Learning to Refine Input Constrained Control Barrier Functions via Uncertainty-Aware Online Parameter Adaptation

Taekyung Kim, Robin Inho Kee and Dimitra Panagou

Abstract— **Control Barrier Functions (CBFs) have become powerful tools for ensuring safety in nonlinear systems. However, finding valid CBFs that guarantee persistent safety and feasibility remains an open challenge, especially in systems with input constraints. Traditional approaches often rely on manually tuning the parameters of the class $\mathcal{K}$ functions of the CBF conditions a priori. The performance of CBF-based controllers is highly sensitive to these fixed parameters, potentially leading to overly conservative behavior or safety violations. To overcome these issues, this paper introduces a learning-based optimal control framework for online adaptation of Input Constrained CBF (ICCBF) parameters in discrete-time nonlinear systems. Our method employs a probabilistic ensemble neural network to predict the performance and risk metrics, as defined in this work, for candidate parameters, accounting for both epistemic and aleatoric uncertainties. We propose a two-step verification process using Jensen-Rényi Divergence and distributionally-robust Conditional Value at Risk to identify valid parameters. This enables dynamic refinement of ICCBF parameters based on current state and nearby environments, optimizing performance while ensuring safety within the verified parameter set. Experimental results demonstrate that our method outperforms both fixed-parameter and existing adaptive methods in robot navigation scenarios across safety and performance metrics.** [1]

I. INTRODUCTION

Safety-critical optimal control is one of the fundamental challenges in robotics, e.g., robots need to navigate safely around obstacles while optimizing various criteria such as energy consumption. Control Barrier Functions (CBFs) have been developed in recent years as a tool to encode safety for nonlinear systems [1]–[5]. High Order CBFs extend this concept to systems with constraint functions of higher relative degree with respect to the system dynamics [6]–[9]. However, a significant challenge lies in finding valid CBFs under input constraints [10]. Functions that do not provide persistent safety and feasibility guarantees are often referred to as *candidate* CBFs [11].

Research in this field has taken three main directions. The first focuses on synthesizing CBFs directly [11]–[14], while the second learns CBFs through function approximation [15]–[20]. Another stream of research concentrates on tuning

Taekyung Kim is with the Department of Robotics, University of Michigan, Ann Arbor, MI, 48109, USA taekyung@umich.edu

Robin Inho Kee is with the Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI, 48109, USA inhokee@umich.edu

Dimitra Panagou is with the Department of Robotics and Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI, 48109, USA dpanagou@umich.edu

[1] Our project page can be found at: https://www.taekyung.me/online-adaptive-cbf

the class $\mathcal{K}$ function parameters of CBFs. Many existing works verify and optimize CBF parameters offline and fix them during deployment [10], [21], [22]. However, these fixed parameters may lead to overly conservative motions in new or changing environments. While several works introduce adaptive CBFs [23]–[26], they address dynamics model uncertainty rather than safety and performance.

Recent efforts in the third category have focused on adapting CBFs' class $\mathcal{K}$ function parameters online [27]. Some approaches [28], [29] optimize decay rates of CBF constraints to guarantee point-wise feasibility, but fall short of guaranteeing safety. More recently, learning-based methods have become increasingly popular. However, their safety guarantees rely on unrealistic assumptions such as unbounded control inputs [30] or perfect learned models [31], [32].

To address these limitations, we propose a learning-based framework for online adaptation of CBF class $\mathcal{K}$ function parameters in input-constrained systems. We leverage a probabilistic ensemble neural network (PENN) to predict the safety and performance characteristics of potential parameter configurations while considering both epistemic and aleatoric uncertainties. The parameters are dynamically adapted based on the current system state and nearby environmental configurations, simultaneously ensuring safety and optimizing performance for navigation tasks. We quantify epistemic uncertainty, which originates from insufficient training data, using Jensen-Rényi Divergence, discarding predictions with low confidence. To account for aleatoric uncertainty, arising from sources such as data ambiguity and measurement noise, we model predicted outputs as distributions. We then employ distributionally robust Conditional Value at Risk to assess the risk associated with these predictions.

We highlight the key contributions of this paper, which differentiate our approach from existing methods:

- We introduce a formal definition of discrete-time Input Constrained CBFs (ICCBFs), specifically parameterized by class $\mathcal{K}$ functions.
- We propose a verification process for using a PENN model to predict the class $\mathcal{K}$ functions of interest, ensuring *robustness to both types of uncertainties*.
- We develop an online adaptive MPC-ICCBF framework under *input constraints*, dynamically adapting class $\mathcal{K}$ functions to simultaneously ensure safety and optimize performance.

II. PRELIMINARIES

Consider a general discrete-time nonlinear system:

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

where $\boldsymbol{x}_t \in \mathcal{X} \subset \mathbb{R}^n$ is the state at time step $t \in \mathbb{Z}^+$, and $\boldsymbol{u}_t \in \mathcal{U} \subset \mathbb{R}^m$ is the control input at time step t, with $\mathcal{U}$ being a set of admissible controls for System (1). The function $f : \mathcal{X} \times \mathcal{U} \to \mathbb{R}^n$ is assumed to be locally Lipschitz continuous with respect to both $\boldsymbol{x}_t$ and $\boldsymbol{u}_t$.

Definition 1 (Discrete-Time CBF [33]). *Let $\mathcal{S} = \{\boldsymbol{x} \in \mathcal{X} \mid h(\boldsymbol{x}) \geq 0\}$, where $h : \mathcal{X} \to \mathbb{R}$ is a continuous function. The function h is a discrete-time CBF for System* (1) *if there exists a class $\mathcal{K}$ function $\alpha(\cdot)$ satisfying $\alpha(z) < z$ for all $z > 0$ such that*

$$\sup_{\boldsymbol{u}_t \in \mathcal{U}} [\Delta h(\boldsymbol{x}_t, \boldsymbol{u}_t)] + \alpha(h(\boldsymbol{x}_t)) \geq 0 \quad \forall \boldsymbol{x}_t \in \mathcal{X}, \quad (2)$$

where $\Delta h(\boldsymbol{x}_t, \boldsymbol{u}_t) = h(\boldsymbol{x}_{t+1}) - h(\boldsymbol{x}_t)$.

Lemma 1. *Given a **valid** discrete-time CBF h satisfying condition* (2) *with the associated set $\mathcal{S}$, any control input $\boldsymbol{u}_t \in K_{\mathrm{cbf}}(\boldsymbol{x}_t)$, with $K_{\mathrm{cbf}}(\boldsymbol{x}_t) \coloneqq \{\boldsymbol{u}_t \in \mathcal{U} : \Delta h(\boldsymbol{x}_t, \boldsymbol{u}_t) + \alpha(h(\boldsymbol{x}_t)) \geq 0\}$, renders the set $\mathcal{S}$ forward invariant for System* (1). *The proof can be found in [34, Theorem 1].*

The discrete-time CBF condition ensures that the system state remains within the safe set $\mathcal{S}$ for all future time steps, provided that the initial state lies within $\mathcal{S}$.

Remark 1. *The class $\mathcal{K}$ function α can be chosen as a linear function with coefficient $\gamma \in (0,1)$. In this case, it follows that $h(\boldsymbol{x}_t) \geq (1 - \gamma)^t h(\boldsymbol{x}_0)$ for all $t \in \mathbb{Z}^+$. For $0 < \gamma \leq 1$, forward invariance of the set is still preserved (which is obviously true when $\gamma = 1$), and the discrete-time CBF becomes the discrete-time exponential CBF [33].*

III. PROBLEM FORMULATION

In this section, we formally define the discrete-time Input Constrained CBFs (ICCBFs), and use this framework to formulate the main problems addressed in this paper.

