{x}}_{t,k}^j)$ with respect to system (2) is still m when the relative degree of $h(\mathbf{x}_{t,k}^j)$ is m. Thus, in order to guarantee safety with forward invariance based on Thm. 1 and Rem. 1, two sufficient conditions need to be satisfied: (1) the sequence of linearized DHOCBF $\tilde{\psi}_0(\cdot), \ldots, \tilde{\psi}_{m_{\text{cbf}}-1}(\cdot)$ is larger or equal to zero at the initial condition $\mathbf{x}_t$, and (2) the highest-order DCBF constraint $\tilde{\psi}_{m_{\text{cbf}}}(\mathbf{x}) \geq 0$ is always satisfied, where $\tilde{\psi}_i(\cdot)$ is defined as:

$$
\begin{aligned}
\tilde{\psi}_0(\mathbf{x}_{t,k}^j) &:= h_\parallel(\mathbf{x}_{t,k}^j, \tilde{\mathbf{x}}_{t,k}^j) \\
\tilde{\psi}_i(\mathbf{x}_{t,k}^j) &:= \tilde{\psi}_{i-1}(\mathbf{x}_{t,k+1}^j) - \tilde{\psi}_{i-1}(\mathbf{x}_{t,k}^j) + \gamma_i \tilde{\psi}_{i-1}(\mathbf{x}_{t,k}^j).
\end{aligned}
\tag{14}
$$

Here, we have $0 < \gamma_i \leq 1$, $i \in \{1, \ldots, m_{\text{cbf}}\}$, and $m_{\text{cbf}} \leq m$ (as in (7)). From Rem. 1, it follows that that m_{cbf} is not necessarily equal to m. A detailed discussion on this can be found in [27], [32].

An important issue is *feasibility*. It is possible that $\psi_i(\mathbf{x}_{t,k}^0) \geq 0$, $1 \leq i \leq m_{\text{cbf}} - 1$, with $k \in \{0, \ldots, N\}$ is not satisfed since the linearized DHOCBF functions $\tilde{\psi}_0(\cdot), \ldots, \tilde{\psi}_{m_{\text{cbf}}-1}(\cdot)$ are more conservative than the original

forms $\psi_0(\cdot), \ldots, \psi_{m_{cbf}-1}(\cdot)$. This problem can occur when the horizon is too large, or the linearization is too conservative. In order to handle this issue, we introduce a slack variable $\omega_{t,k,i}^j$ with a corresponding decay rate $(1-\gamma_i)$:

$$\tilde{\psi}_{i-1}(\mathbf{x}_{t,k+1}^j) \geq \omega_{t,k,i}^j(1-\gamma_i)\tilde{\psi}_{i-1}(\mathbf{x}_{t,k}^j), \ \omega_{t,k,i}^j \geq 0, \quad (15)$$

where $i \in \{1, \ldots, m_{cbf}\}$. The slack variable $\omega_{t,k,i}^j$ is selected by minimizing a cost function term to satisfy DCBF constraints at initial condition at any time step [32].

Another challenge stemming from the linearization of the DCBF is that the constraints in (15) might become nonconvex, e.g., if $i = k = 1$, equation (15) becomes a nonconvex inequality:

$$\tilde{\psi}_0(\mathbf{x}_{t,2}^j) \geq \omega_{t,1,1}^j(1-\gamma_1)\tilde{\psi}_0(\mathbf{x}_{t,1}^j), \ \omega_{t,1,1}^j \geq 0, \quad (16)$$

where both $\omega_{t,1,1}^j$ and $\mathbf{x}_{t,1}^j$ are optimization variables. It is important to note that $\tilde{\psi}_0(\mathbf{x}_{t,0}^j)$ is always constant; therefore, we can position $\omega_{t,k,1}^j$ in front of $\tilde{\psi}_0(\mathbf{x}_{t,0}^j)$ and relocate the other optimization variables to the opposite side of the inequalities as

$$\tilde{\psi}_0(\mathbf{x}_{t,2}^j) \geq \omega_{t,1,1}^j(1-\gamma_1)^2\tilde{\psi}_0(\mathbf{x}_{t,0}^j), \ \omega_{t,1,1}^j \geq 0, \quad (17)$$

which is linear and convex. This motivates the convex formulation (18), which replaces (15) with convex constraints. The following constraints of $\tilde{\psi}_i(\mathbf{x}_{t,k+1}^j)$ are stacked by order i and, within each order, by prediction step k: first all $i{=}0$ constraints for $k = 0, \ldots, k{-}1$, then $i{=}1$ for $k = 0, \ldots, k{-}1$, and so on up to $i{=}m_{cbf}$.

$$\tilde{\psi}_0(\mathbf{x}_{t,1}^j) \geq \omega_{t,0,1}^j(1-\gamma_1)\tilde{\psi}_0(\mathbf{x}_{t,0}^j), \ \omega_{t,0,1}^j \geq 0,$$
$$\tilde{\psi}_0(\mathbf{x}_{t,2}^j) \geq \omega_{t,1,1}^j(1-\gamma_1)^2\tilde{\psi}_0(\mathbf{x}_{t,0}^j), \ \omega_{t,1,1}^j \geq 0,$$
$$\vdots$$
$$\tilde{\psi}_0(\mathbf{x}_{t,k}^j) \geq \omega_{t,k-1,1}^j(1-\gamma_1)^k\tilde{\psi}_0(\mathbf{x}_{t,0}^j), \ \omega_{t,k-1,1}^j \geq 0,$$
$$\tilde{\psi}_1(\mathbf{x}_{t,1}^j) + (\gamma_2-1)\tilde{\psi}_0(\mathbf{x}_{t,1}^j) \geq \omega_{t,0,2}^j(1-\gamma_2)(\gamma_1-1)$$
$$\tilde{\psi}_0(\mathbf{x}_{t,0}^j), \ \omega_{t,0,2}^j \leq \frac{(\gamma_1-1+\gamma_2-1)\tilde{\psi}_0(\mathbf{x}_{t,1}^j)}{(1-\gamma_2)(\gamma_1-1)\tilde{\psi}_0(\mathbf{x}_{t,0}^j)},$$
$$\tilde{\psi}_1(\mathbf{x}_{t,2}^j) + (\gamma_2-1)\tilde{\psi}_0(\mathbf{x}_{t,2}^j) \geq \omega_{t,1,2}^j(\gamma_2-1)(1-\gamma_1)^2$$
$$\tilde{\psi}_0(\mathbf{x}_{t,0}^j), \ \omega_{t,1,2}^j \leq \frac{(\gamma_1-1+\gamma_2-1)\tilde{\psi}_0(\mathbf{x}_{t,2}^j)}{(\gamma_2-1)(1-\gamma_1)^2\tilde{\psi}_0(\mathbf{x}_{t,0}^j)},$$
$$\vdots$$
$$\tilde{\psi}_{m_{cbf}-1}(\mathbf{x}_{t,k}^j) + \sum_{\nu=1}^{m_{cbf}}(\gamma_{m_{cbf}}-1)Z_{\nu,m_{cbf}}(1-\gamma_1)^{k-1}\tilde{\psi}_0(\mathbf{x}_{t,\nu}^j)$$
$$\geq \ \omega_{t,k,m_{cbf}}^j(1-\gamma_{m_{cbf}})Z_{0,m_{cbf}}(1-\gamma_1)^{k-1}\tilde{\psi}_0(\mathbf{x}_{t,0}^j), \quad (18)$$

where $\omega_{t,k,m_{cbf}}^j, Z_{\nu,m_{cbf}}, Z_{0,m_{cbf}}$ can be found in (20), (21). The above constraints can be summarized as

$$\tilde{\psi}_{i-1}(\mathbf{x}_{t,k}^j) + \sum_{\nu=1}^{i}(\gamma_i-1)Z_{\nu,i}(1-\gamma_1)^{k-1}\tilde{\psi}_0(\mathbf{x}_{t,\nu}^j) \geq \omega_{t,k,i}^j(1-\gamma_i)Z_{0,i}(1-\gamma_1)^{k-1}\tilde{\psi}_0(\mathbf{x}_{t,0}^j), \quad (19)$$

where

$$j \leq j_{\max} \in \mathbb{N}^+, \ i \in \{1, \ldots, m_{cbf}\}, \ 1 \leq k \leq N, \ \omega_{t,k,1}^j \geq 0,$$

$$\omega_{t,k,i}^j(1-\gamma_i)Z_{0,i}(1-\gamma_1)^{k-1}\tilde{\psi}_0(\mathbf{x}_{t,0}^j) \geq \sum_{\nu=0}^{i-2}Z_{\nu,i}(1-\gamma_1)^{k-1} \quad (20)$$

$$\tilde{\psi}_0(\mathbf{x}_{t,\nu+1}^j) + \sum_{\nu=1}^{i}(\gamma_i-1)Z_{\nu,i}(1-\gamma_1)^{k-1}\tilde{\psi}_0(\mathbf{x}_{t,\nu}^j) \text{ if } i \geq 2.$$

$Z_{\nu,i}$ is a constant that can be obtained recursively by reformulating $\tilde{\psi}_{i-1}(\cdot)$ back to $\tilde{\psi}_0(\cdot)$ given $\nu \in \{0, .., i\}$. We define $Z_{\nu,i}$ as follows. When $2 \leq i, \nu \leq i-2$, we have

$$Z_{\nu,i} = \sum_{l=1}^{l_{\max}}[(\gamma_{\zeta_1}-1)(\gamma_{\zeta_2}-1)\cdots(\gamma_{\zeta_{i-\nu-1}}-1)]_l, \quad (21)$$
$$\zeta_1 < \zeta_2 < \cdots < \zeta_{i-\nu-1}, \zeta_s \in \{1, 2, \ldots, i-1\},$$

where $l_{\max} = \binom{i-1}{i-\nu-1}$, ζ_s denotes $\zeta_1, \zeta_2, \ldots, \zeta_{i-\nu-1}$. $[\cdot]_l$ denotes the lth combination of the product of the elements in parenthesis, e.g., if $i = 4, \nu = 1, Z_{1,4} = (\gamma_1-1)(\gamma_2-1) + (\gamma_1-1)(\gamma_3-1) + (\gamma_2-1)(\gamma_3-1)$. For the case $\nu = i-1$, we define $Z_{\nu,i} = 1$. Besides that, we define $Z_{\nu,i} = 0$ for the case $\nu = i$ ((17) shows the case where $Z_{0,1} = 1, Z_{1,1} = 0$). With the linearized DHOCBF in place, we now connect the construction to closed-loop safety. Assuming that at each step there exists a control input satisfying (19) under (20), the theorem below establishes forward invariance of the (linearized) safe set.