A. Input Constrained Control Barrier Functions

The concept of CBFs has been generalized to HOCBFs in both continuous-time [7] and discrete time [35] settings, which can be used for constraints of high relative degree. ICCBFs [10] further generalize HOCBFs, allowing for constraints of higher relative degrees and non-uniform relative degrees for control inputs.

More importantly, the ICCBF structure explicitly addresses the problem of generating **valid** CBFs under input constraints, i.e., $\mathcal{U} \neq \mathbb{R}^m$. Concretely, suppose that the safe set $\mathcal{S}$ defined by h cannot be rendered forward invariant by any feedback control input $\boldsymbol{u} \in K_{\mathrm{cbf}}(\boldsymbol{x})$, since there exist some states where it requires $\boldsymbol{u} \notin \mathcal{U}$ to render safe.

Using the same function h in Definition 1, we define a series of functions $b_i : \mathcal{X} \to \mathbb{R}$, $i = 0, \ldots, r$, as:

$$b_0(\boldsymbol{x}_t; \boldsymbol{\alpha}) \coloneqq h(\boldsymbol{x}_t) \quad (3a)$$

$$b_1(\boldsymbol{x}_t; \boldsymbol{\alpha}) \coloneqq \inf_{\boldsymbol{u}_t \in \mathcal{U}} [\Delta b_0(\boldsymbol{x}_t, \boldsymbol{u}_t; \boldsymbol{\alpha})] + \alpha_0(b_0(\boldsymbol{x}_t; \boldsymbol{\alpha})) \quad (3b)$$

$$\vdots$$

$$b_r(\boldsymbol{x}_t; \boldsymbol{\alpha}) \coloneqq \inf_{\boldsymbol{u}_t \in \mathcal{U}} [\Delta b_{r-1}(\boldsymbol{x}_t, \boldsymbol{u}_t; \boldsymbol{\alpha})] + \alpha_{r-1}(b_{r-1}(\boldsymbol{x}_t; \boldsymbol{\alpha})), \quad (3c)$$

where $\boldsymbol{\alpha} = \{\alpha_0, \ldots, \alpha_r\}$ is a set of class $\mathcal{K}$ functions with $\alpha_i : [0, a) \to [0, \infty)$ satisfying $\alpha_i(z) < z$ for $i = 0, \ldots, r$ and all $z > 0$, and $\Delta b_i(\boldsymbol{x}_t, \boldsymbol{u}_t; \boldsymbol{\alpha}) = b_i(\boldsymbol{x}_{t+1}; \boldsymbol{\alpha}) - b_i(\boldsymbol{x}_t; \boldsymbol{\alpha})$. We also define a series of sets $\mathcal{C}_i$ associated with these functions as:

$$\mathcal{C}_1(\boldsymbol{\alpha}) \coloneqq \{\boldsymbol{x}_t \in \mathcal{X} : b_0(\boldsymbol{x}_t; \boldsymbol{\alpha}) \geq 0\} = \mathcal{S} \quad (4a)$$

$$\mathcal{C}_2(\boldsymbol{\alpha}) \coloneqq \{\boldsymbol{x}_t \in \mathcal{X} : b_1(\boldsymbol{x}_t; \boldsymbol{\alpha}) \geq 0\} \quad (4b)$$

$$\vdots$$

$$\mathcal{C}_r(\boldsymbol{\alpha}) \coloneqq \{\boldsymbol{x}_t \in \mathcal{X} : b_{r-1}(\boldsymbol{x}_t; \boldsymbol{\alpha}) \geq 0\}. \quad (4c)$$

Definition 2 (Inner Safe Set). *A non-empty closed set $\mathcal{C}^*(\boldsymbol{\alpha}) \subseteq \mathcal{S}$ is an inner safe set for System* (1) *defined as [10]:*

$$\mathcal{C}^*(\boldsymbol{\alpha}) \coloneqq \bigcap_{i=0}^{r} \mathcal{C}_i(\boldsymbol{\alpha}), \quad (5)$$

and it is conditioned on the class $\mathcal{K}$ functions $\boldsymbol{\alpha}$.

Definition 3 (Discrete-Time ICCBF). *The function b_r is a discrete-time ICCBF on $\mathcal{C}^*(\boldsymbol{\alpha})$ for System* (1) *if there exist class $\mathcal{K}$ functions $\boldsymbol{\alpha} = \{\alpha_0, \ldots, \alpha_r\}$ such that*

$$\sup_{\boldsymbol{u}_t \in \mathcal{U}} [\Delta b_r(\boldsymbol{x}_t, \boldsymbol{u}_t; \boldsymbol{\alpha})] + \alpha_r(b_r(\boldsymbol{x}_t; \boldsymbol{\alpha})) \geq 0 \quad \forall \boldsymbol{x}_t \in \mathcal{C}^*(\boldsymbol{\alpha}). \quad (6)$$

Note, the definition does not require condition (6) to hold for $\boldsymbol{x}_t \in \mathcal{C}_r(\boldsymbol{\alpha})$, instead, for a subset $\mathcal{C}^*(\boldsymbol{\alpha}) \subset \mathcal{C}_r(\boldsymbol{\alpha})$.

Remark 2. *One can observe that the validity of the ICCBF function b_r in satisfying condition* (6) *depends on the design of class $\mathcal{K}$ functions $\boldsymbol{\alpha}$.*

For notational simplicity, we denote the ICCBF constraint function as:

$$\psi(\boldsymbol{x}_t, \boldsymbol{u}_t; \boldsymbol{\alpha}) \coloneqq \Delta b_r(\boldsymbol{x}_t, \boldsymbol{u}_t; \boldsymbol{\alpha}) + \alpha_r(b_r(\boldsymbol{x}_t; \boldsymbol{\alpha})). \quad (7)$$

Lemma 2. *Given the input constrained discrete-time system* (1), *if b_r is a discrete-time ICCBF, then any control input $\boldsymbol{u}_t \in K_{\mathrm{iccbf}}(\boldsymbol{x}_t; \boldsymbol{\alpha})$, with $K_{\mathrm{iccbf}}(\boldsymbol{x}_t; \boldsymbol{\alpha}) \coloneqq \{\boldsymbol{u}_t \in \mathcal{U} : \psi(\boldsymbol{x}_t, \boldsymbol{u}_t; \boldsymbol{\alpha}) \geq 0\}$, renders the set $\mathcal{C}^*(\boldsymbol{\alpha}) \subseteq \mathcal{S}$ forward invariant. The proof is similar to [10, Theorem 1].*

Remark 3. *We can solve the following optimization problem:*

$$\zeta^* = \underset{\boldsymbol{x}_t \in \mathcal{C}^*(\boldsymbol{\alpha})}{\text{minimize}} \sup_{\boldsymbol{u}_t \in \mathcal{U}} [\Delta b_r(\boldsymbol{x}_t, \boldsymbol{u}_t; \boldsymbol{\alpha})] + \alpha_r(b_r(\boldsymbol{x}_t; \boldsymbol{\alpha})). \quad (8)$$

By the definition, b_r is an ICCBF if and only if the optimization problem has a feasible solution $\zeta^* \geq 0$. However, this can only be used to invalidate b_r as a valid ICCBF. The search over integers r and class $\mathcal{K}$ functions $\boldsymbol{\alpha}$ to find valid ICCBFs can be complicated, similar to the challenge of finding Lyapunov functions in general.

B. Problem Setup

In the original ICCBF approach, which is common in many CBF applications, the objective is to identify a fixed set of class $\mathcal{K}$ functions $\boldsymbol{\alpha}$ that make the function b_r a valid ICCBF for all states in the inner safe set $\mathcal{C}^*$. However, this approach presents two main challenges: a) it is difficult

to find such a set, and b) it can be overly conservative, potentially restricting the system's performance.

To address these limitations, we propose an online adaptive approach that seeks to find a set of class $\mathcal{K}$ functions that make the ICCBF candidate function b_r to be locally valid (to be formally defined in this section) at each control iteration, based on the current state and the surrounding environment. This method eliminates the need to exhaustively search over all possible class $\mathcal{K}$ functions $\boldsymbol{\alpha}$ and validate (8) with the corresponding set $\mathcal{C}^*(\boldsymbol{\alpha})$. Moreover, it allows the ICCBF to be dynamically tailored to the specific conditions the system encounters in real time.

Consider a navigation problem for a nonlinear system in the form of (1) in a static environment $\mathcal{W}$ containing information about a set of obstacles $\mathcal{O}$. We define this set as: $\mathcal{W} = \{\mathcal{O}_j : (\boldsymbol{z}_j, l_{\text{obs},j})\}_{j=1}^N$, where $\boldsymbol{z}_j \in \mathbb{R}^2$ is the spatial coordinate and $l_{\text{obs},j} \in \mathbb{R}^+$ is the radius of the j-th obstacle. Let us denote the MPC controller associated with an ICCBF b_r that depends on class $\mathcal{K}$ functions $\boldsymbol{\alpha}$ as $\pi_{\boldsymbol{\alpha}} : \mathcal{C}^* \to \mathcal{U}$ and $\boldsymbol{u} = \pi_{\boldsymbol{\alpha}}(\boldsymbol{x}) \in K_{\text{iccbf}}(\boldsymbol{x}; \boldsymbol{\alpha})$.

MPC-ICCBF:

$$J^*(\boldsymbol{x}_t; \boldsymbol{\alpha}) = \min_{\boldsymbol{u}_{t:t+H-1|t}} F(\boldsymbol{x}_{t+H|t}) +$$

$$\sum_{k=0}^{H-1} L(\boldsymbol{x}_{t+\tau|t}, \boldsymbol{u}_{t+\tau|t}) \quad (9a)$$

$$\text{s.t. } \boldsymbol{x}_{t+\tau+1|t} = f(\boldsymbol{x}_{t+\tau|t}, \boldsymbol{u}_{t+\tau|t}), \ k=0,...,H-1 \quad (9b)$$

$$\boldsymbol{u}_{t+\tau|t} \in \mathcal{U}, \ k=0,...,H-1 \quad (9c)$$

$$\psi(\boldsymbol{x}_{t+\tau|t}, \boldsymbol{u}_{t+\tau|t}; \boldsymbol{\alpha}) \geq 0, \ k=0,...,H-1. \quad (9d)$$

The notation $\boldsymbol{x}_{t+\tau|t}$ represents the predicted state vector at time step $t + \tau$, based on the information available at time step t. This prediction is obtained by initializing the state $\boldsymbol{x}_{t|t} = \boldsymbol{x}_t$ and applying the control sequence $\boldsymbol{u}_{t:t+N-1|t}$ to the system dynamics (9b). We use linear quadratic cost function both for the stage cost L and the terminal cost F in (9a) to navigate towards the goal location. The parameter H represents the prediction horizon. More details can be referred to in [36]–[38].

Let us define the set of states of the system starting from a state $\boldsymbol{x}_t$ for a fixed time T_f, driven by the controller $\pi_{\boldsymbol{\alpha}}$ parameterized by $\boldsymbol{\alpha}$:

$$\mathcal{P}_t(\boldsymbol{\alpha}; \boldsymbol{x}_t, \mathcal{E}_t, \pi_{\boldsymbol{\alpha}}) := \{\boldsymbol{x}_{t+\tau}\}_{\tau=0}^{T_f} \quad (10)$$

where $\boldsymbol{x}_{t+\tau+1} = f(\boldsymbol{x}_{t+\tau}, \pi_{\boldsymbol{\alpha}}(\boldsymbol{x}_{t+\tau}))$. $\mathcal{E}_t \subset \mathcal{W}$ is the local map at time step t, defined as $\mathcal{E}_t = \{\mathcal{O}_j \in \mathcal{W} : \|\boldsymbol{z}_t - \boldsymbol{z}_j\|_2 \leq l_{\text{range}} + l_{\text{obs},j}, j \in \{1, \ldots, N\}\}$, where $\boldsymbol{z}_t \in \mathbb{R}^2$ is the robot's spatial coordinate, and l_{range} is a predefined sensing range.

Definition 4 (Locally Valid Class $\mathcal{K}$ Functions). *Given a set $\mathcal{P}_t(\boldsymbol{\alpha}) \subset \mathcal{C}^*(\boldsymbol{\alpha})$, the class $\mathcal{K}$ functions $\boldsymbol{\alpha}$ are **locally valid** class $\mathcal{K}$ functions for the candidate ICCBF b_r on the set $\mathcal{P}_t(\boldsymbol{\alpha})$ if the following condition is satisfied:*

$$\zeta = \underset{\boldsymbol{x}_t \in \mathcal{P}_t(\boldsymbol{\alpha})}{\text{minimize}} \ \sup_{\boldsymbol{u}_t \in \mathcal{U}} \ [\Delta b_r(\boldsymbol{x}_t, \boldsymbol{u}_t; \boldsymbol{\alpha})] + \alpha_r(b_r(\boldsymbol{x}_t; \boldsymbol{\alpha})) \geq 0.$$
$$(11)$$

Note, the local validity of the class $\mathcal{K}$ functions $\boldsymbol{\alpha}$ depends on current state $\boldsymbol{x}_t$, nearby environment $\mathcal{E}_t$, and the controller $\pi_{\boldsymbol{\alpha}}$ affected by $\boldsymbol{\alpha}$ itself.

Example 1. Consider a scalar discrete-time double integrator system with input constraints:

$$\begin{bmatrix} x_{t+1} \\ v_{t+1} \end{bmatrix} = \begin{bmatrix} x_t + v_t \\ v_t + \boldsymbol{u}_t \end{bmatrix}, \quad \mathcal{U} = [-1, 1]. \quad (12)$$

The safety constraint $\mathcal{S} = \{\boldsymbol{x} \in \mathbb{R}^2 : x \geq -2\}$, i.e., $b_0(\boldsymbol{x}_t; \boldsymbol{\alpha}) = h(\boldsymbol{x}_t) = x_t + 2$, represents a wall-type obstacle located at $x = -2$. Now consider a candidate ICCBF $b_1(\boldsymbol{x}_t; \boldsymbol{\alpha}) = \inf_{\boldsymbol{u}_t \in \mathcal{U}} [\Delta b_0(\boldsymbol{x}_t, \boldsymbol{u}_t; \boldsymbol{\alpha})] + \alpha_0(b_0(\boldsymbol{x}_t; \boldsymbol{\alpha}))$ and linear class $\mathcal{K}$ functions $\boldsymbol{\alpha} = \{\gamma_0, \gamma_1\}$ with $\gamma_0, \gamma_1 \in (0, 1)$.