Theorem 2. *Given a linearized DHOCBF $h_\parallel(\mathbf{x})$ for system (2) with corresponding functions $\tilde{\psi}_0(\mathbf{x}), \tilde{\psi}_1(\mathbf{x}) \ldots, \tilde{\psi}_{m_{cbf}}(\mathbf{x})$ defined by (14), if $\tilde{\psi}_0(\mathbf{x}_0) \geq 0$, then any Lipschitz controller $\mathbf{u}_t$ that satisfies the constraints in (19) given conditions in (20), $\forall t \geq 0$ renders $\tilde{\mathcal{C}}_0 := \{\mathbf{x} \in \mathbb{R}^n : \tilde{\psi}_0(\mathbf{x}) \geq 0\}$ forward invariant for system (2), i.e., $\mathbf{x}_t \in \tilde{\mathcal{C}}_0, \forall t \geq 0$.*

Proof:
By rewriting (14), if $i \geq 2, k \geq 1$, we obtain

$$\tilde{\psi}_i(\mathbf{x}_{t,k}^j) = \sum_{\nu=0}^{i+1}Z_{\nu,i+1}(1-\gamma_1)^{k-1}\tilde{\psi}_0(\mathbf{x}_{t,\nu}^j). \quad (22)$$

Then, we have

$$\tilde{\psi}_i(\mathbf{x}_{t,k}^j) = \tilde{\psi}_{i-1}(\mathbf{x}_{t,k+1}^j) + (\gamma_i-1)\tilde{\psi}_{i-1}(\mathbf{x}_{t,k}^j) \geq 0$$
$$\implies \sum_{\nu=0}^{i}Z_{\nu,i}(1-\gamma_1)^{k-1}\tilde{\psi}_0(\mathbf{x}_{t,\nu+1}^j) +$$
$$\sum_{\nu=0}^{i}(\gamma_i-1)Z_{\nu,i}(1-\gamma_1)^{k-1}\tilde{\psi}_0(\mathbf{x}_{t,\nu}^j) \geq 0$$
$$\implies Z_{i-1,i}(1-\gamma_1)^{k-1}\tilde{\psi}_0(\mathbf{x}_{t,i}^j) + \sum_{\nu=0}^{i-2}Z_{\nu,i}(1-\gamma_1)^{k-1}$$
$$\tilde{\psi}_0(\mathbf{x}_{t,\nu+1}^j) + \sum_{\nu=1}^{i}(\gamma_i-1)Z_{\nu,i}(1-\gamma_1)^{k-1}\tilde{\psi}_0(\mathbf{x}_{t,\nu}^j) \geq$$
$$(1-\gamma_i)Z_{0,i}(1-\gamma_1)^{k-1}\tilde{\psi}_0(\mathbf{x}_{t,0}^j). \quad (23)$$

This article has been accepted for publication in IEEE Open Journal of Control Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/OJCSYS.2025.3612245

LIU *ET AL.*: LEARNING-ENABLED ITERATIVE CONVEX OPTIMIZATION FOR SAFETY-CRITICAL MODEL PREDICTIVE CONTROL

Since $\tilde{\psi}_0(\mathbf{x}_{t,0}^j)$ is a constant, we position the slack variable $\omega_{t,k,i}^j$ in front of it and equation (23) becomes

$$Z_{i-1,i}(1-\gamma_1)^{k-1}\tilde{\psi}_0(\mathbf{x}_{t,i}^j) \geq \omega_{t,k,i}^j(1-\gamma_i)Z_{0,i}(1-\gamma_1)^{k-1}$$
$$\tilde{\psi}_0(\mathbf{x}_{t,0}^j) - \sum_{\nu=0}^{i-2} Z_{\nu,i}(1-\gamma_1)^{k-1}\tilde{\psi}_0(\mathbf{x}_{t,\nu+1}^j)-$$
$$\sum_{\nu=1}^{i}(\gamma_i-1)Z_{\nu,i}(1-\gamma_1)^{k-1}\tilde{\psi}_0(\mathbf{x}_{t,\nu}^j).$$
(24)

Note that $\tilde{\psi}_0(\mathbf{x}_{t,i+k-1}^j) \geq (1-\gamma_1)^{k-1}\tilde{\psi}_0(\mathbf{x}_{t,i}^j)$. If $i=1$, we need to ensure

$$\tilde{\psi}_0(\mathbf{x}_{t,k}^j) \geq \omega_{t,k-1,1}^j(1-\gamma_1)^k\tilde{\psi}_0(\mathbf{x}_{t,0}^j). \qquad (25)$$

Equations (24), (25) are also equivalent to (19). Since $Z_{i-1,i}=1$, in order to make $\tilde{\psi}_0(\mathbf{x}_{t,i}^j) \geq 0$, we need

$$\omega_{t,k,i}^j(1-\gamma_i)Z_{0,i}(1-\gamma_1)^{k-1}\tilde{\psi}_0(\mathbf{x}_{t,0}^j) \geq \sum_{\nu=0}^{i-2}Z_{\nu,i}(1-\gamma_1)^{k-1}$$
$$\tilde{\psi}_0(\mathbf{x}_{t,\nu+1}^j)+\sum_{\nu=1}^{i}(\gamma_i-1)Z_{\nu,i}(1-\gamma_1)^{k-1}\tilde{\psi}_0(\mathbf{x}_{t,\nu}^j) \text{ if } i \geq 2,$$
$$\omega_{t,k-1,1}^j \geq 0 \text{ if } i=1,$$
(26)

which is the same as (20). This means that by satisfying constraints in (19), given the conditions in (20), $\forall t \geq 0$, we have $\tilde{\psi}_0(\mathbf{x}_{t,i}^j) \geq 0, j \leq j_{\max} \in \mathbb{N}^+, \; i \in \{1,\ldots,m_{\mathrm{cbf}}\}, \forall t \geq 0$, thus $\tilde{\psi}_0(\mathbf{x}_t) \geq 0$ is guaranteed. ∎

Remark 3. *Note that if we position $\omega_{t,k,i}^j$ in front of $\sum_{\nu=1}^{i}(\gamma_i-1)Z_{\nu,i}(1-\gamma_1)^{k-1}\tilde{\psi}_0(\mathbf{x}_{t,\nu}^j)$ in (19), equation (15) is the same as (19). This illustrates that the decay rate in (19) used by the iMPC-DHOCBF is partially relaxed compared to the one in (15) due to the requirement of the linearization. This can affect the feasibility of the optimization. Allowing $\omega_{t,k,i}^j \in \mathbb{R}$ (unbounded) can further enlarge feasibility, but it does not enforce $\tilde{\psi}_0(\mathbf{x}) \geq 0$ and thus may violate safety. By constraining $\omega_{t,k,i}^j$ via (20), we guarantee $\tilde{\psi}_0(\mathbf{x}) \geq 0$ (safety) while still enlarging feasibility relative to the no-slack case; hence we adopt (20). Note, however, that a bounded slack only improves **per-step** feasibility and does not ensure **recursive feasibility**: without a control-invariant terminal set/tube, feasibility at time t does not imply feasibility at $t+1$, since under bounded inputs and a finite horizon the next-step linearized constraints may conflict with the dynamics/bounds or demand a slack beyond its admissible range.*

*2) Safety Boundary Detector for **Unknown** Unsafe Boundaries:* Notice that $\tilde{\mathbf{x}}_{t,k}^j$ in Sec. IV-C-1 could be implicit if $h(\cdot)$ is unknown or discontinuous, or for unsafe sets of irregular shapes (like general ellipse), but it could still be numerically approximated as the values of $\bar{\mathbf{x}}_{t,k}^j$ known at iteration j before the linearization, i.e., if we could predict

the nearest point $\tilde{\mathbf{x}}_{t,k}^j$ for the ith unsafe set. For a map, the linearized boundary of the unsafe set can be expressed as

$$h_{\|}(\mathbf{x}_{t,k}^j, \tilde{\mathbf{x}}_{t,k}^j) = (\tilde{\mathbf{x}}_{t,k}^j - \bar{\mathbf{x}}_{t,k}^j)^T(\mathbf{x}_{t,k}^j - \tilde{\mathbf{x}}_{t,k}^j), \qquad (27)$$

which can also be used as a linearized DHOCBF in Sec. IV-C-1. Specifically, it involves finding the nearest point on the boundary of an unsafe set or an inaccessible region relative to the current position, and then constructing a tangent line or plane through that point to serve as the linearized unsafe boundary. Next we will show how to design a Deep Neural Network (DNN) to get the approximate nearest point $\tilde{\mathbf{x}}_{t,k}^j$.

The designed DNN learns a specific map to identify and return the nearest point on each unsafe set relative to the system. A line perpendicular to the vector connecting the robot's current location and this point defines the linearized boundary. During training, nearest points are collected as ground truth for every safe point, enabling the DNN to learn via supervised learning. To efficiently store this information, we focus on architectures with strong associative memory, as conventional DNNs excel at pattern recognition but lack memory capacity. Models such as Hopfield Networks, Neural Turing Machines, Autoencoders [41], and Transformers are well-suited for this task.

Model Selection: We adopt an Autoencoder framework for its ability to encode complex patterns, but use only the encoder to streamline the model and accelerate inference. Instead of reconstructing inputs, our focus is on efficient nearest-point retrieval. During training, the encoder compresses the map into a low-dimensional space by minimizing the loss between predicted and ground-truth nearest points. Once trained, it acts as a memory register, recalling nearest points efficiently. Omitting the decoder reduces computational overhead, and the encoder delivers consistent inference times regardless of map complexity (e.g., the number of unsafe sets).

Map Processing: We begin by identifying the boundaries of unsafe sets on a pixel-based canvas that simulates convex and non-convex shapes. Using image processing, we detect boundary pixels without requiring precise mathematical descriptions. To improve efficiency, we select a subset of boundary pixels $\partial\mathcal{X}_{\mathrm{unsafe}}^i$ to form discrete contours representing i^{th} unsafe set ($i \leq p$). This method generalizes well across different maps and shapes. The processed boundaries are shown in Fig. 3, where $\bar{\mathbf{x}} = [\bar{x}, \bar{y}]^T$ denotes the system's current position, and $[x_1, y_1, \ldots, x_p, y_p]^T$ represent the nearest boundary points $\mathbf{x}_1, \ldots, \mathbf{x}_p$, computed via pixel-wise comparison:

$$\mathbf{x}_i = \Pi_{\partial\mathcal{X}_{\mathrm{unsafe}}^i}(\bar{\mathbf{x}}) := \arg\min_{z_i \in \partial\mathcal{X}_{\mathrm{unsafe}}^i} \|\bar{\mathbf{x}} - z_i\|, \qquad (28)$$

where z_i is a point on the boundary of the ith unsafe set, and $\mathbf{x}_i = \Pi_{\partial\mathcal{X}_{\mathrm{unsafe}}^i}(\bar{\mathbf{x}})$ denotes the nearest boundary point to $\bar{\mathbf{x}}$. Both $\bar{\mathbf{x}}$ and $\mathbf{x}_i$ are shown as red dots.

Network Design and Training: As described above, we use the encoder of an Autoencoder neural network. Figure 3 shows its structure, where the number of nodes

decreases layer by layer to extract essential map information. Each layer is fully connected to ensure comprehensive processing. The network takes the system's current location $\bar{\mathbf{x}} = [\bar{x}, \bar{y}]^T$ as input and outputs predicted nearest points $[\tilde{x}_1, \tilde{y}_1, \ldots, \tilde{x}_p, \tilde{y}_p]^T$ on nearby unsafe sets. Since the number of unsafe sets varies across maps, the output size may change, requiring network re-training for adaptation. The first hidden layer is denoted as h_0. The subsequent layers' dimensions are defined by:

$$h_i = h_0 - i * floor((h_0 - l_{out})/L), \qquad (29)$$

in which h_i stands for the number of nodes in the ith of hidden layers, l_{out} is the output layer size, and L denotes the total number of hidden layers. The floor function rounds down a real number to the nearest integer less than or equal to that number. This model architecture, which uniformly reduces layer by layer, can adaptively adjust to the size of the output and ensure smoothness in dimension reduction.