At $\mathcal{P}_t = \{\boldsymbol{x}_t\}$ where $x_t = -1, v_t = -2$, the ICCBF constraint value in (7) with $\gamma_0 = \gamma_1 = 0.1$ is:

$$\sup_{\boldsymbol{u}_t \in \mathcal{U}} [\Delta b_1(\boldsymbol{x}_t, \boldsymbol{u}_t; \boldsymbol{\alpha})] + \gamma_1 b_1(\boldsymbol{x}_t; \boldsymbol{\alpha}) = \sup_{\boldsymbol{u}_t \in \mathcal{U}} [u_t] - 0.39 > 0,$$
$$(13)$$

satisfying local validity. Contrary, $\gamma_0 = \gamma_1 = 0.5$ are not not locally valid since $\sup_{\boldsymbol{u}_t \in \mathcal{U}} [u_t] - 1.75 < 0$. However, these class $\mathcal{K}$ functions can be locally valid at a different state, e.g., $x_t = -1, v_t = -1$, as the value becomes $\sup_{\boldsymbol{u}_t \in \mathcal{U}} [u_t] - 0.75 > 0$.

Problem 1 (Finding Locally Valid Class $\mathcal{K}$ Functions). Given the current state $\boldsymbol{x}_t$, nearby environment $\mathcal{E}_t$, and a pre-defined controller π, find a set of locally valid class $\mathcal{K}$ functions for the candidate ICCBF b_r:

$$\mathfrak{A}_{\text{valid}}(\boldsymbol{x}_t, \mathcal{E}_t, \pi) := \{\boldsymbol{\alpha} : \zeta(\boldsymbol{x}_t, \mathcal{E}_t, \pi_{\boldsymbol{\alpha}}) \geq 0\} \quad (14)$$

where ζ is defined as in (11).

Based on the solution of Problem 1, we can ensure safety in the smaller inner set $\mathcal{P}_t(\boldsymbol{\alpha}) \subset \mathcal{C}^*(\boldsymbol{\alpha})$ by choosing locally valid class $\mathcal{K}$ functions. As we iterate through the MPC process, we can update the set of locally valid class $\mathcal{K}$ functions based on the current state and nearby environment. This allows us to focus on maximizing the MPC performance by selecting optimal parameters from the valid set.

Problem 2 (Performance Optimization). Given the set of locally valid class $\mathcal{K}$ functions $\mathfrak{A}_{\text{valid}}(\boldsymbol{x}_t, \mathcal{E}_t, \pi)$, optimize the class $\mathcal{K}$ functions such that maximizes the performance by minimizing the MPC objective function (9a). Formally, solve at each time step t:

$$\boldsymbol{\alpha}_t^* = \underset{\boldsymbol{\alpha} \in \mathfrak{A}_{\text{valid}}(\boldsymbol{x}_t, \mathcal{E}_t, \pi)}{\arg\min} J^*(\boldsymbol{x}_t; \boldsymbol{\alpha}) \quad (15)$$

IV. METHODOLOGY

A. Safety Loss Density Function

In this work, we assume a robot modeled as a dynamic unicycle with state $\boldsymbol{x}_t = [x_t, y_t, \theta_t, v_t]^\top$, where x_t and y_t represent the position, θ_t is the heading angle, and v_t is the linear velocity, and the control inputs $\boldsymbol{u}_t = [a_t, \omega_t]^\top$ are the acceleration and the angular velocity, respectively.

In collision avoidance scenarios in robot navigation, the distances to the obstacles are not the sole factor of safety.

The safety hard constraint for obstacle collision avoidance can be simply encoded as the signed distance function:

$$h(\boldsymbol{x}_t) = (x_t - x_{\text{obs}})^2 + (y_t - y_{\text{obs}})^2 - (l_{\text{robot}} + l_{\text{obs}})^2 \geq 0, \quad (16)$$

where x_{obs} and y_{obs} are the coordinates of the obstacle. The terms l_{robot} and l_{obs} represent the radii of the robot and the obstacle, respectively. It is obvious that a controller with a hard constraint like (16) cannot ensure safety from all initial conditions. The ICCBF constraint value in (9d) is a useful metric to measure the safety margin with respect to the current robot state $\boldsymbol{x}_t$ and the system dynamics (1). However, the rate of change of the original hard constraint h in a discrete-time setting is computed via the time difference of subsequent value, it does not directly encode the direction of the robot's motion. Therefore, inspired by [39], we introduce a safety loss density function that incorporates both spatial state and directional information to provide a more comprehensive measure of collision risk.

Let $\Phi : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ be a positive semi-definite density function that describes the safety loss of a particular robot state $\boldsymbol{x}$ and control input $\boldsymbol{u}$:

$$\Phi(\boldsymbol{x}, \boldsymbol{u}) := \max_j \frac{\lambda_j(\boldsymbol{x}, \boldsymbol{u})}{\beta_j(\boldsymbol{x}) d(\boldsymbol{z}, \boldsymbol{z}_j)^2 + 1}, \quad (17)$$

where $d(\boldsymbol{z}, \boldsymbol{z}_j) := \|\boldsymbol{z} - \boldsymbol{z}_j\|_2 - l_{\text{robot}} - l_{\text{obs}}$, and $\boldsymbol{z}, \boldsymbol{z}_j \in \mathbb{R}^2$ are the spatial location of the robot and the j-th among N obstacles.

$\lambda_j : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ determines the peak height of the density function, and it is defined as:

$$\lambda_j(\boldsymbol{x}, \boldsymbol{u}) := \lambda_1 e^{-\lambda_2 \psi_j(\boldsymbol{x}, \boldsymbol{u}; \boldsymbol{\alpha})}, \quad (18)$$

where $\psi_j(\boldsymbol{x}, \boldsymbol{u}; \boldsymbol{\alpha})$ is the ICCBF constraint value (9d) for the j-th nearby obstacle. We use $r = 1$ to construct the ICCBF candidate given that the function h has relative degrees of 2 for both control inputs. $\lambda_1 \in \mathbb{R}$ determines the nominal peak height, and $\lambda_2 \in \mathbb{R}$ defines the risk decay rate.

$\beta_j : \mathcal{X} \to \mathbb{R}$ influences the dissipation rate of the density function based on the directional motion of the robot:

$$\beta_j(\boldsymbol{x}) := \beta_1 e^{\beta_2 (\cos(\Delta\theta_j) + 1)}, \quad (19)$$

where $\Delta\theta_j$ is the angle between the robot's motion direction and the vector pointing from the robot to the j-th obstacle. $\beta_1, \beta_2 \in \mathbb{R}$ are the parameters similar to (18).

By defining (17)-(19), we can measure the collision risk based on the robot state.

B. Training Data Generation - Offline Sampling

To train a prediction model that can infer the safety and performance of the MPC-ICCBF controller based on the state of the robot and the choice of the class $\mathcal{K}$ functions, we collect a dataset $\mathcal{D}$ by generating robot trajectory $\mathcal{P}_0(\boldsymbol{\alpha}; \boldsymbol{x}_0, \mathcal{E}_0, \pi_{\boldsymbol{\alpha}})$ as in (10) offline, similar to [21].

We uniformly sample the initial state of the robot $\boldsymbol{x}_0 = [x_0, y_0, \theta_0, v_0]^\top$ and the positions of N obstacles, while at least one obstacle is placed in the path towards the fixed goal location, requiring the robot to actively avoid it. We also sample linear class $\mathcal{K}$ functions γ_0, γ_1 according to

Remark 1. The initial distance to obstacle d_0 is computed as $d(\boldsymbol{z}_0, \boldsymbol{z}_j)$, where $\boldsymbol{z}_0 = [x_0, y_0]^\top$ and $\boldsymbol{z}_j$ is the location of the obstacle that imposes the greatest safety loss value in (17) at $\boldsymbol{x}_0$ and $\boldsymbol{u} = [0, 0]^\top$. $\Delta\theta_0$ represents the relative angle as defined as in (19).

Based on the configuration above $(d_0, \Delta\theta_0, v_0, \gamma_0, \gamma_1)$, we navigate the robot by solving the MPC-ICCBF problem (9) and record two metrics as the ground truth for prediction. To solve Problem 1, we compute the risk level $\phi \in \mathbb{R}$: $\phi = \max_{t \in [0, T_{\max}]} \Phi(\boldsymbol{x}_t, \boldsymbol{u}_t)$, where $T_{\max}$ is the maximum simulation time. To approximately solve Problem 2, we compute the deadlock time $\delta \in \mathbb{R}$, which is an accumulated time segment where the robot exhibits a smaller velocity than a pre-defined threshold v_{thr}. If the robot stops until $T_{\max}$, we set it as the maximum value $\delta = \delta_{\max}$. A training data pair consists of the input $X = [d_0, \Delta\theta_0, v_0, \gamma_0, \gamma_1]^\top$ and the ground truth $Y = [\phi, \delta]^\top$.

C. Probabilistic Ensemble Neural Network Model

We use Probabilistic Ensemble Neural Network (PENN), denoted as $\mathbf{F}$, to predict the output vector $Y = [\phi, \delta]^\top$. It offers a principled way to quantify both aleatoric and epistemic uncertainties [40]–[42].

Each neural network model $\mathbf{E}_b$ of the ensemble members outputs a Gaussian distribution conditioned on the model input X:

$$\mathbf{E}_b(X; \boldsymbol{\theta}_b) \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta}_b}(X), \boldsymbol{\Sigma}_{\boldsymbol{\theta}_b}(X)), \quad (20)$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are the mean and the diagonal covariance parameterized by the model parameter $\boldsymbol{\theta}_b$. Given a collected dataset $\mathcal{D}$, each probabilistic model is trained with the Gaussian Negative Log-Likelihood (NLL) loss function [42]. Given the ensemble of B models $\mathfrak{E} = \{\mathbf{E}_1, \ldots, \mathbf{E}_B\}$, $B \geq 2$, the predicted vector $\hat{Y}$ follows a Gaussian Mixture Model (GMM) distribution:

$$\hat{Y} \sim \mathbf{F}(X; \boldsymbol{\theta}_{1:B}) = \sum_{b=1}^{B} w_b \mathbf{E}_b(X; \boldsymbol{\theta}_b), \ 0 \leq w_b \leq 1. \quad (21)$$

Here, we simply use equal weights $w_b = \frac{1}{B}, b = 1, \ldots, B$.

Since the initial weights of each ensemble member are randomly initialized individually, disagreement among the predictions serves as a measure of epistemic uncertainty.

D. Uncertainty-Aware Class $\mathcal{K}$ Functions Verification

Now we propose the solution for Problem 1 by considering both types of uncertainties.

1) Epistemic Uncertainty Quantification: We evaluate the *confidence* of the prediction by measuring epistemic uncertainty. We employ Jensen-Rényi Divergence (JRD) with quadratic Rényi entropy [43], which has a closed-form expression of the divergence of a GMM [42], [44]:

$$D(X; \mathfrak{E}) = -\log \left(\frac{1}{B^2} \sum_{b,c}^{B} \mathfrak{D}_{(b,c)} \right) + \frac{1}{B} \sum_b^{B} \log \left[\mathfrak{D}_{(b,b)} \right], \quad (22)$$

$$\mathfrak{D}_{(b,c)} := \frac{1}{|\mathfrak{V}|^{\frac{1}{2}}} \exp \left(-\frac{1}{2} \boldsymbol{\Delta}^\top \mathfrak{V}^{-1} \boldsymbol{\Delta} \right), \quad (23)$$

$\mathfrak{V} := \Sigma_{\boldsymbol{\theta}_b} + \Sigma_{\boldsymbol{\theta}_c}$ and $\boldsymbol{\Delta} := \boldsymbol{\mu}_{\boldsymbol{\theta}_b} - \boldsymbol{\mu}_{\boldsymbol{\theta}_c}$. If the JRD $D(X)$ of the prediction of a given input X is greater than the pre-defined threshold D_{thr}, it is deemed to be out-of-distribution.

2) Distributionally Robust Risk Assessment: We then evaluate the risk of the predicted risk level $\hat{\phi} \in \hat{Y}$, which follows a GMM distribution due to aleatoric uncertainty. To this end, we first establish the connection between the predicted risk level and the local validity of class $\mathcal{K}$ functions. The condition (11) can be extended as:

$$\zeta = \underset{\boldsymbol{x}_t \in \mathcal{P}_t(\boldsymbol{\alpha})}{\text{minimize}} \ \underset{\boldsymbol{u}_t \in \mathcal{U}}{\sup} \ [\Delta b_r(\boldsymbol{x}_t, \boldsymbol{u}_t; \boldsymbol{\alpha})] + \alpha_r(b_r(\boldsymbol{x}_t; \boldsymbol{\alpha})) \quad (24)$$

$$\geq \underset{\boldsymbol{x}_t \in \mathcal{P}_t(\boldsymbol{\alpha})}{\min} \ \psi(\boldsymbol{x}_t, \boldsymbol{u}_t; \boldsymbol{\alpha}) \geq 0. \quad (25)$$

By further transforming this condition (25) and using the definition of safety loss density function Φ (17) and risk level $\phi = \max_{t \in [0, T_{\max}]} \Phi(\boldsymbol{x}_t, \boldsymbol{u}_t)$, we can derive:

$$\phi = \underset{\boldsymbol{x}_t \in \mathcal{P}_t(\boldsymbol{\alpha})}{\max} \ \underset{j}{\max} \ \frac{\lambda_1 e^{-\lambda_2 \psi(\boldsymbol{x}_t, \boldsymbol{u}_t; \boldsymbol{\alpha})}}{\beta_1 e^{\beta_2(\cos(\Delta\theta_j)+1)} d(\boldsymbol{z}, \boldsymbol{z}_j)^2 + 1} \leq$$

$$\underset{\boldsymbol{x}_t \in \mathcal{P}_t(\boldsymbol{\alpha})}{\max} \ \underset{j}{\max} \ \frac{\lambda_1}{\beta_1 d(\boldsymbol{z}, \boldsymbol{z}_j)^2 + 1} \leq \bar{\phi} := \frac{\lambda_1}{\beta_1 d_{\min}^2 + 1}, \quad (26)$$

where $d_{\min}$ is the closest allowable distance to obstacles, considering the robot's physical dimensions and a safety buffer. Therefore, if the risk level satisfies condition (26), i.e., $\phi \leq \bar{\phi}$, then condition (11) holds, indicating that the class $\mathcal{K}$ functions are locally valid.