We train a neural network $f_\theta(\bar{\mathbf{x}})$ with parameters θ to approximate the nearest points by minimizing the mean squared error loss:

$$\mathcal{L}(\theta) = \min_\theta \frac{1}{p} \sum_{i=1}^{p} \|f_\theta(\bar{\mathbf{x}}^{\text{train}}) - \mathbf{x}_i^{\text{train}}\|^2, \qquad (30)$$

where the system locations $\bar{\mathbf{x}}^{\text{train}}$ are sampled across the map and their nearest boundary points $\mathbf{x}_1^{\text{train}}, \ldots, \mathbf{x}_p^{\text{train}}$ are identified by (28) as ground truth. If multiple nearest points exist (e.g., with nonconvex boundaries), one is selected arbitrarily. More generally, the learning objective can be viewed as minimizing the expected loss over the data distribution $\mathcal{D}$:

$$\mathcal{L}(\theta) = \min_\theta \mathbb{E}_{\bar{\mathbf{x}}^{\text{train}} \sim \mathcal{D}} \left[\|f_\theta(\bar{\mathbf{x}}^{\text{train}}) - \Pi_{\partial \mathcal{X}_{\text{unsafe}}^i}(\bar{\mathbf{x}}^{\text{train}})\|^2 \right]. \qquad (31)$$

To quantify generalization, we evaluate the performance of the trained model on unseen test data using:

$$\text{MSE}_{\text{test}} = \frac{1}{p} \sum_{j=1}^{p} \|f_\theta(\bar{\mathbf{x}}^{\text{test}}) - \mathbf{x}_j^{\text{test}}\|^2. \qquad (32)$$

Note that training samples $\bar{\mathbf{x}}^{\text{train}}$ include system locations both inside and outside the unsafe sets, which improves the network's generalization. In contrast, testing samples $\bar{\mathbf{x}}^{\text{test}}$ contain only locations outside the unsafe sets, to evaluate prediction accuracy. Since the network's output $\tilde{\mathbf{x}}_i = [\tilde{x}_i, \tilde{y}_i]^T = f_\theta(\bar{\mathbf{x}}), i \in \{1, \ldots, p\}$ could be used as the approximate nearest points, we can incorporate the above process into the SBD to define linearized safe sets from (27) by $h_\|(\mathbf{x}_{t,k}^j, \tilde{\mathbf{x}}_{t,k}^j) \geq 0, \forall t \in \mathbb{N}$ similar to Fig. 2. The linearization of the DHOCBF $h_\|(\mathbf{x}_{t,k}^j, \tilde{\mathbf{x}}_{t,k}^j)$ up to the highest order will be the same as shown in Sec. IV-C-1.

Remark 4 (*Continuity of $h_\|(\cdot)$*). *Whether the equation of a linearized safe boundary* (27) *to a continuous boundary is continuous with respect to point $\mathbf{x}_{t,k}^j$ depends on the properties and smoothness of the curve. For smooth boundaries, such as circles or ellipses, the nearest point $\tilde{\mathbf{x}}_{t,k}^j$ to*
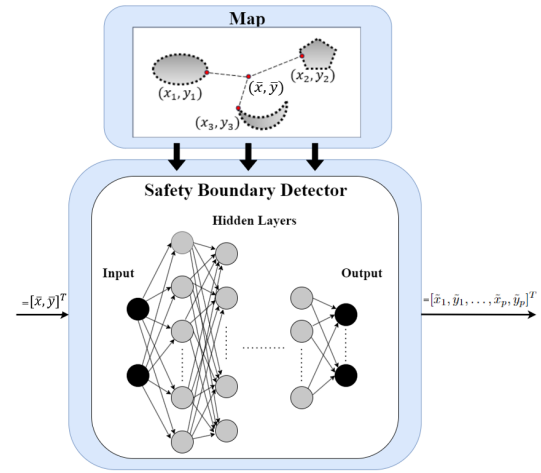

FIGURE 3: **System identification for the Safety Boundary Detector (SBD), a DNN $f_\theta(\bar{\mathbf{x}})$ trained on map data. Given the robot's location $(\bar{x}, \bar{y})$, the SBD outputs the nearest point $(\tilde{x}_i, \tilde{y}_i)$ on each of the p unsafe set boundaries, one per set.**

an external point $\mathbf{x}_{t,k}^j$ can typically be calculated, and this point changes continuously as $\mathbf{x}_{t,k}^j$ moves. In these cases, (27) *is usually continuous as well. However, for shapes with sharp corners or discontinuities, like stars, the location of the nearest point may jump abruptly with minor movements of $\mathbf{x}_{t,k}^j$ (jumping from one vertex to another or from an edge to a vertex), leading to discontinuities in* (27) *at these points. The pixel-based boundary approximation introduced in Sec. IV-C-2 will also bring discontinuity. Discontinuity affects the optimality of the obtained controller. However, pixel-based discontinuity could be ameliorated by increasing the number of pixels (minimizing the distance between adjacent pixels).*

Remark 5 (*Motivation for Using Neural Networks as Nearest-Point Detectors*). *While traditional methods such as Euclidean distance functions (EDF [42]) and spatial tree structures (e.g., k-d trees [43]) are widely used for nearest-boundary queries, our motivation for adopting a DNN-based approach is three-fold. First, it is memory-efficient, avoiding the need to store dense distance maps or indexing structures, whose memory usage scales with map size and resolution. In contrast, the trained DNN serves as a compact function approximator with fixed memory. Second, when boundary pixels are sparse, EDF and spatial-tree methods rely on limited samples, potentially leading to suboptimal nearest-point queries. Our DNN can interpolate from sparse data to predict locally optimal points, mitigating discretization issues. Third, under noisy observations, traditional methods may fail due to reliance on disturbed pixels, while the DNN can learn to filter outliers and predict smooth, reliable points by leveraging spatial patterns in training. Its accuracy can further improve through model and hyperparameter tuning.*

Remark 6 (*The Convexity of Unsafe Sets*). *Let $O \subset \mathbb{R}^n$ be the unsafe set and $C = \mathbb{R}^n \setminus O$ the safe set. At the*

*current location $\bar{\mathbf{x}}$, the SBD returns the nearest boundary point $\tilde{\mathbf{x}} \in \partial O$. We form the local linear constraint $H(\bar{\mathbf{x}}) := \{\mathbf{x} : h_\parallel \geq 0\}$ where $h_\parallel$ is defined by (27). If O is convex, the hyperplane at $\tilde{x}$ is supporting and $H(\bar{\mathbf{x}}) \subset C$, so enforcing $h_\parallel \geq 0$ implies the original safety constraint along the horizon (as shown in Fig. 2). If O is nonconvex, $H(\bar{\mathbf{x}})$ is a **local** approximation of C near the boundary component containing $\tilde{\mathbf{x}}$. In either case, the closed-loop safety can be strengthened by re-linearizing at every iMPC iteration, using a small discretization step Δt to limit inter-step drift, and (optionally) intersecting a few tangents, $\bigcap_{m=1}^{l} H^{(m)}(\bar{\mathbf{x}})$, i.e., having the SBD return the l nearest boundary points $\tilde{\mathbf{x}}_{t,k}^{j,(m)}$ to $\bar{\mathbf{x}}$ for each obstacle and forming multiple tangents to obtain a tighter local approximation of C, or adding a margin $h_\parallel \geq \varepsilon > 0$. Because the SBD is agnostic to convexity and supplies nearest boundary points for arbitrary shapes, the procedure applies to complex unsafe sets.*

Remark 7 (*DNN Prediction Errors and Certified Bounds*). *Note that our DNN-based Safety Boundary Detector does not provide a strict a priori bound on the prediction error. Since our setting involves low-dimensional inputs (e.g., 2D location states), existing techniques such as Lipschitz-based certificates [56], neural network verification tools [57], or robust training with certified bounds [58] could, in principle, be applied to derive worst-case guarantees. Incorporating such certified error bounds into our framework will be pursued as future work, which can further strengthen the theoretical safety guarantees. At the same time, extending these verification techniques to high-dimensional sensory inputs (e.g., images or point clouds) and real-time MPC remains an open challenge for the broader community.*

D. CFTOC PROBLEM

In Secs. IV-B and IV-C, we have demonstrated the linearization of system dynamics and the safety constraints using DHOCBF. This enables us to incorporate them as constraints in a convex MPC formulation at each iteration, which we refer to as Convex Finite-Time Constrained Optimization Control (CFTOC). This is solved at iteration j with optimization variables $\mathbf{U}_t^j = [\mathbf{u}_{t,0}^j, \ldots, \mathbf{u}_{t,N-1}^j]$ and $\Omega_{t,i}^j = [\omega_{t,0,i}^j, \ldots, \omega_{t,N,i}^j]$, where $i \in \{1, \ldots, m_{\text{cbf}}\}$.

CFTOC of iMPC-DHOCBF at iteration j:

$$\min_{\mathbf{U}_t^j, \Omega_{t,1}^j, \ldots, \Omega_{t,m_{\text{cbf}}}^j} p(\mathbf{x}_{t,N}^j) + \sum_{k=0}^{N-1} q(\mathbf{x}_{t,k}^j, \mathbf{u}_{t,k}^j, \omega_{t,k,i}^j) \quad (33a)$$

$$\text{s.t. } \mathbf{x}_{t,k+1}^j - \bar{\mathbf{x}}_{t,k+1}^j = A_{t,k}^j (\mathbf{x}_{t,k}^j - \bar{\mathbf{x}}_{t,k}^j) + B_{t,k}^j (\mathbf{u}_{t,k}^j - \bar{\mathbf{u}}_{t,k}^j) + c_{t,k}^j, \quad (33b)$$

$$\mathbf{u}_{t,k}^j \in \mathcal{U}, \ \mathbf{x}_{t,k}^j \in \mathcal{X}, \ \omega_{t,k,i}^j \text{ s.t. } (20), \quad (33c)$$

$$\tilde{\psi}_{i-1}(\mathbf{x}_{t,k}^j) + \sum_{\nu=1}^{i} (\gamma_i - 1) Z_{\nu,i}(1 - \gamma_1)^{k-1} \tilde{\psi}_0(\mathbf{x}_{t,\nu}^j) \geq$$

$$\omega_{t,k,i}^j (1 - \gamma_i) Z_{0,i}(1 - \gamma_1)^{k-1} \tilde{\psi}_0(\mathbf{x}_{t,0}^j). \quad (33d)$$

In the CFTOC, the linearized dynamics constraints in (12)

and the linearized DHOCBF constraints in (19) are enforced with constraints (33b) and (33d) at each open-loop time step $k \in \{0, \ldots, N-1\}$. The state and input constraints are considered in (33c). The slack variables are subject to constraints in (20) to enhance feasibility while guaranteeing safety, as discussed in Thm. 2. Note that, for ensuring the safety guarantee established by the DHOCBF, the constraints (33d) are enforced with $i \in \{0, \ldots, m_{\text{cbf}}\}$, where $Z_{\nu,i} \in \mathbb{R}$ is as defined in (21) with $\nu \in \{0, .., i\}$. The optimal decision variables of (33) at iteration j is a list of control input vectors as $\mathbf{U}_t^{*,j} = [\mathbf{u}_{t,0}^{*,j}, \ldots, \mathbf{u}_{t,N-1}^{*,j}]$ and a list of slack variable vectors as $\Omega_{t,i}^{*,j} = [\omega_{t,0,i}^{*,j}, \ldots, \omega_{t,N-1,i}^{*,j}]$. For avoidance of multiple unsafe sets, the DHOCBF constraint (33d) in the CFTOC framework (33) originally considers only a single set. To handle p unsafe sets, the slack variable vectors $\Omega_{t,i}^j$ (used in both the cost (33a) and (33d)) and the constraint itself must be expanded accordingly. The CFTOC is solved iteratively in the proposed iMPC-DHOCBF framework, and a solution is extracted once the convergence criteria or maximum iteration number $j_{\max}$ is reached, as shown in Fig. 1.

E. COMPLEXITY OF iMPC-DHOCBF

1) Size of Map and Network: In Sec. IV-C-2, before we train the network, we need to gather as many sampled system's locations as possible across the map, while also identifying the nearest points on the boundaries of unsafe sets relative to the system. This means that the size of the map will influence the number of sampled locations and ground truth data. The larger the map, the greater the number of sampled locations and ground truth needed, which in turn extends the time required to train the network. Additionally, the more complex the network structure (such as more hidden layers or more nodes), the longer it will take to train the network.