Given that the predicted risk level $\hat{\phi}$ follows a GMM distribution due to ensemble predictions, we employ the distributionally robust Conditional Value at Risk (CVaR) as a risk metric. For notational simplicity, we slightly abuse the notation in this section by treating the risk level ϕ as the output vector Y. For $\hat{\phi} \sim \mathbf{E}_b(X)$, we define the Value at Risk (VaR) as:

$$\text{VaR}_\epsilon^{\mathbf{E}_b(X)}(\hat{\phi}) := \inf\{\nu \in \mathbb{R} \mid \text{Prob}^{\mathbf{E}_b(X)}(\hat{\phi} > \nu) \leq \epsilon\}. \quad (27)$$

VaR represents the potential risk level with the allowable probability ϵ. We then define the CVaR:

$$\text{CVaR}_\epsilon^{\mathbf{E}_b(X)}(\hat{\phi}) := \inf \left\{ \eta \in \mathbb{R} \mid \eta + \frac{1}{\epsilon} \mathbb{E}_{\mathbf{E}_b(X)} \left[(\hat{\phi} - \eta)^+ \right] \right\}, \quad (28)$$

where $(\cdot)^+ := \max\{\cdot, 0\}$. CVaR is interpreted as the expected value over VaR, and it adheres to a group of axioms crucial for rational risk assessment [45].

Finally, we compute distributionally robust CVaR by treating the ensemble set of distributions $\mathfrak{E}$ as the ambiguity set [46], [47]:

$$\underset{\mathbf{E}_b \in \mathfrak{E}}{\sup} \text{CVaR}_\epsilon^{\mathbf{E}_b(X)}(\hat{\phi}) \leq \bar{\phi} \Rightarrow \underset{\mathbf{E}_b \in \mathfrak{E}}{\sup} \text{VaR}_\epsilon^{\mathbf{E}_b(X)}(\hat{\phi}) \leq \bar{\phi} \quad (29)$$

$$\Leftrightarrow \underset{\mathbf{E}_b \in \mathfrak{E}}{\inf} \text{Prob}^{\mathbf{E}_b(X)}(\hat{\phi} \leq \bar{\phi}) \geq 1 - \epsilon. \quad (30)$$

By (30), the constraint $\hat{\phi} \leq \bar{\phi}$ is deemed as distributionally robust with the probability of $1 - \epsilon$.

By evaluating these two uncertainties, (22) and (29), we can ensure that the prediction from the PENN model is both **confident** and satisfies the **local validity** condition (11).

Algorithm 1: Online Adaptive MPC-ICCBF

Given: $\boldsymbol{\alpha}_0^*$: Initial class $\mathcal{K}$ functions; $\pi_{(\cdot)}$: MPC-ICCBF controller;
$\mathbf{F}$: PENN model with trained parameters $\boldsymbol{\theta}_{1:B}$;
for $t = 1, \ldots, T_{\max}$ **do**
    $\boldsymbol{x}_t, \mathcal{E}_t \leftarrow \texttt{Sensor}()$;
    $\mathfrak{A}_{\text{cand}} = \{\boldsymbol{\alpha}^{(k)}\}_{k=1}^K \leftarrow \texttt{SampleCandidates}(\boldsymbol{\alpha}_{t-1}^*)$;
    $\mathbf{X} = \{X^{(k)}\}_{k=1}^K \leftarrow \texttt{CreateInputs}(\boldsymbol{x}_t, \mathcal{E}_t, \mathfrak{A}_{\text{cand}})$;
    $\mathfrak{E}, \hat{\mathbf{Y}} = \{\hat{Y}^{(k)}\}_{k=1}^K \leftarrow \mathbf{F}(\mathbf{X}; \boldsymbol{\theta}_{1:B})$;
    $\mathfrak{A}_{\text{valid}} = \{\boldsymbol{\alpha} : D(X; \mathfrak{E}) < D_{\text{thr}}$
                       $\wedge \sup_{\mathbf{E}_b \in \mathfrak{E}} \text{CVaR}_\epsilon^{\mathbf{E}_b(X)}(\hat{\phi}) \leq \bar{\phi}\}$;
    $\boldsymbol{\alpha}_t^* \leftarrow \arg\min_{\boldsymbol{\alpha} \in \mathfrak{A}_{\text{valid}}} \hat{\delta}$;
    $\boldsymbol{u}_t \leftarrow \pi_{\boldsymbol{\alpha}_t^*}(\boldsymbol{x}_t)$;
    $\texttt{ApplyControlInput}(\boldsymbol{u}_t)$;

E. Online Parameter Adaptation

Integrating the concepts presented in Sec. IV-A-IV-D, we now introduce an online adaptive MPC-ICCBF algorithm (see Alg. 1) that dynamically adapts the class $\mathcal{K}$ functions to ensure safety and optimize performance in robot navigation scenarios.

Initially, for $t = 0$, we set $\boldsymbol{\alpha}_0^* = \{\gamma_{0,0}^*, \gamma_{1,0}^*\}$ to the numerically minimum values required to satisfy $\boldsymbol{x}_0 \in \mathcal{C}^*(\boldsymbol{\alpha})$ as per Lemma 2. At each subsequent time step, the algorithm generates K candidate class $\mathcal{K}$ function parameters $\mathfrak{A}_{\text{cand}} = \{\boldsymbol{\alpha}^{(k)}\}_{k=1}^K$ by sampling uniformly around the previous optimal values. These candidates form a batch of inputs $\mathbf{X} = \{X^{(k)}\}_{k=1}^K$ for the PENN model $\mathbf{F}$. The model then predicts GMM distributions of the risk level and the deadlock time for each input. To identify locally valid class $\mathcal{K}$ functions $\mathfrak{A}_{\text{valid}}$, the algorithm filters these predictions based on two criteria as described in Sec. IV-D.

To approximately solve Problem 2, the algorithm selects the class $\mathcal{K}$ functions $\boldsymbol{\alpha}_t^*$ from $\mathfrak{A}_{\text{valid}}$ that has the minimum predicted deadlock time $\hat{\delta}$. This minimizes the chance of overly restrictive CBF constraints impeding navigation [48]. These parameters are then updated in the MPC-ICCBF (9) problem to compute the control input $\boldsymbol{u}_t$. By repeating this process at each time step, it continuously refines the ICCBF parameters based on the current state and the predictions from the PENN model, thereby optimizing performance while maintaining safety.

Remark 4. *Assuming the discretized time step Δt in the MPC is sufficiently small such that $\boldsymbol{x}_{t+1} \in \mathcal{P}_t(\boldsymbol{\alpha}_t^*)$. By continuously updating the locally valid class $\mathcal{K}$ functions $\boldsymbol{\alpha}_t^* \in \mathfrak{A}_{\text{valid}}$, the system remains safe by the controller $\pi_{\boldsymbol{\alpha}_t^*} \in K_{\text{iccbf}}(\boldsymbol{x}_t; \boldsymbol{\alpha}_t^*)$ rendering the set $\mathcal{P}_t(\boldsymbol{\alpha}_t^*) \subset \mathcal{C}^*(\boldsymbol{\alpha}_t^*)$ forward invariant.*

V. RESULTS

A. Experimental Setup

We consider a system has constrained inputs (9c) as $a \in [-0.5, 0.5]$ m/s^2 and $\omega \in [-0.5, 0.5]$ rad/s. The discretization time step and the MPC sampling time are set to $\Delta t = 0.05$ s. The quadratic costs in (9a) are tuned such that the controller converges to the goal state without severe overshooting when there are no obstacles.
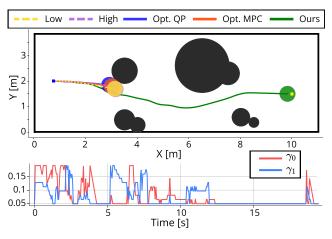
Fig. 1: Visualization of robot trajectories generated by five different approaches and class $\mathcal{K}$ function parameters refined over time by our online adaptive MPC-ICCBF method. The blue and yellow squares represent the start and goal location. The black circles represent the obstacles.
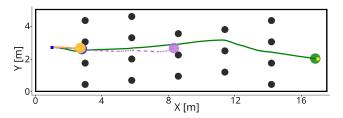

Fig. 2: Visualization of robot trajectories in a more complex environment.