2) Number of Unsafe Sets: In (33), every system's state $\mathbf{x}_{t,k}^j$ corresponds to a nearest point $\tilde{\mathbf{x}}_{t,k}^j$ on each unsafe set, and each nearest point corresponds to a linearized DHOCBF (33d). Therefore, the more unsafe sets there are, the more DHOCBF constraints exist, leading to increased complexity in the CFTOC. One can choose to only consider unsafe sets within a certain range based on the current location to reduce complexity. This means that when the horizon N is large, the predictive capability of the MPC will be limited. Nonetheless, the complexity related to unsafe sets in our algorithm is lower than the state-of-the-art [40], because each polytopic unsafe set corresponds to multiple DHOCBFs.

3) Horizon Length: In (33), the horizon N results in a linear increase in the number of all constraints. Compared to distance constraints, DHOCBF constraints incorporate additional higher-order sufficient constraints to ensure safety. As discussed in [40], DHOCBF constraints enable effective avoidance behavior for unsafe sets with a smaller horizon length. One could consider to require the horizon of DHOCBF N_{cbf} smaller than N to reduce the complexity.

(a) iMPC-DHOCBF when $N = 24$, $\gamma_1 = \gamma_2 = 0.4$. **(b)** iMPC-DHOCBF with $m_{\text{cbf}} = 2$. **(c)** NMPC-DHOCBF with $m_{\text{cbf}} = 2$. **(d)** iMPC-DHOCBF and NMPC-DHOCBF with $m_{\text{cbf}} = 1$.
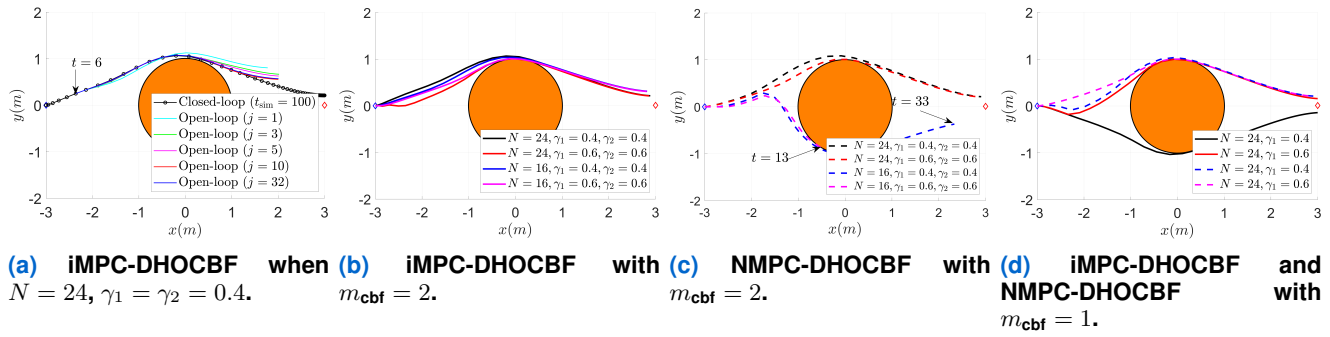
FIGURE 4: Open-loop and closed-loop trajectories with controllers iMPC-DHOCBF (solid lines) and NMPC-DHOCBF (dashed lines): (a) several open-loop trajectories at different iterations predicted at $t = 6$ and one closed-loop trajectory with controller iMPC-DHOCBF; (b) closed-loop trajectories with controller iMPC-DHOCBF with different choices of N and γ; (c) closed-loop trajectories with controller NMPC-DHOCBF with different choices of N and γ. Note that two trajectories stop at $t = 13$ and $t = 33$ because of infeasibility; (d) closed-loop trajectories with controllers iMPC-DHOCBF and NMPC-DHOCBF with $m_{\text{cbf}} = 1$. Both methods work well for safety-critical navigation. This figure demonstrates that at a specific time step, iMPC-DHOCBF can iteratively drive the open-loop trajectory to converge to a local minimum while ensuring the safety of the closed-loop trajectory over t_{sim}.

(a) Location x **(b) Location y** **(c) Orientation θ** **(d) Speed v**
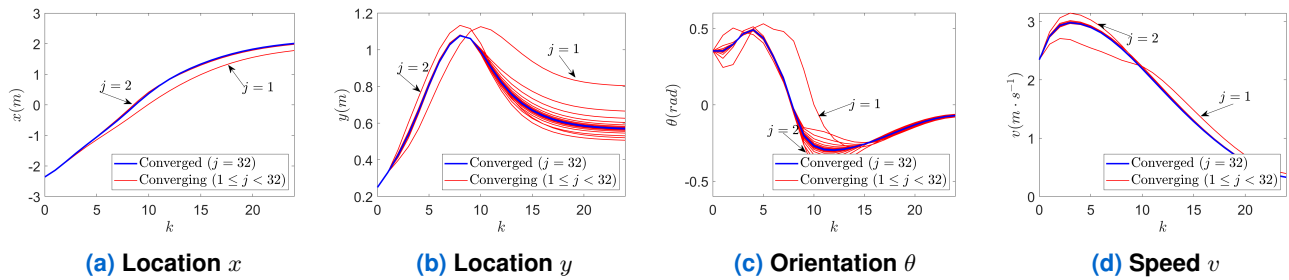
FIGURE 5: Iterative convergence of all states at converged iteration $j_{6,\text{conv}} = 32$ with $N = 24, m_{\text{cbf}} = 2, \gamma_1 = \gamma_2 = 0.4$. iMPC does help to optimize the cost function to reach local optimal minimum.

4) Convergence Criterion: In Fig. 1, the number of iterations at each time step is determined by the convergence criterion or $j_{\max}$. One can flexibly choose which variables to include in the convergence criterion, e.g., state $\mathbf{X}_t^{*,j}$, input $\mathbf{U}_t^{*,j}$, or both state and input. Appropriately relaxing the convergence conditions (such as increasing the allowable error) or reducing $j_{\max}$ can both decrease the complexity of the algorithm.

5) Highest Order of DHOCBF: In (33), the highest order of DHOCBF m_{cbf} results in a linear increase in the number of DHOCBF constraints (33d). As discussed in Rem. 1, it is not necessary to formulate DCBF constraints up to the mth order. In other words, the highest order for DHOCBF could be m_{cbf} with $m_{\text{cbf}} \leq m$. Making m_{cbf} smaller than m but larger than one will reduce the complexity of the algorithm without significantly compromising safety.

6) Complexity Scaling with State and Input Dimensions: Because each prediction step carries its own copy of the state vector $x_{t,k} \in \mathbb{R}^n$ and input vector $u_{t,k} \in \mathbb{R}^q$, enlarging n or q inflates the stacked decision vector that the CFTOC must solve. Every extra state or input triggers additional rows for the linearized dynamics, box constraints, and any DHOCBF or slack-variable terms. Since the number of

constraints already scales linearly with the horizon N, the overall problem size grows roughly $\mathcal{O}(N(n+q))$ in memory. More states also mean larger Jacobians $A_{t,k}^j, B_{t,k}^j$ and extra slack variables—each adding per-iteration linearization and matrix-assembly cost.

V. CASE STUDIES

We focus on a unicycle robot. Case Study I compares our method with the baseline NMPC-DHOCBF using a circular obstacle as the unsafe set. The baseline is extended with a relaxed DHOCBF based on (11), following [32, Rem. 4]. Case studies II and III demonstrate the proposed iMPC-DHOCBF on more complex maps. Case study III also includes results for a vehicle model. Animation videos are available at https://youtu.be/G9S7y90LRig.

A. CASE STUDY I: POINT ROBOT AVOIDANCE OF A CIRCULAR OBSTACLE WITH KNOWN BOUNDARY

Since the nearest point on a circle from a point outside the circle can be expressed by an accurate geometric equation, SBD is equivalent to finding tangent lines on this equation using derivatives, so we do not need to train a DNN here.

This article has been accepted for publication in IEEE Open Journal of Control Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/OJCSYS.2025.3612245

LIU *ET AL.*: LEARNING-ENABLED ITERATIVE CONVEX OPTIMIZATION FOR SAFETY-CRITICAL MODEL PREDICTIVE CONTROL

1) System Dynamics: Consider a discrete-time unicycle model in the form

$$\begin{bmatrix} x_{t+1}-x_t \\ y_{t+1}-y_t \\ \theta_{t+1}-\theta_t \\ v_{t+1}-v_t \end{bmatrix} = \begin{bmatrix} v_t \cos(\theta_t)\Delta t \\ v_t \sin(\theta_t)\Delta t \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \begin{bmatrix} u_{1,t} \\ u_{2,t} \end{bmatrix}, \quad (34)$$

where $\mathbf{x}_t = [x_t, y_t, \theta_t, v_t]^T$ captures the 2-D location, heading angle, and linear speed; $\mathbf{u}_t = [u_{1,t}, u_{2,t}]^T$ represents angular velocity (u_1) and linear acceleration (u_2), respectively. The system is discretized with $\Delta t = 0.1$ and the total number of discretization steps t_{sim} equals T in Problem. 1. System (34) is subject to the state and input constraints $\mathcal{X}$ and $\mathcal{U}$ which can be found in [38].

2) System Configuration: The initial state is $[-3, 0, 0, 0]^T$ and the target state is $\mathbf{x}_r = [3, 0.01, 0, 0]^T$, which are marked as blue and red diamonds in Fig. 4. The circular obstacle is centered at $(0,0)$ with $r = 1$, which is displayed in orange. The other reference vectors are $\mathbf{u}_r = [0,0]^T$ and $\omega_r = [1,1]^T$. We use the offset $y = 0.01m$ in $\mathbf{x}_r$ to prevent singularity of the optimization problem.

3) DHOCBF: As a candidate DHOCBF function $\psi_0(\mathbf{x}_t)$, we choose a quadratic distance function for circular obstacle avoidance $h(\mathbf{x}_t) = (x_t-x_0)^2+(y_t-y_0)^2-r^2$, where (x_0, y_0) and r denote the obstacle center location and radius, respectively. The linearized DHOCBF $\tilde{\psi}_0(\mathbf{x}_{t,k}^j)$ in (14) follows the formulation presented in [38].

4) MPC Design: The CFTOC cost penalizes deviations from reference vectors and includes slack variables to enhance feasibility while ensuring safety. The details of the MPC design can be found in [38].

5) Convergence Criteria: The iterative optimization terminates when absolute or relative convergence criteria are met, as defined in Fig. 1. See [38] for a similar formulation. The maximum iteration number is set as $j_{\max} = 1000$. To ensure a fair comparison with NMPC-DHOCBF, the hyperparameters P, Q, R, S are kept consistent across all configurations.

6) Solver and CPU Specs: For iMPC-DHOCBF, we used OSQP [44] to solve the convex optimizations at all iterations. The baseline approach NMPC-DHOCBF is open-source, and was solved using IPOPT [45] with the modeling language Yalmip [46]. We used a Windows desktop with Intel Core i7-8700 (CPU 3.2 GHz) running Matlab.

7) Iterative Convergence: The iterative convergence is shown in Figs. 4a and 5. Figure 4a shows the closed-loop trajectory (the black line) generated by solving the iMPC-DHOCBF until the converged iteration $j_{t,\text{conv}}$ from $t = 0$ to $t = t_{\text{sim}} = 100$ and open-loop iteratively converging trajectories (colored lines) at different iterations at $t = 6$. Figure 5 presents more details on the iterative convergence of states at different iterations at $t = 6$ with number of iterations $j_{t,\text{conv}} = 32$. We note that, after around 10 iterations, the converging lines for the states (red lines)

nearly overlap with the converged line (blue line) in Fig. 5. This verifies the relations of the converging trajectory (red line) and the converged trajectory (blue line) in Fig. 4a. The convergence behavior of iMPC-DHOCBF varies over time and with different γ values, typically requiring fewer than 100 iterations except near obstacles (see [38] for details).

8) Convergence with Different Hyperparameters: Figs. 4b, 4c, and 4d show the closed-loop trajectories generated by solving iMPC-DHOCBF (solid lines) and NMPC-DHOCBF (dashed lines) at converged iteration $j_{t,\text{conv}}$ from $t = 0$ to $t = t_{\text{sim}} = 45$ with different hyperparameters. Both controllers show good performance on obstacle avoidance. Based on black, red, blue and magenta lines with the highest order of CBF constraint $m_{\text{cbf}} = 2$ in Figs. 4b and 4c, as γ_1, γ_2 become smaller, the system tends to turn further away from the obstacle when it is getting closer to obstacle, which indicates a safer control strategy. From the lines in Fig. 4d where $m_{\text{cbf}} = 1$, we can see that the system can still safely navigate around the obstacle, although it turns away from the obstacle later than when having one more CBF constraint in Figs. 4b and 4c, indicating that having CBF constraints up to the relative degree enhances safety. The blue and magenta dashed lines in Fig. 4c stop at $t = 33$ and $t = 13$, respectively, due to infeasibility with $N = 16$. Although the model has passed the obstacle, NMPC-DHOCBF may still become infeasible under certain hyperparameter settings due to overly conservative constraints. This illustrates its sensitivity to horizon selection. In contrast, iMPC-DHOCBF generates complete closed-loop trajectories with both $N = 16$ and $N = 24$, showing improved robustness to horizon choices, as seen in Fig. 4b.

9) Computation Time: To compare computational times between iMPC-DHOCBF and the baseline NMPC-DHOCBF, we generate 1000 randomized safe states within the constraint set $\mathcal{X}$. Both methods use the same N and m_{cbf}, and are evaluated on computation time and feasibility at each sampled state. Table 2 shows the distributions of computation times and infeasibility rates for one-step trajectory generation. For NMPC-DHOCBF, the mean and standard deviation of computation times grow significantly with increasing N and m_{cbf}, whereas iMPC-DHOCBF remains largely unaffected. Overall, iMPC-DHOCBF achieves much faster computation—up to $100 \sim 300\times$ speedup over the baseline—depending on the chosen hyperparameters.

10) Optimization Feasibility: For iMPC-DHOCBF, the infeasibility rate increases with a longer horizon N or smaller m_{cbf}, while NMPC-DHOCBF shows less sensitivity to these parameters. As N increases, iMPC-DHOCBF achieves lower infeasibility rates than NMPC-DHOCBF. This difference arises from variations in convergence criteria, warm starts, and relaxation techniques, as discussed in Rems. 2 and 3. NMPC-DHOCBF, solved via IPOPT, applies stricter convergence criteria but benefits from a more refined warm start and relaxed nonlinear DCBF constraints (15). In contrast, iMPC-

TABLE 2: Statistical benchmark for computation time and feasibility between NMPC-DHOCBF and iMPC-DHOCBF with randomized states. The target position is shared among four approaches and the hyperparameters are fixed as $\gamma_1 = \gamma_2 = 0.4$ for all random scenarios.

Approaches		$N = 4$	$N = 8$	$N = 12$	$N = 16$	$N = 20$	$N = 24$
NMPC-DHOCBF	mean / std (s)	3.687 ± 6.360	23.882 ± 17.988	27.329 ± 20.115	28.953 ± 22.058	30.970 ± 23.564	29.929 ± 22.105
($m_{\mathrm{cbf}} = 2$)	infeas. rate	5.8%	27.5%	21.1%	16.4%	14.5%	14.4%
NMPC-DHOCBF	mean / std (s)	2.933 ± 4.678	19.077 ± 14.024	20.418 ± 15.401	22.749 ± 17.039	24.053 ± 17.811	25.365 ± 18.211
($m_{\mathrm{cbf}} = 1$)	infeas. rate	6.3%	13.9%	13.0%	14.6%	13.8%	15.4%
iMPC-DHOCBF	mean / std (s)	0.135 ± 0.294	0.104 ± 0.242	0.102 ± 0.217	0.131 ± 0.301	0.165 ± 0.400	0.135 ± 0.274
($m_{\mathrm{cbf}} = 2$)	infeas. rate	6.3%	8.0%	10.4%	10.9%	10.9%	10.2%
iMPC-DHOCBF	mean / std (s)	0.131 ± 0.286	0.114 ± 0.260	0.109 ± 0.237	0.137 ± 0.316	0.173 ± 0.414	0.152 ± 0.317
($m_{\mathrm{cbf}} = 1$)	infeas. rate	6.3%	8.0%	10.4%	10.9%	10.9%	11.1%

DHOCBF uses relaxed linear CBF constraints (19), which reduce the feasible region (Fig. 2). This explains the slightly lower feasibility at small N, but as N grows, the flexible convergence criteria in iMPC-DHOCBF lead to significantly improved feasibility, as confirmed in Tab. 2.

B. CASE STUDY II: POINT ROBOT AVOIDANCE OF IRREGULAR OBSTACLES WITH UNKNOWN BOUNDARIES

Here we consider both convex and nonconvex obstacles. The nearest points to the robot on their boundaries are difficult to represent with continuous and accurate equations. We train a DNN for the SBD to predict the nearest points.

1) System Dynamics: We consider the same dynamics as in Sec. V-A with $\Delta t = 0.05$. System (34) is subject to the following state and input constraints:

$$\begin{aligned} \mathcal{X} &= \{\mathbf{x}_t \in \mathbb{R}^4 : -10 \cdot \mathcal{I}_{4 \times 1} \leq \mathbf{x}_t \leq 10 \cdot \mathcal{I}_{4 \times 1}\}, \\ \mathcal{U} &= \{\mathbf{u}_t \in \mathbb{R}^2 : -10 \cdot \mathcal{I}_{2 \times 1} \leq \mathbf{u}_t \leq 10 \cdot \mathcal{I}_{2 \times 1}\}. \end{aligned} \quad (35)$$

2) System Configuration: The five different start points are $(-2, -2)$, $(-2, 2)$, $(-0.8, -0.8)$, $(0, 0.8)$, $(1.2, 0.4)$ and the corresponding end points are $(2, 2)$, $(2, -2)$, $(0, 0.8)$, $(1.2, 0.4)$, $(0, 0)$, which are marked as solid dots in Fig. 8. The robot can stop when it reaches inside the target areas centered at end points with a radius of 0.1. The initial and reference linear speeds are 0 and the initial and reference heading angles are calculated by $\theta_0 = \theta_r = atan2(\frac{y_r - y_t}{x_r - x_t})$, where (x_t, y_t) denotes the current location and (x_r, y_r) denotes the end point. Five obstacles are displayed in red, with convex and nonconvex shapes. The other reference vectors are $\mathbf{u}_r = [0, 0]^T$ and $\omega_r = \mathcal{I}_{1 \times 10}$.

3) Map Processing: The scope of the map is $x \in [-2.5, 2.5]$ and $y \in [-2, 2]$. The coordinates of pixels on the boundaries of obstacles are manually chosen, and we limit the number of boundary pixels for each obstacle to 100.

4) SBD Training and Evaluation: Our neural network has 9 hidden layers and begins with 512 nodes in the first hidden layer, denoted as h_0. The subsequent layers' dimensions are defined by (29). Within the range of the map, we extracted 154993 data entries (random locations across the entire map) for training and another 25741 (random locations outside of obstacles) for testing. For p obstacles, we use the Mean Squared Error (MSE) equation (32) to evaluate the accuracy

of the SBD's predictions of the nearest points on obstacle boundaries relative to the robot.

5) DHOCBF: Since the predicted nearest points $\tilde{\mathbf{x}}_{t,k}^j$ can be obtained from the output of the SBD, we choose (27) as a linearized DHOCBF whose relative degree is 2. Note that five obstacles correspond to five linearized DHOCBFs. The sequence of linearized DHOCBF is defined by (14). From (21), we have $Z_{0,2} = \gamma_1 - 1$, $Z_{1,2} = 1$, $Z_{0,1} = 1$, $Z_{2,2} = Z_{1,1} = 0$.

6) MPC Design: The cost function of the CFTOC problem (33) consists of stage cost $q(\mathbf{x}_{t,k}^j, \mathbf{u}_{t,k}^j, \omega_{t,k,i}^j) = \sum_{k=0}^{N-1}(||\mathbf{x}_{t,k}^j - \mathbf{x}_r||_Q^2 + ||\mathbf{u}_{t,k}^j - \mathbf{u}_r||_R^2 + ||\omega_{t,k,i}^j - \omega_r||_S^2)$ and terminal cost $p(\mathbf{x}_{t,N}^j) = ||\mathbf{x}_{t,N}^j - \mathbf{x}_r||_P^2$, where $Q = P = 10 \cdot \mathcal{I}_4$, $R = \mathcal{I}_2$ and $S = 1000 \cdot \mathcal{I}_{10}$. In this case study, the slack variable $\omega_{t,k,i}^j$ is subject to the constraint in (20) for iMPC-DHOCBF to enhance feasibility while guaranteeing safety, as discussed in Thm. 2.

7) Convergence Criteria: We use the same absolute or relative convergence criteria from Sec. V-A-5. The hyperparameter related to the decay rate is set to a relatively smaller value as $\gamma_1 = 0.1$ to reduce the needed maximum iteration number which is set as $j_{\max} = 30$.

8) Solver and CPU Specs: For iMPC-DHOCBF, we used OSQP [44] in Python to solve the convex optimizations at all iterations. For model training and inference, we used Pytorch. The training was conducted on a Linux desktop equipped with a Nvidia RTX 4090 graphics card. For inference, we used a Linux laptop with an AMD Ryzen 7 5800U.

9) SBD Prediction Accuracy: The prediction accuracy of SBD is demonstrated in Figs. 6 and 7. In Fig. 6, we selected eight different locations from testing data on the map, each represented by a solid dot in a corresponding color. For these eight locations, there are five reference nearest points on the obstacle boundaries (diamonds) and the nearest points predicted by the SBD (plus symbols). When the robot is near the convex boundary of an obstacle, the predicted nearest points closely match the references. However, near concave boundaries, there is some deviation, though not substantial. Note that the predicted nearest point of the black dot is closer than the ground truth, demonstrating that the SBD can interpolate from sparse pixels to predict locally optimal nearest points, as discussed in Rem. 5. Figure 7 shows

TABLE 3: Statistical benchmark for computation time and feasibility for iMPC-DHOCBF with SBD. The randomized states and end points are shared among four approaches and the other irrelevant hyperparameters are fixed for all scenarios.