For training data, we generate trajectories with $N = 1$ obstacle in this paper. The robot and obstacle radii are set to $l_{\text{robot}} = 0.3$ m and $l_{\text{obs}} = 0.2$ m, respectively. The probability ϵ in CVaR constraint (29) is set to 0.05. In total, we generate $M = 16,808$ training samples offline. To simulate observation and model uncertainties, we add 3% of i.i.d. Gaussian noise to $d_0, \Delta\theta_0, v_0$ in the input data X. The PENN model $\mathbf{F}$ comprises $B = 3$ ensemble models, each has 5 MLP layers with ReLU activation function. The rest of the implementation details follow [42]. More implementation details can be found in our public code repository.[2]

B. Online Adaptive MPC-ICCBF

We compare the performance of our method with four different approaches: MPC-ICCBF using 1) "Low" ($\gamma_0 = \gamma_1 = 0.01$) and 2) "High" ($\gamma_0 = \gamma_1 = 0.2$) *fixed* class $\mathcal{K}$ function parameters, 3) Optimal-decay CBF-QP [28] that can optimize class $\mathcal{K}$ functions *online* (abbreviated as "Opt. QP"), and 4) "Optimal-decay MPC-CBF" [29] where it incorporates the same optimization technique as [28] within the MPC structure (abbreviated as "Opt. MPC"). For 3) and 4), we optimize both γ_0 and γ_1 and penalize them in the cost function. For our method, we empirically set the adaptation range as $\gamma_0, \gamma_1 \in (0.01, 0.2)$.

We test these approaches in two different environments. Fig. 1 and Fig. 2 illustrate the trajectories generated by each method. To quantitatively assess the effectiveness of

[2]https://github.com/tkkim-robot/online_adaptive_cbf

TABLE I: Performance comparison of five approaches across two environments. Metrics include collision rate, goal-reaching rate, and average reach time. Environment #1 corresponds to Fig. 1, and environment #2 to Fig. 2.

	Collision Rate		Reach Rate		Time [s]	
	#1	#2	#1	#2	#1	#2
Low	10.2%	32.2%	89.8%	67.8%	54.4	125.9
High	100%	100%	0%	0%	N/A	N/A
Opt. QP [28]	0%	0%	0%	0%	N/A	N/A
Opt. MPC [29]	13.0%	38.9%	87.0%	61.1%	51.8	116.5
Ours	0%	0%	100%	100%	26.3	81.9

our method, we further conduct 108 navigation trials with uniformly varied initial states $\boldsymbol{x}_0$ in both environments. We evaluate three metrics: collision rate with obstacles, goal-reaching rate, and average reach time when the robot successfully reaches the goal (see Table I).

The conservative approach, using "Low" ICCBF parameters, is significantly affected by the initial nearby obstacle due to the ICCBF constraints. In some cases, the high initial velocities $v_0 \in \boldsymbol{x}_0$ result in states that fall outside the inner safe set defined by fixed class $\mathcal{K}$ functions, leading to collisions. The aggressive approach with "High" ICCBF parameters becomes infeasible in early iterations and eventually collides with obstacles. While "Opt. QP" shows zero collisions, it is prone to deadlock. "Opt. MPC" avoids infeasibility but still results in collisions, as it only guarantees the point-wise feasibility of the solution rather than persistent safety. In contrast, our proposed method actively refines the class $\mathcal{K}$ function parameters online based on the PENN model's predictions. It successfully reaches the goal without collisions or deadlocks, achieving the highest goal-reaching rate in both scenarios and demonstrating its ability to optimize performance while ensuring safety.

We also conducted hardware experiments comparing the five methods. These experiments, along with more detailed analysis, can be found in our accompanying video.

VI. CONCLUSION

In this paper, we present an online class $\mathcal{K}$ function parameters adaptation approach in discrete-time ICCBFs. To this end, we first introduce the concept of locally valid class $\mathcal{K}$ functions, which ensure forward invariance for a smaller subset of the inner safe set. By leveraging a PENN model with a novel two-step verification process that accounts for both epistemic and aleatoric uncertainties, our method can dynamically refine the ICCBF parameters to ensure safety and optimize MPC performance. Experimental results demonstrate that it outperforms the compared baselines in collision avoidance and goal-reaching rates across various scenarios. Future work may focus on enhancing multi-obstacle handling through graph neural network feature encoding, which could improve scalability to varying numbers of obstacles. Also, extending our method to higher-dimensional systems, such as quadrotors, is another area of interest.

REFERENCES

[1] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *IEEE Conference on Decision and Control (CDC)*, 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 Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2017.

[3] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control Barrier Functions: Theory and Applications," in *European Control Conference (ECC)*, 2019, pp. 3420–3431.

[4] B. Li, S. Wen, Z. Yan, G. Wen, and T. Huang, "A Survey on the Control Lyapunov Function and Control Barrier Function for Nonlinear-Affine Control Systems," *IEEE/CAA Journal of Automatica Sinica*, vol. 10, no. 3, pp. 584–602, 2023.

[5] K. Garg, J. Usevitch, J. Breeden, M. Black, D. Agrawal, H. Parwana, and D. Panagou, "Advances in the Theory of Control Barrier Functions: Addressing practical challenges in safe control synthesis for autonomous and robotic systems," *Annual Reviews in Control*, vol. 57, p. 100945, 2024.

[6] Q. Nguyen and K. Sreenath, "Exponential Control Barrier Functions for enforcing high relative-degree safety-critical constraints," in *American Control Conference (ACC)*, 2016, pp. 322–328.

[7] W. Xiao and C. Belta, "Control Barrier Functions for Systems with High Relative Degree," in *IEEE Conference on Decision and Control (CDC)*, 2019, pp. 474–479.

[8] J. Breeden and D. Panagou, "High Relative Degree Control Barrier Functions Under Input Constraints," in *IEEE Conference on Decision and Control (CDC)*, 2021, pp. 6119–6124.

[9] X. Tan, W. S. Cortez, and D. V. Dimarogonas, "High-Order Barrier Functions: Robustness, Safety, and Performance-Critical Control," *IEEE Transactions on Automatic Control*, vol. 67, no. 6, pp. 3021–3028, 2022.

[10] D. R. Agrawal and D. Panagou, "Safe Control Synthesis via Input Constrained Control Barrier Functions," in *IEEE Conference on Decision and Control (CDC)*, 2021, pp. 6113–6118.

[11] S. Tonkens and S. Herbert, "Refining Control Barrier Functions through Hamilton-Jacobi Reachability," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2022, pp. 13 355–13 362, iSSN: 2153-0866.

[12] A. Robey, H. Hu, L. Lindemann, H. Zhang, D. V. Dimarogonas, S. Tu, and N. Matni, "Learning Control Barrier Functions from Expert Demonstrations," in *IEEE Conference on Decision and Control (CDC)*, 2020, pp. 3717–3724.

[13] J. Lee, J. Kim, and A. D. Ames, "A Data-driven Method for Safety-critical Control: Designing Control Barrier Functions from State Constraints," in *arXiv preprint arXiv:2312.07786*, 2023.

[14] X. Ding, H. Wang, Y. Ren, Y. Zheng, C. Chen, and J. He, "Online Control Barrier Function Construction for Safety-Critical Motion Control of Manipulators," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 54, no. 8, pp. 4761–4771, 2024.