| Approaches		$N = 4$	$N = 8$	$N = 12$	$N = 16$	$N = 20$	$N = 24$
iMPC-DHOCBF	mean / std (s)	0.029 ± 0.020	0.065 ± 0.042	0.126 ± 0.069	0.202 ± 0.090	0.340 ± 0.126	0.446 ± 0.138
($m_{\text{cbf}} = 2, \gamma_1 = \gamma_2 = 0.1$)	infeas. rate	4.5%	5.4%	5.0%	6.0%	5.9%	6.4%
iMPC-DHOCBF	mean / std (s)	0.028 ± 0.020	0.053 ± 0.030	0.108 ± 0.055	0.167 ± 0.068	0.235 ± 0.082	0.374 ± 0.106
($m_{\text{cbf}} = 1, \gamma_1 = 0.1$)	infeas. rate	4.9%	5.4%	4.9%	5.4%	5.0%	5.3%
iMPC-DHOCBF	mean / std (s)	0.031 ± 0.020	0.073 ± 0.046	0.141 ± 0.070	0.242 ± 0.091	0.416 ± 0.113	0.545 ± 0.144
($m_{\text{cbf}} = 2, \gamma_1 = \gamma_2 = 0.2$)	infeas. rate	4.6%	4.9%	6.3%	7.5%	9.3%	10.9%
iMPC-DHOCBF	mean / std (s)	0.032 ± 0.024	0.071 ± 0.040	0.139 ± 0.055	0.217 ± 0.063	0.296 ± 0.063	0.435 ± 0.063
($m_{\text{cbf}} = 1, \gamma_1 = 0.2$)	infeas. rate	4.5%	4.6%	4.3%	4.9%	4.5%	5.0%
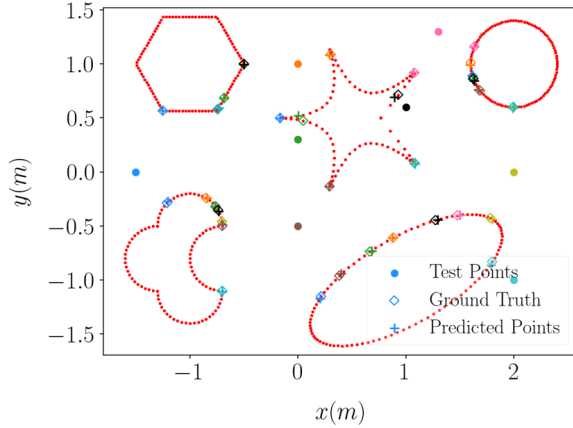
FIGURE 6: Demonstration of SBD prediction performance. Solid dots indicate the positions of eight robots. Diamonds show the ground-truth nearest points on obstacles, while plus signs denote the SBD predictions. Most predictions closely match the ground truth, except for the one corresponding to the green dot, which shows a larger deviation.

the relationship between the SBD's prediction error (MSE) and the robot's positions in 3D and projected views. Since testing data are from outside the obstacles, MSE within obstacles is not shown (darkest areas). The MSE is relatively small near convex boundaries (darker colors) and larger near concave boundaries (brighter colors). Higher MSE values are observed near concave boundaries and extend along certain radial lines due to the presence of multiple equidistant nearest points on the obstacle boundaries relative to these lines. During data collection, one of the equidistant points is randomly chosen as a reference. This randomness leads the trained SBD to predict a nearest point as a blend of equidistant candidates, causing noticeable deviation from the reference point and a high MSE.

10) Safe Trajectory Generation: Five safe closed-loop trajectories start from corresponding start points and end in corresponding target areas, shown in Fig. 8. We observe that even with multiple obstacles, including those with complex shapes like the sharp protrusions of a pentagram, the robot navigates safely in free space, regardless of proximity to obstacles or path length. Under the guidance of waypoints, the generated trajectories are smooth. They can be further

improved by increasing the number of waypoints, using adaptive warm starts, and refining convergence criteria.

11) Computation Time and Feasibility: In order to compare computational time and feasibility of our proposed iMPC-DHOCBF under different hyperparameters, 1000 independent randomized safe states are generated in state constraint $\mathcal{X}$ in (35). To make a fair comparison, the randomized states and end points are shared among four sets of hyperparameters and the other irrelevant hyperparameters are fixed for all scenarios. We can observe in Tab. 3 that as N increases, the computation time per step significantly increases. Similarly, when m_{cbf} increases or γ_1, γ_2 increase, the computation time per step also rises. The computation time is primarily affected by N and m_{cbf} because each call to the SBD consumes a certain amount of time. Increases in N and m_{cbf} linearly raise the number of SBD calls, thus significantly increasing the computation time per step. However, these hyperparameters are shown not to affect the infeasibility rate of the iMPC-DHOCBF method proportionally. Compared to the iMPC-DHOCBF in Tab. 2, iMPC-DHOCBF triggering SBD still maintains a fast computation speed per step and has a lower infeasibility rate, even in a more complex map.

C. CASE STUDY III: CIRCULAR ROBOT NAVIGATION IN A NARROW TRACK WITH UNKNOWN BOUNDARIES

Similar to Sec. V-B, we will train a DNN for the SBD to predict the nearest points.

1) System Dynamics: The robot is circular with a radius of 0.4, and its geometric center possesses the lateral vehicle model (see Eqs. (2.52), (2.53) in [51]) expressed by

$$
\begin{bmatrix} x_{t+1} - x_t \\ y_{t+1} - y_t \\ \psi_{t+1} - \psi_t \\ v_{t+1} - v_t \\ \beta_{t+1} - \beta_t \\ r_{t+1} - r_t \end{bmatrix} = \begin{bmatrix} v_t \cos(\psi_t + \beta_t)\Delta t \\ v_t \sin(\psi_t + \beta_t)\Delta t \\ r_t \Delta t \\ 0 \\ E_t \Delta t \\ F_t \Delta t \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \Delta t & 0 \\ 0 & \frac{C_f}{m(v_t + \epsilon)}\Delta t \\ 0 & \frac{C_f l_f}{I_z}\Delta t \end{bmatrix} \begin{bmatrix} u_{1,t} \\ u_{2,t} \end{bmatrix},
$$
(36)

where

$$
E_t = \frac{C_r l_r - C_f l_f}{m(v_t + \epsilon)^2} r_t - \frac{C_f + C_r}{m(v_t + \epsilon)} \beta_t + \frac{g \sin(\phi)}{v_t + \epsilon} - r_t,
$$
$$
F_t = \frac{1}{I_z}\left((-C_f l_f + C_r l_r)\beta_t - \frac{C_f l_f^2 + C_r l_r^2}{v_t + \epsilon} r_t \right).
$$
(37)

$\mathbf{x}_t = [x_t, y_t, \psi_t, v_t, \beta_t, r_t]^T$ captures the 2-D location, heading angle, linear speed, slip angle and yaw rate; $\mathbf{u}_t =$
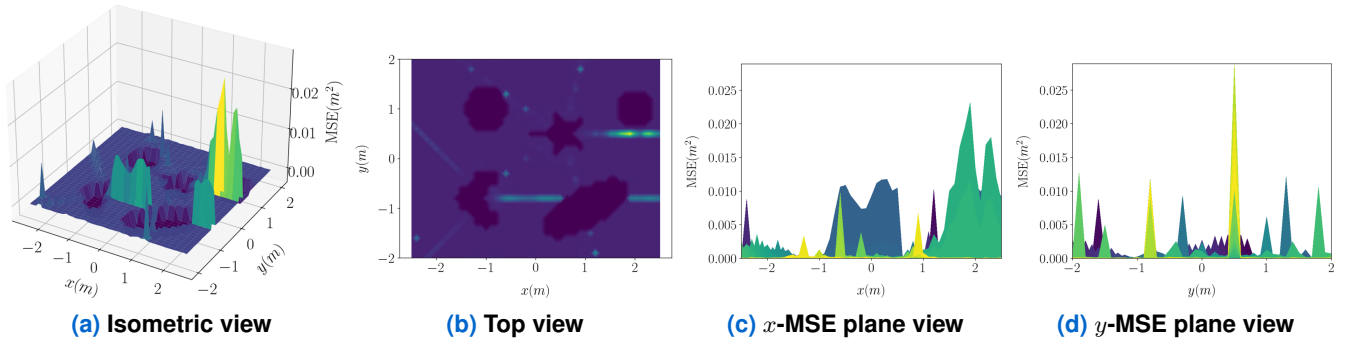
(a) Isometric view   **(b) Top view**   **(c) x-MSE plane view**   **(d) y-MSE plane view**

FIGURE 7: 3D visualization of the MSE of the testing data. x and y represent the positions of the testing points, while MSE represents the prediction error corresponding to the point (x, y). **The maximum MSE is 0.029 at** $(1.9, 0.5)$.
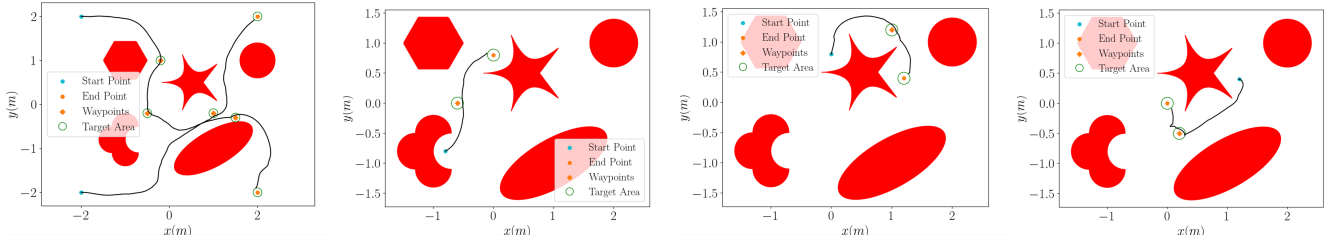

FIGURE 8: Five closed-loop trajectories (black) with different start and end points controlled by iMPC-DHOCBF with $m_{\mathbf{cbf}} = 1, N = 24, \gamma_1 = 0.1$. The obstacles are shown in red. The waypoints used in each subfigure are: $(-0.2, 1.0)$, $(-0.5, -0.2)$, $(1.0, -0.2)$, $(1.5, -0.3)$; $(-0.6, 0.0)$; $(1.0, 1.2)$; $(0.2, -0.5)$, respectively. The animation video can be found at https://youtu.be/G9S7y90LRig.

$[u_{1,t}, u_{2,t}]^T$ represents linear acceleration (u_1) and the front wheel steering angle (u_2), respectively. The system is discretized with $\Delta t = 0.05$. Other parameters are defined as $\phi = 0, m = 1500, g = 9.8, l_f = 1.2, l_r = 1.6, I_z = 2500, C_f = C_r = 10000, \epsilon = 11$. Let $\mathcal{X} \subset \mathbb{R}^6$ and $\mathcal{U} \subset \mathbb{R}^2$ denote the state and input constraint sets, respectively. We define $\mathbf{x}_t \in \mathcal{X}$ and $\mathbf{u}_t \in \mathcal{U}$ if and only if $x_t, y_t, \psi_t, v_t, r_t, u_{1,t}, u_{2,t} \in [-10, 10]$ and $\beta_t \in [-1, 1]$.

2) System Configuration: We plan to generate five connected closed-loop trajectories, where the start point of each trajectory is the end point of the previous one. The initial location and four waypoints marked as solid dots are $(4, -4.5), (2.5, -2), (2, 2), (-4, 2), (-7.5, -3.3)$ and the end point is $(-2.5, -4.25)$ in Fig. 10. The robot can terminate the current trajectory when its center reaches inside circles centered at each waypoint with a radius of 0.1 and then begin a second trajectory. The initial and reference linear speeds, slip angles and yaw rates are 0 and the initial and reference heading angles are calculated by $\psi_0 = \psi_r = atan2(\frac{y_r - y_t}{x_r - x_t})$, where (x_t, y_t) denotes current location and (x_r, y_r) denotes each waypoint. The robot's operating area is a narrow track enclosed by two bar-shaped obstacles with point contact connections. The width of the track varies within $[\sqrt{2}, 2]$, and the track shape consists of Z-shaped bends and S-shaped bends. The other reference vectors are $\mathbf{u}_r = [0, 0]^T$ and $\omega_r = \mathcal{I}_{1 \times 4}$.

3) Map Processing: The scope of the map is $x \in [-10, 6]$ and $y \in [-6, 6]$. We manually set the coordinates of pixels on the two obstacle boundaries in contact with the track, limiting each boundary to a maximum of 400 pixels. This is equivalent to enclosing the free track space with a closed shape formed by two boundary lines from the two obstacles.

4) SBD Training and Evaluation: Our neural network's structure is completely identical to that in Sec. V-B-4. Within the range of the map, we extracted 714875 data entries (random locations across the entire map) for training and another 48761 (random locations inside the track) for testing. We use the same MSE (32) to evaluate the accuracy of the SBD's predictions of the nearest points on obstacle boundaries relative to the robot.

5) DHOCBF: The design of DHOCBFs is exactly the same as that in Sec. V-B-5. Note that our robot is circular, therefore, equation (27), which imposes constraints on the robot's center position, should be shifted a distance equal to the radius toward the robot's center.

6) MPC Design: The cost function of the CFTOC problem (33) consists of stage cost $q(\mathbf{x}_{t,k}^j, \mathbf{u}_{t,k}^j, \omega_{t,k,i}^j) = \sum_{k=0}^{N-1}(||\mathbf{x}_{t,k}^j - \mathbf{x}_r||_Q^2 + ||\mathbf{u}_{t,k}^j - \mathbf{u}_r||_R^2 + ||\omega_{t,k,i}^j - \omega_r||_S^2)$ and terminal cost $p(\mathbf{x}_{t,N}^j) = ||\mathbf{x}_{t,N}^j - \mathbf{x}_r||_P^2$, where $Q = P = 10 \cdot \mathcal{I}_6, R = \mathcal{I}_2$ and $S = 1000 \cdot \mathcal{I}_4$. In this case study, the slack variable $\omega_{t,k,i}^j$ is subject to the constraint in (20) for

(a) Isometric view **(b)** Top view **(c)** x-MSE plane view **(d)** y-MSE plane view
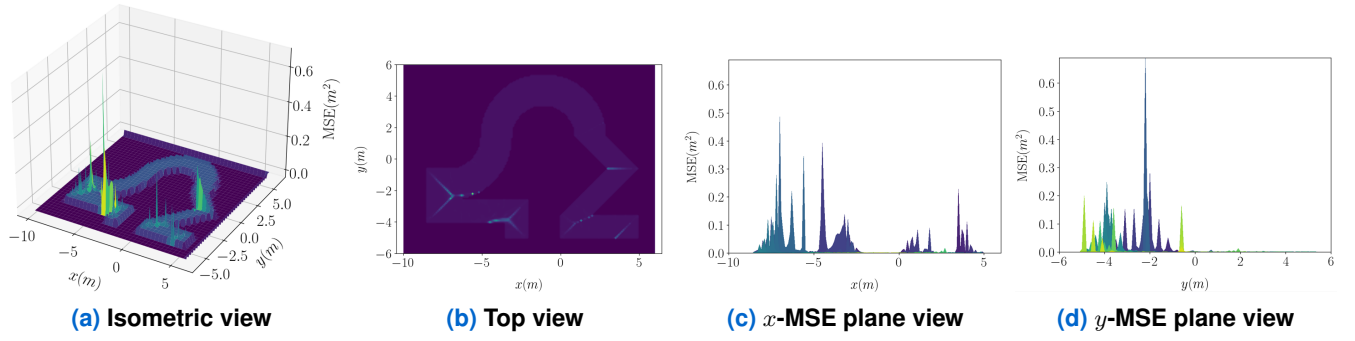
FIGURE 9: 3D visualization of the MSE of the testing data. x and y represent the positions of the testing points, while MSE represents the prediction error corresponding to the point (x, y). **The maximum MSE is 0.690 at** $(-5.5, -2.1)$.

iMPC-DHOCBF to enhance feasibility while guaranteeing safety, as discussed in Thm. 2.

7) Convergence Criteria: same as Sec. V-B-7.

8) Solver and CPU Specs: same as Sec. V-B-8.

9) SBD Prediction Accuracy: The prediction accuracy of SBD is demonstrated in Fig. 9, where we show the relationship between the SBD's prediction error (MSE) and the robot's various positions. Since the testing data was selected from the robot track, the MSE outside the track on the chart is devoid of values and the darkest. We can observe that when the robot's position is near the convex boundary of the track, the MSE of the predicted nearest point is relatively small (indicated by darker colors). Conversely, when the robot's position is near the concave boundary of the track, the MSE of the predicted nearest point is relatively large (indicated by brighter colors), which is similar to the observations made in Fig. 7. Unlike Fig. 7, not all concave boundary areas exhibit large prediction errors, e.g., the concave boundary sections of Z-shaped bends display a linear pattern of increasing MSE while concave boundary sections of the S-shaped bends do not show large MSE. This shows that the concavity of a boundary relative to a point outside the boundary also affects the MSE of nearest point prediction. The greater the concavity, the larger the prediction error. Additionally, we noted that the largest MSE is 0.690, which is much larger than the maximum MSE in Fig. 7. The peak MSEs in Fig. 9 are also significantly larger than that in Fig. 7. This is because the size of the map also affects the prediction error. Large maps often require significantly large quantities of training data to reduce prediction errors to very low levels. We can enhance safety using the approach described in Rem. 6, or adjust the slack-variable weight S to reduce the conservativeness introduced by prediction errors.

10) Safe Trajectory Generation: We evaluated the robot using two models: the vehicle model (36) and the unicycle model (34). The parameters and design of control strategy of the unicycle model are identical to those described in Sec. V-B. Five closed-loop trajectories—depicted in pink for the vehicle model and green for the unicycle model—connecting
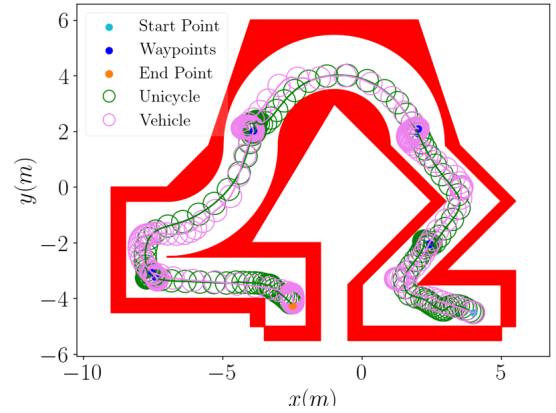

FIGURE 10: Five connected closed-loop trajectories (pink for the vehicle model and green for the unicycle model) are generated by iMPC-DHOCBF with $m_{cbf} = 1$, $N = 24$, and $\gamma_1 = 0.1$. Each trajectory starts where the previous ends. The narrow track is enclosed by two red obstacles. The method ensures safe navigation in tight spaces; animation is available at https://youtu.be/G9S7y90LRig.

the waypoints are shown in Fig. 10. We can see that even within a narrow track, sometimes requiring sharp turns, the robot governed by both models is still able to navigate safely, demonstrating the scalability of our method across different dynamic constraints. In Fig. 11, we can observe the trends in the robot's states and inputs as they change with each time step during its movement. Since the reference values for u_1, u_2 in the cost function are all zero, they fluctuate within a small range around zero as a baseline. All system variables are subject to the corresponding constraints $\mathcal{X}$ and $\mathcal{U}$, which highlights the safety and planning features under constraints of our implementation. In terms of computational complexity, the unicycle model takes an average of 0.6 seconds per time step, while the vehicle model takes 0.7 seconds. The slightly higher computation time of the vehicle model is mainly due to its more nonlinear dynamics (e.g., slip angle β and yaw rate r), with additional factors detailed in Sec.IV-E-6.
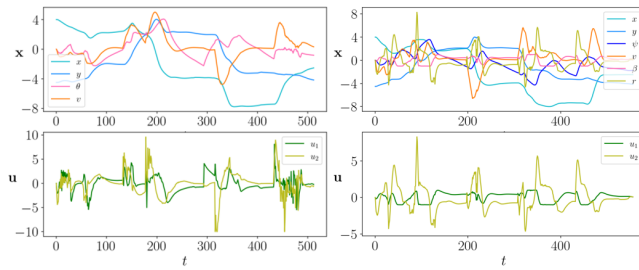
FIGURE 11: States and control inputs vary over time steps. These system variables are relative to the motion of the robot governed by the unicycle model (left) and the vehicle model (right) shown in Fig. 10 and are subject to the corresponding constraints $\mathcal{X}, \mathcal{U}$.

VI. CONCLUSION & FUTURE WORK

We proposed an iterative convex optimization procedure for safety-critical model predictive control design. Central to our approach is a learning-based safety boundary detector (SBD) to predict linearized safety boundaries for unsafe sets with arbitrary shapes, the transformation of these safety boundaries into discrete-time high-order control barrier functions (DHOCBFs), and relaxations for the system dynamics and for DHOCBF in the form of linearized constraints. We validate iMPC-DHOCBF on a unicycle robot in three environments (circular obstacles, irregular obstacles, and a narrow track) and extend it to a robot with vehicle dynamics in the narrow-track scenario. Our method outperforms the baseline in computation time and feasibility rate, and both models safely follow pre-designed waypoints through complex environments. There are still some limitations of iMPC-DHOCBF with SBD that could be ameliorated. One limitation of the proposed method is for locations near the concave boundary of an unsafe set, the SBD's prediction error for the nearest point remains relatively large. Another limitation is that our DNN-based SBD does not provide a strict a priori bound on its prediction error. Moreover, the feasibility of the optimization and system safety are not always guaranteed at the same time in the whole state space. We will address these limitations in future work.

REFERENCES

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

[2] A. D. Ames, X. Xu, J. W. Grizzle and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Trans. Autom. Control.*, vol. 62, no. 8, pp. 3861–3876, Aug. 2017, DOI: 10.1109/TAC.2016.2638961.

[3] J. Zeng, B. Zhang and K. Sreenath, "Safety-critical model predictive control with discrete-time control barrier function," in *Proc. 2021 Am. Control Conf.*, New Orleans, LA, USA, 2021, pp. 3882–3889.

[4] M. Diehl, H. G. Bock, and J. P. Schlöder, "A real-time iteration scheme for nonlinear optimization in optimal feedback control," *SIAM J. Control Optim.*, vol. 43, no. 5, pp. 1714–1736, Jan. 2005, DOI: 10.1137/S0363012902400713.

[5] G. Paolo, I. Ferrara, and L. Magni, "MPC for robot manipulators with integral sliding mode generation," *IEEE/ASME Trans. Mechatronics*, vol. 22, no. 3, pp. 1299–1307, June. 2017, DOI: 10.1109/T-MECH.2017.2674701.

[6] N. Scianca, D. De Simone, L. Lanari, and G. Oriolo, "MPC for humanoid gait generation: Stability and feasibility," *IEEE Trans. Robot.*, vol. 36, no. 4, pp. 1171–1188, Aug. 2020, DOI: 10.1109/TRO.2019.2958483.

[7] 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 *Proc. Robot.: Sci. Syst.*, New York City, NY, USA, 2022.

[8] T. D. Son and Q. Nguyen, "Safety-critical control for non-affine nonlinear systems with application on autonomous vehicle," in *Proc. 58th IEEE Conf. Decis. Control*, Nice, France, 2019, pp. 7623–7628.

[9] J. M. Eklund, J. Sprinkle, and S. S. Sastry, "Switched and symmetric pursuit/evasion games using online model predictive control with application to autonomous aircraft," *IEEE Trans. Control Syst. Technol.*, vol. 20, no. 3, pp. 604–620, May. 2012, DOI: 10.1109/TCST.2011.2136435.

[10] 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 *Proc. Eur. Control Conf.*, Zürich, Switzerland, 2013, pp. 4136–4141.

[11] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1:43 scale RC cars," *Optim. Control Appl. Methods*, vol. 36, no. 5, pp. 628–647, July. 2015.

[12] X. Zhang, A. Liniger, and F. Borrelli, "Optimization-based collision avoidance," *IEEE Trans. Control Syst. Technol.*, vol. 29, no. 3, pp. 972–983, May. 2020, DOI: 10.1109/TCST.2019.2949540.

[13] K. Galloway, K. Sreenath, A. D. Ames, and J. W. Grizzle, "Torque saturation in bipedal robotic walking through control Lyapunov function-based quadratic programs," *IEEE Access*, vol. 3, pp. 323–332, Apr. 2015, DOI: 10.1109/ACCESS.2015.2419630.