[15] C. Dawson, Z. Qin, S. Gao, and C. Fan, "Safe Nonlinear Control Using Robust Neural Lyapunov-Barrier Functions," in *Conference on Robot Learning (CoRL)*, 2021.

[16] J. J. Choi, D. Lee, K. Sreenath, C. J. Tomlin, and S. L. Herbert, "Robust Control Barrier–Value Functions for Safety-Critical Control," in *IEEE Conference on Decision and Control (CDC)*, 2021, pp. 6814–6821.

[17] S. Liu, C. Liu, and J. Dolan, "Safe Control Under Input Limits with Neural Control Barrier Functions," in *Conference on Robot Learning (CoRL)*, 2022.

[18] C. Dawson, S. Gao, and C. Fan, "Safe Control With Learned Certificates: A Survey of Neural Lyapunov, Barrier, and Contraction Methods for Robotics and Control," *IEEE Transactions on Robotics*, vol. 39, no. 3, pp. 1749–1767, Jun. 2023.

[19] O. So, Z. Serlin, M. Mann, J. Gonzales, K. Rutledge, N. Roy, and C. Fan, "How to Train Your Neural Control Barrier Function: Learning Safety Filters for Complex Input-Constrained Systems," in *International Conference on Robotics and Automation (ICRA)*, 2024.

[20] W. Lavanakul, J. J. Choi, K. Sreenath, and C. J. Tomlin, "Safety Filters for Black-Box Dynamical Systems by Learning Discriminating Hyperplanes," in *Learning for Dynamics and Control (L4DC)*, 2024.

[21] W. Xiao, C. A. Belta, and C. G. Cassandras, "Feasibility-Guided Learning for Constrained Optimal Control Problems," in *IEEE Conference on Decision and Control (CDC)*, 2020, pp. 1896–1901.

[22] X. Wang, "Ensuring Safety of Learning-Based Motion Planners Using Control Barrier Functions," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4773–4780, 2022.

[23] A. J. Taylor, V. D. Dorobantu, H. M. Le, Y. Yue, and A. D. Ames, "Episodic Learning with Control Lyapunov Functions for Uncertain Robotic Systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 6878–6884.

[24] A. J. Taylor and A. D. Ames, "Adaptive Safety with Control Barrier Functions," in *American Control Conference (ACC)*, 2020, pp. 1399–1405.

[25] M. Black, E. Arabi, and D. Panagou, "A Fixed-Time Stable Adaptation Law for Safety-Critical Control under Parametric Uncertainty," in *European Control Conference (ECC)*, 2021, pp. 1328–1333.

[26] W. Xiao, C. Belta, and C. G. Cassandras, "Adaptive Control Barrier Functions," *IEEE Transactions on Automatic Control*, vol. 67, no. 5, pp. 2267–2281, 2022.

[27] H. Parwana, A. Mustafa, and D. Panagou, "Trust-based Rate-Tunable Control Barrier Functions for Non-Cooperative Multi-Agent Systems," in *IEEE Conference on Decision and Control (CDC)*, 2022, pp. 2222–2229.

[28] J. Zeng, B. Zhang, Z. Li, and K. Sreenath, "Safety-Critical Control using Optimal-decay Control Barrier Function with Guaranteed Pointwise Feasibility," in *American Control Conference (ACC)*, 2021, pp. 3856–3863.

[29] J. Zeng, Z. Li, and K. Sreenath, "Enhancing Feasibility and Safety of Nonlinear Model Predictive Control with Discrete-Time Control Barrier Functions," in *IEEE Conference on Decision and Control (CDC)*, 2021, pp. 6137–6144.

[30] H. Ma, B. Zhang, M. Tomizuka, and K. Sreenath, "Learning Differentiable Safety-Critical Control using Control Barrier Functions for Generalization to Novel Environments," in *European Control Conference (ECC)*, 2022, pp. 1301–1308.

[31] W. Xiao, T.-H. Wang, R. Hasani, M. Chahine, A. Amini, X. Li, and D. Rus, "BarrierNet: Differentiable Control Barrier Functions for Learning of Safe Robot Control," *IEEE Transactions on Robotics*, pp. 1–19, 2023, conference Name: IEEE Transactions on Robotics.

[32] Z. Gao, G. Yang, and A. Prorok, "Online Control Barrier Functions for Decentralized Multi-Agent Navigation," in *International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, 2023, pp. 107–113.

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

[34] M. Ahmadi, A. Singletary, J. W. Burdick, and A. D. Ames, "Safe Policy Synthesis in Multi-Agent POMDPs via Discrete-Time Barrier Functions," in *IEEE Conference on Decision and Control (CDC)*, 2019, pp. 4797–4803.

[35] Y. Xiong, D.-H. Zhai, M. Tavakoli, and Y. Xia, "Discrete-Time Control Barrier Function: High-Order Case and Adaptive Case," *IEEE Transactions on Cybernetics*, vol. 53, no. 5, pp. 3231–3239, 2023.

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

[37] J. Zeng, B. Zhang, and K. Sreenath, "Safety-Critical Model Predictive Control with Discrete-Time Control Barrier Function," in *American Control Conference (ACC)*, 2021, pp. 3882–3889.

[38] A. Thirugnanam, J. Zeng, and K. Sreenath, "Safety-Critical Control and Planning for Obstacle Avoidance between Polytopes with Control Barrier Functions," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2022, pp. 286–292.

[39] R. M. Bena, C. Zhao, and Q. Nguyen, "Safety-Aware Perception for Autonomous Collision Avoidance in Dynamic Environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 12, pp. 7962–7969, 2023.

[40] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models," in *Neural Information Processing Systems (NeurIPS)*, 2018.

[41] J. Buckman, D. Hafner, G. Tucker, E. Brevdo, and H. Lee, "Sample-Efficient Reinforcement Learning with Stochastic Ensemble Value Expansion," in *Neural Information Processing Systems (NeurIPS)*, 2018.

[42] T. Kim, J. Mun, J. Seo, B. Kim, and S. Hong, "Bridging Active Exploration and Uncertainty-Aware Deployment Using Probabilistic

Ensemble Neural Network Dynamics," in *Robotics: Science and Systems (RSS)*, 2023.

[43] A. Rényi, "On measures of entropy and information," in *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability*, 1961, pp. 547–562.

[44] F. Wang, T. Syeda-Mahmood, B. C. Vemuri, D. Beymer, and A. Rangarajan, "Closed-Form Jensen-Renyi Divergence for Mixture of Gaussians and Applications to Group-Wise Shape Registration," in *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, vol. 5761, 2009, pp. 648–655.

[45] A. Majumdar and M. Pavone, "How Should a Robot Assess Risk? Towards an Axiomatic Theory of Risk in Robotics," in *Robotics Research*, N. M. Amato, G. Hager, S. Thomas, and M. Torres-Torriti, Eds. Cham: Springer International Publishing, 2020, pp. 75–84.

[46] H. Rahimian and S. Mehrotra, "Distributionally Robust Optimization: A Review," *Open Journal of Mathematical Optimization*, vol. 3, pp. 1–85, 2022.

[47] K. Ryu and N. Mehr, "Integrating Predictive Motion Uncertainties with Distributionally Robust Risk-Aware Control for Safe Robot Navigation in Crowds," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[48] J. S. Grover, C. Liu, and K. Sycara, "Deadlock Analysis and Resolution for Multi-robot Systems," in *International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, S. M. LaValle, M. Lin, T. Ojala, D. Shell, and J. Yu, Eds., 2021, pp. 294–312.