[14] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *Proc. 18th Eur. Control Conf.*, Naples, Italy, 2019, pp. 3420–3431.

[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 *Proc. 55th IEEE Conf. Decis. Control*, Las Vegas, NV, USA, 2016, pp. 827–834.

[16] Q. Nguyen and K. Sreenath, "Exponential control barrier functions for enforcing high relative-degree safety-critical constraints," in *Proc. 2016 Am. Control Conf.*, Boston, MA, USA, 2016, pp. 322–328.

[17] W. Xiao and C. Belta, "High-order control barrier functions," *IEEE Trans. Autom. Control*, vol. 67, no. 7, pp. 3655–3662, Jul. 2022, DOI: 10.1109/TAC.2021.3105491.

[18] X. Tan, W. S. Cortez, and D. V. Dimarogonas, "High-order barrier functions: Robustness, safety, and performance-critical control," *IEEE Trans. Autom. Control*, vol. 67, no. 6, pp. 3021–3028, Jun. 2021, DOI: 10.1109/TAC.2021.3089639.

[19] J. Usevitch and D. Panagou, "Adversarial resilience for sampled-data systems under high-relative-degree safety constraints," *IEEE Trans. Autom. Control*, vol. 68, no. 3, pp. 1537–1552, Mar. 2023, DOI: 10.1109/TAC.2022.3157791.

[20] W. Xiao, C. Belta, and C. G. Cassandras, "Adaptive control barrier functions," *IEEE Trans. Autom. Control*, vol. 67, no. 5, pp. 2267–2281, May. 2022, DOI: 10.1109/TAC.2021.3074895.

[21] S. Liu, W. Xiao, and C. A. Belta, "Auxiliary-variable adaptive control barrier functions for safety critical systems," in *Proc. 62nd IEEE Conf. Decis. Control*, Singapore, 2023, pp. 8602–8607.

[22] Q. Nguyen and K. Sreenath, "Optimal robust control for bipedal robots through control Lyapunov function based quadratic programs," in *Proc. Robot.: Sci. Syst.*, Rome, Italy, 2015.

[23] M. Jankovic, "Robust control barrier functions for constrained stabilization of nonlinear systems," *Automatica*, vol. 96, pp. 359–367, Oct. 2018.

[24] A. Clark, "Control barrier functions for stochastic systems," *Automatica*, vol. 130, p. 109–688, Aug. 2021.

[25] S. Liu, W. Xiao, and C. A. Belta, "Feasibility-guaranteed safety-critical control with applications to heterogeneous platoons," in *Proc. 63rd IEEE Conf. Decis. Control*, Milan, Italy, 2024, pp. 8066–8073.

[26] A. Agrawal and K. Sreenath, "Discrete control barrier functions for safety-critical control of discrete systems with application to bipedal robot navigation," in *Proc. Robot.: Sci. Syst.*, Cambridge, MA, USA, 2017.

[27] H. Ma, X. Zhang, S. E. Li, Z. Lin, Y. Lyu, and S. Zheng, "Feasibility enhancement of constrained receding horizon control using generalized control barrier function," in *Proc. IEEE Int. Conf. Ind. Cyber-Phys. Syst.*, 2021, pp. 551–557.

[28] Y. Xiong, D.-H. Zhai, M. Tavakoli, and Y. Xia, "Discrete-time control barrier function: High-order case and adaptive case," *IEEE Trans. Cybern.*, vol. 53, no. 5, pp. 3231–3239, May. 2023, DOI: 10.1109/T-CYB.2022.3170607.

[29] S. He, J. Zeng, and K. Sreenath, "Autonomous racing with multiple vehicles using a parallelized optimization with safety guarantee using control barrier functions," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2022, pp. 3444–3451.

[30] Z. Li, J. Zeng, A. Thirugnanam, and K. Sreenath, "Bridging model-based safety and model-free reinforcement learning through system identification of low dimensional linear models," in *Proc. Robot.: Sci. Syst.*, New York City, NY, 2022.

[31] J. Zeng, B. Zhang, Z. Li, and K. Sreenath, "Safety-critical control using optimal-decay control barrier function with guaranteed point-wise feasibility," in *Proc. Am. Control Conf.*, 2021, pp. 3856–3863.

[32] J. Zeng, Z. Li, and K. Sreenath, "Enhancing feasibility and safety of nonlinear model predictive control with discrete-time control barrier functions," in *Proc. 60th IEEE Conf. Decis. Control*, 2021, pp. 6137–6144.

[33] M. Srinivasan, A. Dabholkar, S. Coogan, and P. A. Vela, "Synthesis of control barrier functions using a supervised machine learning approach," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 7139–7145.

[34] M. Saveriano and D. Lee, "Learning barrier functions for constrained motion planning with dynamical systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 112–119.

[35] C. Peng, O. Donca, G. Castillo, and A. Hereid, "Safe bipedal path planning via control barrier functions for polynomial shape obstacles estimated using logistic regression," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2023, pp. 3649–3655.

[36] A. Robey, H. Hu, L. Lindemann, H. Zhang, D. V. Dimarogonas, S. Tu, and N. Matni, "Learning control barrier functions from expert demonstrations," in *Proc. 59th IEEE Conf. Decis. Control*, 2020, pp. 3717–3724.

[37] W. Lavanakul, J. Choi, K. Sreenath, and C. Tomlin, "Safety filters for black-box dynamical systems by learning discriminating hyperplanes," in *Proc. Learn. Dyn. Control Conf.*, 2024, pp. 1278–1291.

[38] S. Liu, J. Zeng, K. Sreenath, and C. A. Belta, "Iterative convex optimization for model predictive control with discrete-time high-order control barrier functions," in *Proc. Am. Control Conf.*, 2023, pp. 3368–3375.

[39] M. Sun and D. Wang, "Initial shift issues on discrete-time iterative learning control with system relative degree," *IEEE Trans. Autom. Control*, vol. 48, no. 1, pp. 144–148, Jan. 2003, DOI: 10.1109/TAC.2002.806668.

[40] A. Thirugnanam, J. Zeng, and K. Sreenath, "Duality-based convex optimization for real-time obstacle avoidance between polytopes with control barrier functions," in *Proc. Am. Control Conf.*, 2022, pp. 2239–2246.

[41] M. A. Kramer, "Autoassociative neural networks," *Comput. Chem. Eng.*, vol. 16, no. 4, pp. 313–328, Apr. 1992.

[42] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3D Euclidean signed distance fields for on-board MAV planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 1366–1373.

[43] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *Proc. Int. Conf. Comput. Vis. Theory Appl.*, 2009, pp. 331–340.

[44] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: An operator splitting solver for quadratic programs," *Math. Program. Comput.*, vol. 12, no. 4, pp. 637–672, Feb. 2020.

[45] L. T. Biegler and V. M. Zavala, "Large-scale nonlinear programming using IPOPT: An integrating framework for enterprise-wide dynamic optimization," *Comput. Chem. Eng.*, vol. 33, no. 3, pp. 575–582, Mar. 2009.

[46] J. Lofberg, "YALMIP: A toolbox for modeling and optimization in MATLAB," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2004, pp. 284–289.

[47] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin, "A general safety framework for learning-based control in uncertain robotic systems," *IEEE Trans. Autom. Control*, vol. 64, no. 7, pp. 2737–2752, Jul. 2019, DOI: 10.1109/TAC.2018.2876389.

[48] H. Wang and G. Zhao, "Risk-aware safe feedback motion planning in Gaussian splatting world," in *Proc. IEEE Int. Conf. Unmanned Syst. (ICUS)*, Nanjing, China, 2024, pp. 447–452.

[49] K. P. Wabersich, L. Hewing, A. Carron, and M. N. Zeilinger, "Probabilistic model predictive safety certification for learning-based control," *IEEE Trans. Autom. Control*, vol. 67, no. 1, pp. 176–188, Jan. 2021, DOI: 10.1109/TAC.2021.3049335.

[50] U. Rosolia and F. Borrelli, "Learning Model Predictive Control for Iterative Tasks: A Data-Driven Control Framework," *IEEE Trans. Autom. Control*, vol. 63, no. 7, pp. 1883–1896, Jul. 2018, DOI: 10.1109/TAC.2017.2753460.

[51] R. Rajamani, *Vehicle Dynamics and Control*, Springer Science & Business Media, 2011.

[52] F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control for Linear and Hybrid Systems*, Cambridge University Press, 2017.

[53] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *Proc. Int. Conf. Machine Learning (ICML)*, 2017, pp. 22–31.

[54] C. Belta, B. Yordanov, and E. A. Gol, *Formal Methods for Discrete-Time Dynamical Systems*, vol. 89. Springer, 2017.

[55] Z. Qin, K. Zhang, Y. Chen, J. Chen, and C. Fan, "Learning safe multi-agent control with decentralized neural barrier certificates," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2021.

[56] Y. Tsuzuku, I. Sato, and M. Sugiyama, "Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 31, 2018.

[57] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient SMT solver for verifying deep neural networks," in *Proc. Int. Conf. Computer Aided Verification (CAV)*, 2017, pp. 97–117.

[58] E. Wong and Z. Kolter, "Provable defenses against adversarial examples via the convex outer adversarial polytope," in *Proc. Int. Conf. Machine Learning (ICML)*, 2018, pp. 5286–5295.

ZHE HUANG received his M.S. degree in Robotics from Boston University, Boston, USA, in 2023 and an M.S. degree in Computer Science from the University of Dayton, Dayton, USA, in 2017. He earned his B.S. degree in Electro-Optics from Jianghan University, Wuhan, China, in 2014. His research interests include deep learning, computer vision, robot perception, and robot control.
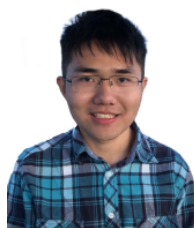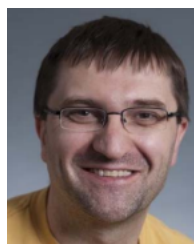

JUN ZENG (Member, IEEE) received his Ph.D. in Control and Robotics from the Department of Mechanical Engineering at the University of California, Berkeley, USA, in 2022, his Dipl. Ing. degree from École Polytechnique, France, in 2017, and his B.S.E. degree from Shanghai Jiao Tong University, China, in 2016. His research interests lie at the intersection of optimization, control, planning, and learning, with applications to various robotics platforms.

KOUSHIL SREENATH (Senior Member, IEEE) received the M.S. degree in Applied Mathematics and the Ph.D. degree in Electrical Engineering and Systems from the University of Michigan, Ann Arbor, MI, USA, in 2011. He is an Assistant Professor of Mechanical Engineering with the University of California, Berkeley, Berkeley, CA, USA. He was previously an Assistant Professor with Carnegie Mellon University, Pittsburgh, PA, USA, from 2013 to 2017. His research interests include the intersection of highly dynamic robotics and applied nonlinear control.

CALIN A. BELTA (Fellow, IEEE) received his Ph.D. in Mechanical Engineering from the University of Pennsylvania, Philadelphia, PA, USA, in 2003. He is currently the Brendan Iribe Endowed Professor of Electrical and Computer Engineering and Computer Science at the University of Maryland, College Park, MD, USA. His research focuses on control theory and formal methods, with particular emphasis on hybrid and cyber-physical systems, synthesis and verification, and applications in robotics and biology.

SHUO LIU (Student Membe, IEEE) received his M.S. degree in Mechanical Engineering from Columbia University, New York, NY, USA, in 2020 and his B.Eng. degree in Mechanical Engineering from Chongqing University, Chongqing, China, in 2018. He is currently a Ph.D. candidate in Mechanical Engineering at Boston University, Boston, USA. His research interests include optimization, nonlinear control, deep learning, and robotics.